eZ80® CPU

User Manual

UM007713-0306

ZiLOG Worldwide Headquarters • 532 Race Street • San Jose, CA 95126
Telephone: 408.558.8500 • Fax: 408.558.8300 • www.zilog.com

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.zilog.com

Document Disclaimer

UM007713-0306

Table of Contents

List of Figures

List of Tables

Manual Objectives

This user manual describes the architecture and instruction set of the eZ80® CPU User Manual.

About This Manual

ZiLOG recommends you to read all the chapters and instructions provided in this manual before using the software.

Intended Audience

This document is written for ZiLOG customers who are experienced at working with microcontrollers or in writing assembly code or compilers.

Manual Organization

The eZ80® CPU User Manual is divided into twelve sections; each section details a specific topic about the product.

Introduction

This chapter provides an introduction to eZ80® CPU, ZiLOG's next-generation processor core.

Architectural Overview

This chapter provides an overview of eZ80® CPU's features and benefits, and a description of the eZ80® processor.

Memory Modes

This chapter describes eZ80®'s two memory modes: ADL and Z80.

Registers and Bit Flags

This chapter provides register and bit descriptions for ADL and Z80 modes.

Memory Mode Switching

This chapter provides description of switching capability between ADL and Z80 modes.

Interrupts

This chapter describes interrupt operation in maskable and nonmaskable mixed memory modes.

Illegal Instruction Traps

This chapter describes the consequences of undefined operations.

I/O Space

This chapter describes input/output memory for on- and off-chip peripherals.

Addressing Modes

This chapter describes methods of accessing different addressing modes.

Mixed-Memory Mode Applications

This chapter describes the MADL control bit and mixed memory mode guidelines.

CPU Instruction Set

This chapter lists assembly language instructions, including mnemonic definitions and a summary of the eZ80® CPU instruction set.

Opcode Maps

This chapter provides a detailed diagram of each opcode segment.

Related Documents

eZ80190	eZ80190 Product Specification	PS0066
	eZ80190 Module Product Specification	PS0191
eZ80L92	eZ80L92 Product Specification	PS0130
	eZ80L92 Module Product Specification	PS0170
eZ80F92	eZ80F92 Product Specification	PS0153
	eZ80F92 Ethernet Module Product Specification	PS0186
	eZ80F92 Flash Module Product Specification	PS0189
eZ80F91	eZ80F91 Product Specification	PS0192
	eZ80F91 Module Product Specification	PS0193

Manual Conventions

The following conventions are used to provide clarity in the document.

Courier Typeface

Commands, code lines and fragments, bits, equations, hexadecimal addresses, and various executable items are distinguished from general text by the use of the `Courier` typeface. Where the use of the font is not indicated, as in the Index, the name of the entity is presented in upper case.

- Example: FLAGS[1] is `smrf`.

Hexadecimal Values

Hexadecimal values are designated by a lowercase *h* and appear in the `Courier` typeface.

- Example: STAT is set to `F8h`.

Brackets

The square brackets, [ ], indicate a register or bus.

- Example: for the register REG1[7:0], REG1 is an 8-bit register, REG1[7] is the msb, and REG1[0] is the lsb.

Braces

The curly braces, { }, indicate a single register or bus created by concatenating some combination of smaller registers, or buses.

- Example: the 24-bit register {`00h`, REG1[7:0], REG2[7:0]} is composed of an 8-bit hexadecimal value (`00h`) and two 8-bit registers, REG1 and REG2. `00h` is the MSB of the 24-bit register, and REG2 is the LSB of the 24-bit register.

Parentheses

The parentheses, ( ), indicate an indirect register address lookup.

- Example: (BC) is the memory location referenced by the address contained in the BC register.

Parentheses/Bracket Combinations

The parentheses, ( ), indicate an indirect register address lookup and the square brackets, [ ], indicate a register or bus.

- Example: assume BC[15:0] contains the value `1234h`. ({`37h`, BC[15:0]}) then refers to the contents of the memory location at address `371234h`.

Use of the Words *Set* and *Clear*

The words *set* and *clear* imply that a register bit or a condition contains a logical 1 and a logical 0, respectively. When either of these terms is followed by a number, the word *logical* may not be included; however, it is implied.

Use of the Terms *LSB* and *MSB*

In this document, the terms *LSB* and *MSB,* when appearing in upper case, mean *least significant byte* and *most significant byte*, respectively. The lowercase forms, *msb* and *lsb*, mean *least significant bit* and *most significant bit*, respectively.

Use of Initial Uppercase Letters

Initial uppercase letters designate settings, modes, and conditions in general text.

- Example 1: The Slave receiver leaves the data line High.
- Example 2: The receiver forces the SCL line to Low.
- Example 3: The Master can generate a Stop condition to abort the transfer.

Use of All Uppercase Letters

The use of all uppercase letters designates the names of states, modes, and commands.

- Example 1: The bus is considered BUSY after the Start condition.
- Example 2: In TRANSMIT mode, the byte is sent most significant bit first.
- Example 3: A START command triggers the processing of the initialization sequence.

Register Access Abbreviations

Register access is designated by the following abbreviations:

Designation	Description
R	Read Only
R/W	Read/Write
W	Write Only
—	Unspecified or indeterminate

Bit Numbering

Bits are numbered from *0* to *n–1*.

Safeguards

It is important that you understand the following safety terms, which are defined here.

Caution: Means a procedure or file may become corrupted if you do not follow directions.

Trademarks

eZ80®, Z80® and Z180 are trademarks of ZiLOG, Inc.

Introduction

The eZ80® CPU is a high-speed, 8-bit microcontroller capable of executing code four times faster than a standard Z80 operating at the same clock speed. The increased processing efficiency of the eZ80® CPU improves available bandwidth and decrease power consumption. The eZ80® CPU's 8-bit processing power rivals the performance of competitors' 16-bit microcontrollers.

The eZ80® CPU is also the first 8-bit microcontroller to support 16 MB linear addressing. Each software module, or each task, under a real-time executive or operating system can operate in Z80-compatible (64 KB) mode or full 24-bit (16 MB) address mode.

The eZ80® CPU's instruction set is a superset of the instruction sets for the Z80 and Z180 CPUs. The Z80 and Z180 programs are executed on an eZ80® CPU with little or no modification.

The eZ80® CPU is combined with peripherals, I/O devices, volatile and nonvolatile memory, etc., for various eZ80® CPU products within the eZ80® and eZ80Acclaim!® product lines. Refer to the *eZ80® and eZ80Acclaim!® product specifications* for more information on these products.[1]

1. The term eZ80® CPU is referred to as CPU in this document.

Architectural Overview

The eZ80® CPU is ZiLOG's next-generation Z80 processor core. It is the basis of a new family of integrated microcontrollers and includes the following features:

- Upward code-compatible from Z80 and Z180 products.
- Several address-generation modes, including 24-bit linear addressing.
- 24-bit registers and ALU.
- 8-bit data path.
- Single-cycle fetch.
- Pipelined fetch, decode, and execute.

Processor Description

The eZ80® CPU is an 8-bit microcontroller that performs certain 16- or 24-bit operations. A simplified block diagram of the CPU is illustrated in Figure 1. Understanding the separation between the control block and the data block is helpful toward understanding the two eZ80® memory modes—Z80 mode and ADDRESS AND DATA LONG (ADL) mode.

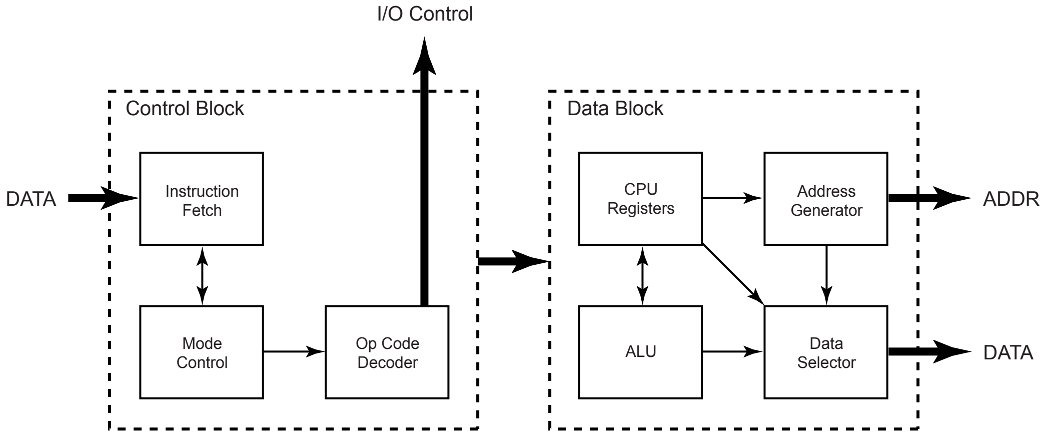


Figure 1. eZ80® CPU Block Diagram

Instruction Fetch

The instruction fetch block contains a state machine which controls the READs from memory. It fetches opcodes and operands and keeps track of the start and end of each instruction. An instruction fetch block stores opcodes during external memory READs and WRITEs. It also discards prefetched instructions when jumps, interrupts, and other control transfer events occur.

Mode Control

The Mode Control block of the CPU controls which mode the processor is currently operating in: HALT mode, SLEEP mode, Interrupt mode, debug mode, and ADL mode[1].

1. The debug interface is discussed in greater detail in the *eZ80® product specification and eZ80Acclaim!® product specification.*

Opcode Decoder

The opcodes are decoded within the CPU control block. After each instruction is fetched, it is passed to the decoder. The opcode decoder is organized similarly to a large microcoded ROM.

CPU Registers

The CPU registers are contained within the CPU's data block. Some are special purpose registers, such as the Program Counter, the Stack Pointer, and the Flags register. There are also a number of CPU control registers.

ALU

The arithmetic logic unit (ALU) is contained within the CPU's data block. The ALU performs the arithmetic and logic functions on the addresses and the data passed over from the control block or from the CPU registers.

Address Generator

The address generator creates the addresses for all CPU memory READ and WRITE operations. The address generator also contains the Z80 Memory Mode Base Address register (MBASE) for address translation in Z80 mode operation.

Data Selector

The data selector places the appropriate data onto the data bus. The data selector controls the data path based on the instruction currently being executed.

Pipeline Description

The CPU pipeline reduces the overall cycle time for each instruction. In principle, each instruction must be fetched, decoded, and executed. This process normally spans at least three cycles. The CPU pipeline, however, can reduce the overall time of some instructions to as little as one cycle by allowing the next instruction to be prefetched and decoded while it executes the current instruction as illustrated in Figure 2. The CPU operates on multiple instructions simultaneously to improve operating efficiency.

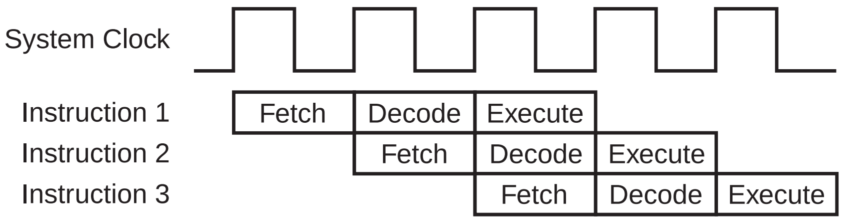


Figure 2. Pipeline Overview

In Figure 3, the pipelining process is demonstrated using a series of instructions. The first **LD** instruction prefetches its opcode and first operand during the decode and execute phases of the preceding **INC** instruction. However, the second **LD** instruction in the sequence only prefetches its opcode. The bus WRITE during the execute phase of the first **LD** instruction prevents the pipeline from prefetching the first operand of the next instruction. Thus, the number of bytes prefetched is a function of the command currently executing in the CPU.

When a control transfer takes place, the Program Counter (PC) does not progress sequentially. Therefore, the pipeline must be flushed. All prefetched values are ignored. Control transfer can occur because of an interrupt or during execution of a Jump (**JP**), **CALL**, Return (**RET**), Restart (**RST**), or similar instruction. After the control transfer instruction is executed, the pipeline must start over to fetch the next operand.

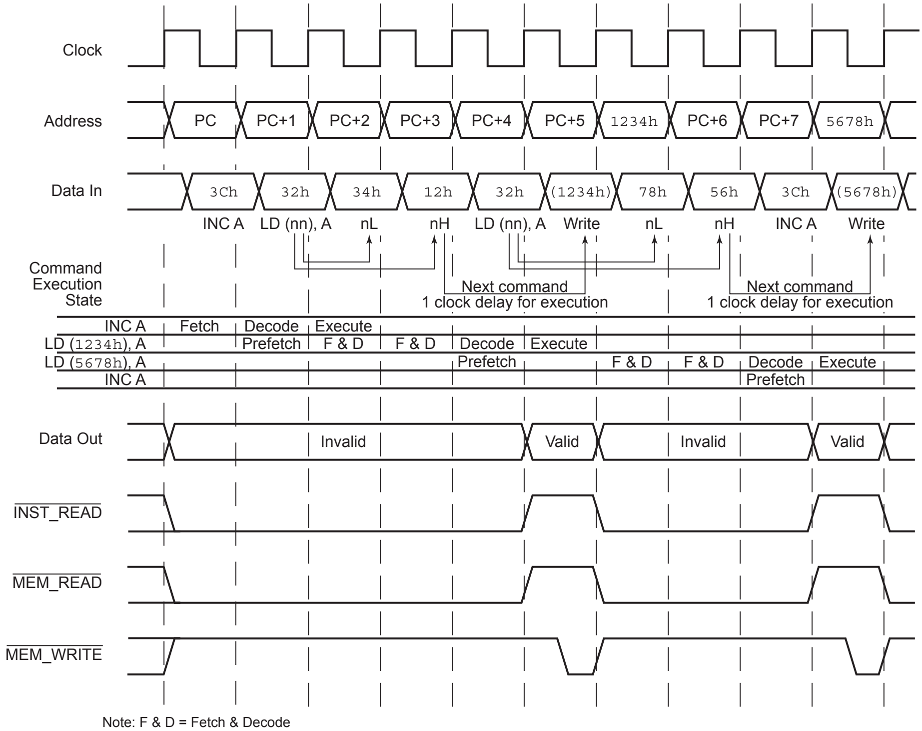


Figure 3. Pipeline Example

Memory Modes

The eZ80® CPU is capable of operating in two memory modes: Z80 mode and ADL mode. For backward compatibility with legacy Z80 programs, the CPU operates in Z80 MEMORY mode with 16-bit addresses and 16-bit CPU registers. For 24-bit linear addressing and 24-bit CPU registers, the CPU operates in ADDRESS AND DATA LONG (ADL) mode. Selection of the memory mode is controlled by the ADL mode bit.

The multiple memory modes of the processor allow CPU products to easily mix existing Z80 code or Z180 code with new ADL mode code. Collectively, the Z80 and ADL memory modes may be referred to as ADL modes, because they are controlled by the ADL bit.

Z80 MEMORY Mode

When the ADL bit is cleared to 0, the CPU operates using Z80-compatible addressing and Z80-style, 16-bit CPU registers. This Z80 MEMORY mode is also occasionally referred to as non-ADL mode. Z80 MEMORY mode is the default operating mode on reset.

In Z80 MEMORY mode (or its alternate term, Z80 mode), all of the multibyte internal CPU registers are 16 bits. Also, the 16-bit Stack Pointer Short (SPS) register is used to store the stack pointer value.

In addition, the CPU employs an 8-bit MBASE address register that is always prepended to the 16-bit Z80 mode address. The complete 24-bit address is returned by {MBASE, ADDR[15:0]}. The MBASE address register allows Z80 code to be placed anywhere within the available 16 MB addressing space. This placement allows for 256 unique Z80 code blocks within the 16 MB address space, as illustrated in Figure 4.

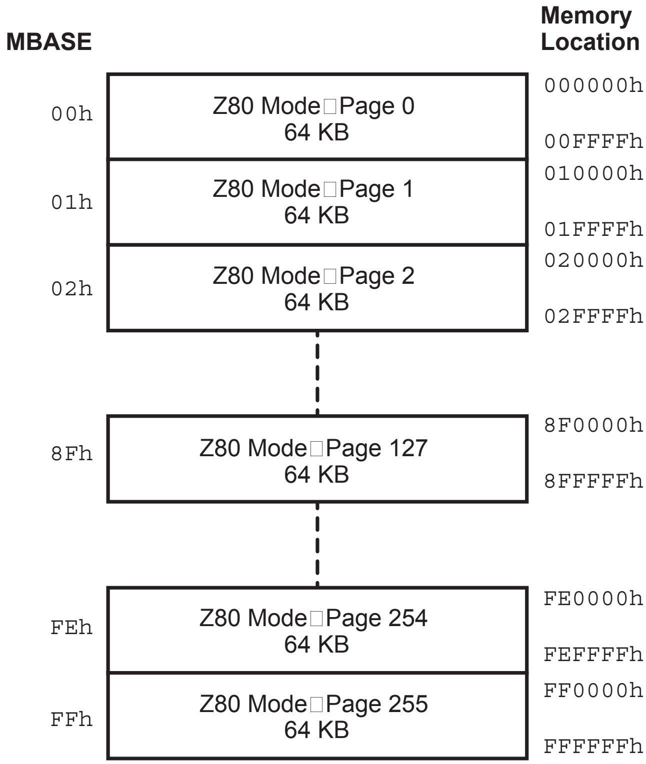


Figure 4. Z80 MEMORY Mode Map

When MBASE is set to `00h`, the CPU operates like a classic Z80 with 16-bit addressing from `0000h` to `00FFh`. When MBASE is set to a non-zero value, the 16-bit Z80-style addresses are offset to a new page, as defined by MBASE.

By altering MBASE, multiple Z80 tasks can possess their own individual Z80 partitions. The MBASE register can only be changed while in ADL mode, thereby preventing accidental page switching when operating in Z80 MEMORY mode. The MBASE address register does not affect the length of the CPU register. In Z80 mode, the CPU registers remain 16 bits, independent of the value of MBASE. For more information on the CPU registers in Z80 mode, see the eZ80® CPU Registers in Z80 Mode section on page 14.

ADL MEMORY Mode

Setting the ADL bit to 1 selects ADL mode. This memory mode is referred to as ADL MEMORY mode or ADL mode. In ADL mode, the user application can take advantage of the CPU's 16MB linear addressing space, 24-bit CPU registers, and enhanced instruction set. When ADL mode is selected, MBASE does not affect memory addressing. Figure 5 illustrates the ADL mode memory map.

Note: There are no pages in ADL mode.

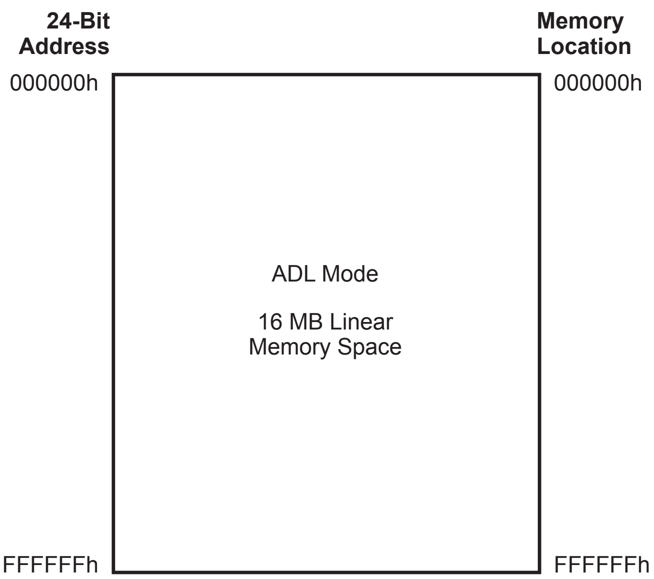


Figure 5. ADL Addressing Mode Memory Map

In ADL mode, the CPU's multibyte registers are expanded from 16 to 24 bits. A 24-bit Stack Pointer Long (SPL) register replaces the 16-bit Stack Pointer Short (SPS) register. For more information on the CPU registers in ADL mode, see the eZ80® CPU Registers in ADL Mode section on page 16.

In ADL mode, all addresses and data are 24 bits. All data READ and WRITE operations pass 3 bytes of data to and from the CPU when operating in ADL mode (as opposed to only 2 bytes of data while in Z80 mode operation). Thus, instructions operating in ADL mode may require more clock cycles to complete than in Z80 mode. Although MBASE does not affect operation during ADL mode, the MBASE register can only be written to when operating in ADL mode.

Registers and Bit Flags

eZ80® CPU Working Registers

The CPU contains two banks of working registers—the main register set and the alternate register set. The main register set contains the 8-bit accumulator register (A) and six 8-bit working registers (B, C, D, E, H, and L). The six 8-bit working registers can be combined to function as the multibyte register pairs BC, DE, and HL. The 8-bit Flag register F completes the main register set.

Similarly, the alternate register set also contains an 8-bit accumulator register (A') and six 8-bit working registers (B', C', D', E', H', and L'). These six 8-bit alternate working registers can also be combined to function as the multibyte register pairs BC', DE', and HL'. The 8-bit Flag register F' completes the alternate register set.

High-speed exchange between these two register banks is performed. See the EX and EXX instructions on pages 175 through 179 for directions on exchanging register bank contents. High-speed exchange between these banks can be used by a single section of application code. Alternatively, the main program could use one register bank while the other register banks are allocated to interrupt service routines.

eZ80® CPU Control Register Definitions

In addition to the two working register sets described in the previous section, the CPU contains several registers that control CPU operation.

- Interrupt Page Address Register (I)—the 16-bit I register stores the upper 16 bits of the interrupt vector table address for Mode 2 vectored interrupts.

Note: The 16-bit I register is not supported on the eZ80190, eZ80L92, or eZ80F92/F93 devices.

- Index Registers (IX and IY)—the multibyte registers IX and IY allow standard addressing and relative displacement addressing in memory. Many instructions employ the IX and IY registers for relative addressing in which an 8-bit two's-complement displacement (**d**) is added to the contents of the IX or IY register to generate an address. Additionally, certain 8-bit opcodes address the High and Low bytes of these registers directly. For Index Register IX, the High byte is indicated by IXH, while the Low byte is indicated by IXL. Similarly, for Index Register IY, the High byte is indicated by IYH, while the Low byte is indicated by IYL.
- Z80 Memory Mode Base Address (MBASE) register—the 8-bit MBASE register determines the page of memory currently employed when operating in Z80 mode. The MBASE register is only used during Z80 mode. However, the MBASE register can only be altered from ADL mode.
- Program Counter (PC) register—the multibyte Program Counter register stores the address of the current instruction being fetched from memory. The Program Counter is automatically incremented during program execution. When a program jump occurs, the new value is placed in the Program Counter, overriding the incremented value. In Z80 mode, the Program Counter is only 16 bits; however, a full 24-bit address {MBASE,PC[15:0]}, is used. In ADL mode, the Program Counter is returned by {PC[23:0]}.
- Refresh Counter (R) register—the Refresh Counter register contains a count of executed instruction fetch cycles. The 7 least significant bits (lsb) of the R register are automatically incremented after each instruction fetch. The most significant bit (msb) can only be changed by writing to the R register. The R register can be read from and written to using dedicated instructions **LD A,R** and **LD R,A**, respectively.
- Stack Pointer Long (SPL) register—in ADL mode, the 24-bit Stack Pointer Long stores the address for the current top of the external

stack. In ADL mode, the stack can be located anywhere in memory. The external stack is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack or popped off of the stack using the **PUSH** and **POP** instructions. Interrupts, traps, calls, and returns also employ the stack.

- Stack Pointer Short register (SPS)—in Z80 mode, the 16-bit Stack Pointer Short stores the address for the current top of the stack. In Z80 mode, the stack can be located anywhere within the current Z80 memory page. The current Z80 memory page is selected by the MBASE register. The 24-bit Stack Pointer address in Z80 mode is {MBASE, SPS}. The stack is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack or popped off of the stack using the **PUSH** and **POP** instructions. Interrupts, traps, calls, and returns also employ the stack.

eZ80® CPU Control Bits

- Address and Data Long Mode Bit (ADL)—the ADL mode bit indicates the current memory mode of the CPU. An ADL mode bit reset to 0 indicates that the CPU is operating in Z80 MEMORY mode with 16-bit Z80-style addresses offset by the 8-bit MBASE register. An ADL mode bit set to 1 indicates that the CPU is operating in ADL mode with 24-bit linear addressing. The default for the ADL mode bit is reset (cleared to 0). The ADL mode bit can only be changed by those instructions that allow persistent memory mode changes, interrupts, and traps. The ADL mode bit cannot be directly written to.
- Mixed-ADL Bit (MADL)—the MADL control bit is used to configure the CPU to execute programs containing code that uses both ADL and Z80 MEMORY modes. The MADL control bit is explained in more detail in the Interrupts in Mixed Memory Mode Applications section of the Interrupts chapter, on page 49. An additional explanation is available in the Mixed-Memory Mode Applications chapter, on page 45.

- Interrupt Enable Flags (IEF1 and IEF2)—in the CPU, there are two interrupt enable flags that are set or reset using the Enable Interrupt (**EI**) and Disable Interrupt (**DI**) instructions. When IEF1 is reset to 0, a maskable interrupt cannot be accepted by the CPU. The Interrupt Enable flags are described in more detail in the Interrupts chapter, on page 48.

eZ80® CPU Registers in Z80 Mode

In Z80 mode, the BC, DE, and HL register pairs and the IX and IY registers function as 16-bit registers for multibyte operations and indirect addressing. The active Stack Pointer is the 16-bit Stack Pointer Short register (SPS). The Program Counter register (PC) is also 16 bits long. The address is 24 bits long and is composed as {MBASE, ADDR[15:0]}. While the MBASE register is only used during Z80 mode operations, it cannot be written while operating in this mode. Tables 1 and 2 lists the CPU registers and bit flags during Z80 mode operation.

Caution: In Z80 mode, the upper byte (bits 23:16) of each multibyte register is undefined. When performing 16-bit operations with these registers, the application program cannot assume values or behavior for the upper byte. The upper byte is only valid in ADL mode.

Note: In Z80 mode, the upper byte of the I register, bits [15:8], is not used.

Table 1. CPU Working Registers in Z80 Mode

Main Register Set				Alternate Register Set			
8-Bit Registers				**8-Bit Registers**			
A				A’			
F				F’			
Individual 8-Bit Registers		**Or**	**16-Bit Registers**	**Individual 8-Bit Registers**		**Or**	**16-Bit Registers**
B	C		BC	B’	C’		BC’
D	E		DE	D’	E’		DE’
H	L		HL	H’	L’		HL’

Table 2. CPU Control Registers and Bit Flags in Z80 Mode

8-Bit Registers	16-Bit Registers	Single-Bit Flags
I	SPS	ADL
MBASE	PC	MADL
R		IEF1
		IEF2

Individual 8-Bit Registers		Or	16-Bit Registers
IXH	IXL		IX
IYH	IYL		IY

eZ80® CPU Registers in ADL Mode

In ADL mode, the BC, DE, HL, IX and IY registers are 24 bits long for multibyte operations and indirect addressing. The most significant bytes (MSBs) of these 3 multibyte registers are designated with a *U* to indicate the upper byte. For example, the upper byte of multibyte register BC is designated BCU. Thus, the 24-bit BC register in ADL mode is composed of the three 8-bit registers {BCU, B, C}. Likewise, the upper byte of the IX register is designated IXU. The 24-bit IX register in ADL mode is composed of the three 8-bit registers {IXU, IXH, IXL}.

Note: None of the upper bytes (BCU, DEU, IXU, etc.) are individually accessible as standalone 8-bit registers.

MBASE is not used for address generation in ADL mode; however, it can only be written in ADL mode. The Program Counter is 24 bits long, as is SPL. IEF1, IEF2, ADL, and MADL are single bit flags.

The CPU registers and bit flags during Z80 mode operation are indicated in Tables 3 and 4. Reset states are detailed in Table 5.

Table 3. CPU Working Registers in ADL Mode

Main Register Set					Alternate Register Set				
8-Bit Registers					8-Bit Registers				
A					A’				
F					F’				
Individual 8-Bit Registers				24-Bit Registers	Individual 8-Bit Registers				24-Bit Registers
BCU	B	C	Or	BC	BCU’	B’	C’	Or	BC’
DEU	D	E		DE	DEU’	D’	E’		DE’
HLU	H	L		HL	HLU’	H’	L’		HL’

Table 4. CPU Control Registers and Bit Flags in ADL Mode

Control Registers and Bit Flags		
8-Bit Registers	**24-Bit Registers**	**Single-Bit Flags**
I	SPL	ADL
MBASE	PC	MADL
R		IEF1
		IEF2

	Individual 8-Bit Registers		Or	24-Bit Registers
IXU	IXH	IXL		IX
IYU	IYH	IYL		IY

Table 5. CPU Register and Bit Flag Reset States

	CPU Register or Bit Flag	Reset State
8-Bit Working Registers	A, A’	Undefined
	B, B’	Undefined
	C, C’	Undefined
	D, D’	Undefined
	E, E’	Undefined
	F, F’	Undefined
	H, H’	Undefined
	L, L’	Undefined
Upper Bytes of 24-Bit Multibyte Working Registers	BCU	Undefined
	DEU	Undefined
	HLU	Undefined
8-Bit Control Registers	I	`00h`
	IXH	`00h`

Table 5. CPU Register and Bit Flag Reset States (Continued)

	CPU Register or Bit Flag	Reset State
	IXL	00h
	IYH	00h
	IYL	00h
	MBASE	00h
	R	00h
Upper Bytes of 24-Bit Multibyte Control Registers	IXU	00h
	IYU	00h
16- and 24-Bit Control Registers	PC	000000h
	SPS	0000h
	SPL	000000h
Single-Bit Flags	ADL	0
	IEF1	0
	IEF2	0
	MADL	0

eZ80® CPU Status Indicators (Flag Register)

The Flag register (F and F’) contains status information for the CPU. The bit position for each flag is indicated in Table 6.

Table 6. Flag Register Bit Positions

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Flag	S	Z	X	H	X	P/V	N	C

where:

C=Carry Flag
N=Add/Subtract Flag

P/V=Parity/Overflow Flag
H=Half-Carry Flag
Z=0 Flag
S=Sign Flag
X=Not used

Each of the two CPU flag registers contain six bits of status information that are set or reset by CPU operations. Bits 3 and 5 are not used. Four of these bits are testable (C, P/V, Z and S) for use with conditional **jump**, **call** or **return** instructions. Two flags are not testable (H, N) and are used for BCD arithmetic.

Carry Flag (C)

The Carry Flag bit is set or reset, depending on the operation that is performed. For **ADD** instructions that generate a carry and **SUBTRACT** instructions that generate a borrow, the Carry flag is set to 1. The Carry flag is reset by an **ADD** that does not generate a carry, and a subtract that does not generate a borrow. This saved carry facilitates software routines for extended precision arithmetic. Also, the **DAA** instruction sets the Carry flag to 1 if the conditions for making the decimal adjustment are met.

For the **RLA**, **RRA**, **RLC** and **RRC** instructions, the Carry flag is used as a link between the least significant bit (lsb) and most significant bit (msb) for any register or memory location. During the **RLCA**, **RLC m** and **SLA m** instructions, the carry contains the last value shifted out of bit 7 of any register or memory location. During the **RRCA**, **RRC m**, **SRA m** and **SRL m** instructions, the carry contains the last value shifted out of bit 0 of any register or memory location. For the logical instructions **AND A s**, **OR A s**, and **XOR A s**, the carry is reset. The Carry flag can also be set (**SCF**) and complemented (**CCF**).

Add/Subtract Flag (N)

The Add/Subtract (N) flag is used by the decimal adjust accumulator instructions (**DAA**) to distinguish between **ADD** and **SUBTRACT** instructions. For all **ADD** instructions, N is set to 0. For all **SUBTRACT** instructions, N is set to 1.

Parity/Overflow Flag (P/V)

The Parity/Overflow (P/V) flag is set or reset, depending on the operation that is performed. For arithmetic operations, this flag indicates an overflow condition when the result in the accumulator is greater than the maximum possible number (+127) or is less than the minimum possible number (–128). This overflow condition can be determined by examining the sign bits of the operands.

For addition, operands with different signs never causes overflow. When adding operands with like signs where the result yields a different sign, the overflow flag is set to 1, as indicated in Table 7.

Table 7. Overflow Flag Addition Settings

+120	=	0111	1000	ADDEND
+105	=	0110	1001	AUGEND
+225		1110	0001	(–95) SUM

The two numbers added together result in a number that exceeds +127 and the two positive operands result in a negative number (–95), which is incorrect. Thus, the Overflow flag is set to 1.

For subtraction, overflow can occur for operands of unlike signs. Operands of like signs never causes overflow, as indicated in Table 8.

Table 8. Overflow Flag Subtraction Settings

	+127	0111	1111	MINUEND
(–)	–64	1100	0000	SUBTRAHEND
	+191	1011	1111	DIFFERENCE

The minuend sign is changed from positive to negative, returning an incorrect difference. Thus, overflow is set to 1. Another method for predicting an overflow is to observe the carry into and out of the sign bit. If there is a carry in and no carry out, then overflow occurs.

This flag is also used with logical operation and rotate instructions to indicate the parity of the result. The number of 1 bits in a byte are counted. If the total is odd, then odd parity (P=0) is flagged. If the total is even, then even parity (P=1) is flagged.

During search instructions (**CPI**, **CPIR**, **CPD**, **CPDR**) and block transfer instructions (**LDI**, **LDIR**, **LDD**, **LDDR**), the P/V flag monitors the state of the byte count register (BC). When decrementing, the byte counter results in a 0 value and the flag is reset to 0; otherwise the flag is logical 1.

During **LD A, I** and **LD A, R** instructions, the P/V flag is set to 1 with the contents of the interrupt enable flip-flop (IEF2) for storage or testing. When inputting a byte from an I/O device, **IN r,(C)**, the flag is adjusted to indicate the parity of the data.

The P/V flag is set to 1 to indicate even parity, and cleared to 0 to indicate odd parity.

Half-Carry Flag (H)

The Half-Carry flag (H) is set or reset, depending on the carry and borrow status between bits 3 and 4 of an 8-bit arithmetic operation. This flag is used by the decimal adjust accumulator instruction (**DAA**) to correct the result of a packed BCD addition or subtraction. The H flag is set to 1 or reset to 0, as indicated in Table 9.

Table 9. H Flag Settings

H	ADD	SUBTRACT
1	There is a carry from bit 3 to bit 4	There is a borrow from bit 4.
0	There is no carry from bit 3 to bit 4	There is no borrow from bit 4.

Zero Flag (Z)

The Zero flag (Z) is set to 1 if the result generated by the execution of certain instructions is 0. For 8-bit arithmetic and logical operations, the Z flag is set to 1 if the resulting byte in the accumulator is 0. If the byte is not 0, the Z flag is reset to 0.

For compare instructions, the Z flag is set to 1 if the value in the accumulator is the same as the data it is being compared against. When testing a bit in a register or memory location, the Z flag contains the complemented state of the indicated bit (see the **BIT b, r** instruction.

When inputting or outputting a byte between a memory location and an I/O device (for example, **INI**, **IND**, **OUTI** and **OUTD**), the B register is decremented. If the result of this decrement is 0 (that is, B–1 = 0), then the Z flag is set to 1. Otherwise, the Z flag is reset (cleared to 0). Also, for byte inputs from I/O devices using **IN r,(C)**, the Z flag is set to 1 to indicate a zero-byte input.

Sign Flag (S)

The Sign flag stores the state of the most significant bit of the accumulator (bit 7). When the CPU performs arithmetic operations on signed numbers, binary two's-complement notation is used to represent and process numerical information. A positive number is identified by a 0 in bit 7. A negative number is identified by a 1.

The binary equivalent of the magnitude of a positive number is stored in bits 0–6 for a total range of 0–127. A negative number is represented by the two's-complement of the equivalent positive number. The total range for negative numbers is –1 to –128.

When inputting a byte from an I/O device to a register, **IN r,(C)**, the S flag indicates either positive (S=0) or negative (S=1) data.

Memory Mode Switching

ADL Mode and Z80 Mode

The CPU is capable of easily switching between the two available memory modes (ADL mode and Z80 mode). There are two types of mode changes available to the CPU: persistent and single-instruction. For example, persistent mode switches allow the CPU to operate indefinitely in ADL mode, then switch to Z80 mode to run a section of Z80 code, and then return to ADL mode. Conversely, single-instruction mode changes allow certain instructions to operate using either addressing mode without making a persistent change to the mode.

Memory Mode Compiler Directives

In the ZiLOG ZMASM/ZDS assembler, the application code is assembled for a given state of the ADL mode bit by placing one of the two following compiler directives at the top of the code:

```
.ASSUME ADL=1
.ASSUME ADL=0
```

These compiler directives indicate that either ADL MEMORY mode (ADL=1) or Z80 MEMORY mode (ADL=0) is the default memory mode for the code being currently compiled. The code developer is responsible for ensuring that this source file setting matches the state of the hardware ADL mode bit when the code is executed.

Opcode Suffixes for Memory Mode Control

When developing application code for CPU applications, care must be taken when manipulating the ADL and Z80 memory modes. Special opcode suffixes are added to the instruction set to assist with memory mode switching operations. There are four individual suffixes available for use: **.SIS**, **.SIL**, **.LIS**, and **.LIL**. These suffixes are appended to many instructions to indicate that a memory mode change or an exception to standard memory mode operation is being requested.

Even with the compiler directives described in the Memory Mode Compiler Directives section on page 24, the code developer must still employ these opcode suffixes to allow exceptions to the default memory mode. For example, the opcode suffixes can be used to allow persistent memory mode switching between ADL and Z80 modes. In addition, there may be times when ADL mode code may fetch a 16-bit address generated from a section of Z80 mode code. Alternatively, a section of Z80 mode code may retrieve immediate data created by a section of ADL mode code. The memory mode control suffixes facilitate these requirements.

Each of the four suffixes **.SIS**, **.SIL**, **.LIS**, and **.LIL** is composed of 2 parts that define the operation in the control block and the data block within the CPU (see the eZ80® CPU Block Diagram on page 3 and Table 10, below). The first part of the suffix, either Short (**.S**) or Long (**.L**), directs operations within the data block of the CPU. **.S** and **.L** control whether the overall operation of the instruction and the internal registers should use 16 or 24 bits. The **.S** and **.L** portions of the suffix also indicate if MBASE is used to define the 24-bit address. The last part of the suffix, either **.IS** or **.IL**, directs the control block within the CPU. The Instruction Stream Short and Instruction Stream Long suffixes, **.IS** and **.IL**, control whether a multibyte immediate data or address value fetched during instruction execution is 2 or 3 bytes long (for example, a **LD HL, Mmn** instruction versus a **LD HL, mn** instruction). The CPU must know whether to fetch 3 bytes (**Mmn**) or 2 bytes (**mn**) of data. The **.IS** and **.IL** portions of the suffix tell the CPU the length of the instruction. If the length of the instruction is unambiguous, the **.IS** and **.IL** suffixes yield no effect.

Table 10. Opcode Suffix Description

Full Suffix	Suffix Components	Description
.SIS	.S	The CPU data block operates in Z80 mode using 16-bit registers. All addresses use MBASE.
	.IS	The CPU control block operates in Z80 mode. For instructions with an ambiguous number of bytes, the .IS suffix indicates that only 2 bytes of immediate data or address must be fetched.
.SIL	.S	The CPU data block operates in Z80 mode using 16-bit registers. All addresses use MBASE.
	.IL	The CPU control block operates in ADL mode. For instructions with an ambiguous number of bytes, the .IL suffix indicates that 3 bytes of immediate data or address must be fetched.
.LIS	.L	The CPU data block operates in ADL mode using 24-bit registers. Addresses do not use MBASE.
	.IS	The CPU control block operates in Z80 mode. For instructions with an ambiguous number of bytes, the .IS suffix indicates that only 2 bytes of immediate data or address must be fetched.
.LIL	.L	The CPU data block operates in ADL mode using 24-bit registers. Addresses do not use MBASE.
	.IL	The CPU control block operates in ADL mode. For instructions with an ambiguous number of bytes, the .IL suffix indicates that 3 bytes of immediate data or address must be fetched.

Single-Instruction Memory Mode Changes

Often, the CPU must perform a single operation using the memory mode opposite from that currently set by the ADL mode bit. The CPU is capable of changing between ADL mode and Z80 mode for a single instruction. Certain CPU instructions can be appended with the memory mode opcode suffixes **.SIS**, **.LIL**, **.LIS**, and **.SIL** to indicate that a particular memory mode is appropriate for this instruction only. The following three examples serve to make the suffix operation for single-instruction memory mode changes more clear.

Suffix Example 1: LD HL, Mmn in Z80 Mode

In Z80 mode (ADL mode bit=0), only two bytes of immediate data are normally fetched and the upper byte of all CPU multibyte registers is undefined. Compare the operation of the following lines of code to observe the effect of the opcode suffixes.

```
.ASSUME ADL=0         ;Z80 mode operation is default.
LD HL, 3456h          ;HL[23:0] ¨ {00h, 3456h}.
LD HL, 123456h        ;Invalid—Z80 mode cannot load 24-
                      ;bit value.
LD.SIS HL, 3456h      ;Same as LD HL, 3456, because
                      ;ADL=0. HL[23:0] ¨ {00h, 3456h}.
                      ;.IS directs eZ80 to fetch only
                      ;16 bits of data.
                      ;.S forces upper byte of HL
                      ;register to an undefined state.
LD.LIL HL, 123456h    ;HL[23:0] ¨ 123456h.
                      ;.IL directs eZ80 to fetch 24-
                      ;bits of data.
                      ;.L uses all 3 bytes of HL
                      ;register.
LD.LIS HL, 3456h      ;HL[23:0] ¨ {00h, 3456h}. .IS
                      ;directs eZ80 to fetch only 16-
                      ;bits of data. .L uses all 3 bytes
                      ;of HL register.
LD.SIL HL, 123456h    ;HL[23:0] ¨ {00h, 3456h}.
                      ;.IL directs eZ80 to fetch 24 bits
                      ;of data. .S forces upper byte of
                      ;HL register to an undefined
                      ;state because registers are
                      ;defined to be only 16-bits.
```

In all cases of Suffix Example 1, the memory mode is unchanged after the operation, as it remains in Z80 mode (ADL mode bit=0) following completion of each instruction. However, during operation of the **LD.LIS**, **LD.LIL, and LD.SIL** instructions, all or parts of the CPU function temporarily in ADL mode. The **.IL** segment of the suffix forces the control block, to operate in ADL mode. The **.L** segment of the suffix forces the data block to operate in ADL mode.

Suffix Example 2: LD HL, Mmn in ADL Mode

Suffix Example 2 considers the same examples as in Suffix Example 1. However, for this example, it is assumed that the part begins in ADL mode.

```
.ASSUME ADL=1        ;ADL mode operation is default.
LD HL, 3456h         ;HL[23:0] ← 003456h.
                     ;3456h is valid 24-bit value.
                     ;Leading 0s are assumed.
LD HL, 123456h       ;HL[23:0] ← 123456h.
LD.SIS HL, 3456h     ;HL[23:0] ← {00h, 3456h}.
                     ;.IS directs the eZ80 to fetch
                     ;only 16 bits of data.
                     ;.S forces upper byte of the HL
                     ;register to an undefined state.
LD.LIL HL, 123456h   ;Same as LD HL, 123456h, because
                     ;ADL=1. HL[23:0] ←123456h.
                     ;.IL directs eZ80 to fetch 24
                     ;bits of data.
                     ;.L uses all 3 bytes of HL
                     ;register.
LD.LIS HL, 3456h     ;HL[23:0] ← {00h, 3456h}.
                     ;.IS directs eZ80 to fetch only
                     ;16 bits of data.
                     ;.L uses all 3 bytes of HL
                     ;register.
LD.SIL HL, 123456h   ;HL[23:0] ← {00h, 3456h}.
                     ;.IL directs eZ80 to fetch 24 bits
                     ;of data.
                     .S forces upper byte of HL
                     ;register to an undefined state.
```

From these two suffix examples, it can be seen that with the extensions applied, operation is consistent regardless of the persistent memory mode in operation at the time. To explain, a **LD.LIS** instruction operates in the same manner whether or not the CPU is currently operating in Z80 mode or ADL mode. The same is also true for the **LD.SIS**, **LD.SIL**, and **LD.LIL** instructions.

Suffix Example 3: Risks with Using the .SIL Suffix

As Suffix Examples 1 and 2 demonstrate, special care must be taken when using the **.SIL** suffix. Wherever possible, the **.SIL** suffix should be avoided whenever both segments of the suffix (**.S** and **.IL**) are relevant. The **.IL** segment of the suffix indicates a long direct memory address or immediate data in the instruction stream and the CPU reads the 24-bit value. Because the **.S** is active, the internal registers are treated as 16-bit registers and the upper bits (23–16) that were read from the instruction are discarded (replaced with `00h`). Additionally, all memory WRITEs use Z80 mode employing MBASE. Therefore, the upper byte of a 24-bit memory WRITE address is replaced by MBASE.

```
LD.SIL HL, 123456h  ;HL[23:0] ←{00h, 3456h}.
                    ;.IL directs eZ80 to fetch 24 bits
                    ;of data. .S forces upper byte of
                    ;HL register to an undefined
                    ;state. A different value is
                    ;stored in HL than expected.
LD.SIL (123456h), HL;(3456h) ←HL.
                    ;.IL forces a fetch of a 24-bit
                    ;indirect address. .S forces Z80
                    ;mode for writes to memory, thus
                    ;address of write is {MBASE,
                    ;3456h} rather than the address
                    ;123456h that may be expected.
```

Suffix Example 4: LD (HL), BC in Z80 Mode

The following two examples, Suffix Example 4 and Suffix Example 5, further demonstrate how the suffixes affect internal CPU register operation and the creation of addresses. In these two suffix examples, the .IS and .IL portions of the suffix have no effect because the length of this instruction is unambiguous.

```
.ASSUME ADL = 0      ;Z80 Mode operation is default.
LD (HL), BC          ;16-bit value stored in BC[15:0]
                     ;is written to the 24-bit memory
                     ;location given by
                     ;{MBASE, HL[15:0]}.
LD.SIS (HL), BC      ;16-bit value stored in BC[15:0]
                     ;is written to the 24-bit memory
                     ;location given by
                     ;{MBASE, HL[15:0]}. The .S portion
                     ;of the suffix has no effect since
                     ;already operating in Z80 Mode.
                     ;The .IS portion of the suffix has
                     ;no effect since instruction
                     ;length is unambiguous.
LD.LIL (HL), BC      ;24-bit value stored in BC[23:0]
                     ;is written to the 24-bit memory
                     ;location given by HL[23:0]. The
                     ;.L portion of the suffix forces
                     ;the use of 24-bit registers and
                     ;24-bit addresses without MBASE.
                     ;The .IL portion of the suffix has
                     ;no effect since instruction
                     ;length is unambiguous.
LD.SIL (HL), BC      ;16-bit value stored in BC[15:0]
                     ;is written to the 24-bit memory
                     ;location given by
                     ;{MBASE,HL[15:0]}. The .S portion
                     ;of the suffix has no effect since
                     ;already operating in Z80 Mode.
                     ;The .IL portion of the suffix has
```

```
                        ;no effect since instruction
                        ;length is unambiguous.
LD.LIS (HL), BC         ;24-bit value stored in BC[23:0]
                        ;is written to the 24-bit memory
                        ;location given by HL[23:0]. The
                        ;.L portion of the suffix forces
                        ;the use of 24-bit registers and
                        ;24-bit addresses without
                        ;MBASE.
                        ;The .IS portion of the suffix has
                        ;no effect since instruction
                        ;length is unambiguous.
```

Suffix Example 5: LD (HL), BC in ADL Mode

```
.ASSUME ADL = 1         ;ADL Mode operation is default.
LD (HL), BC             ;24-bit value stored in BC[23:0]
                        ;is written to the 24-bit memory
                        ;location given by HL[23:0].
LD.SIS (HL), BC         ;16-bit value stored in BC[15:0]
                        ;is written to the 24-bit memory
                        ;location given by
                        ;{MBASE,HL[15:0]}. The .S portion
                        ;of the suffix forces the use of
                        ;16-bit values from the registers
                        ;and uses MBASE with the address.
                        ;The .IS portion of the suffix has
                        ;no effect since instruction
                        ;length is unambiguous.
LD.LIL (HL), BC         ;24-bit value stored in BC[23:0]
                        ;is written to the 24-bit memory
                        ;location given by HL[23:0].
                        ;Since operating in ADL mode, the
                        ;.L suffix has no effect on this
                        ;instruction execution.
                        ;The .IL portion of the suffix has
                        ;no effect since instruction
                        ;length is unambiguous.
```

```
LD.SIL (HL), BC     ;16-bit value stored in BC[15:0]
                    ;is written to the 24-bit memory
                    ;location provided by
                    ;{MBASE,HL[15:0]}. The .S
                    ;portion of the suffix forces the
                    ;use of 16-bit registers and MBASE
                    ;with the address.
                    ;The .IL portion of the suffix has
                    ;no effect because instruction
                    ;length is unambiguous.
LD.LIS (HL), BC     ;24-bit value stored in BC[23:0]
                    ;is written to the 24-bit memory
                    ;location given by HL[23:0].
                    ;Because it is operating in ADL
                    ;Mode, the.L portion of the suffix
                    ;has no effect on this instruction
                    ;execution.
                    ;The .IS portion of the suffix has
                    ;no effect because instruction
                    ;length is unambiguous.
```

Suffix Completion by the Assembler

Ultimately, the assembler for the CPU creates one of the four full suffixes **.SIS**, **.SIL**, **.LIS**, or **.LIL**, depending on the current memory mode. Often, the programmer is not required to write the entire suffix. Partial suffixes (**.S**, **.L, .IS**, or **.IL**) can be employed. If **.S**, **.L, .IS**, or **.IL** is used by the code developer, the remainder of the full suffix is deduced from the current memory mode state. The suffix completion by the assembler is listed in Table 11.

Table 11. Opcode Suffix Completion by the Assembler

CPU Code Partial Suffix	ADL Mode Bit	Full Suffix Used by CPU Assembler
.S	0	.SIS
.S	1	.SIL
.L	0	.LIS
.L	1	.LIL
.IS	0	.SIS
.IS	1	.LIS
.IL	0	.SIL
.IL	1	.LIL

Assembly of the Opcode Suffixes

During assembly, the opcode suffixes become prefixes in the assembled code. The processor must know what type of memory mode exceptions must be applied to the instruction to follow. The four assembled prefixes that correspond to the four full suffixes are illustrated in Table 12.

Table 12. CPU Code Suffix to Assembled Prefix Mapping

CPU Code Suffix	Assembled Prefix Byte (hex)
.SIS	40
.LIS	49
.SIL	52
.LIL	5B

The assembled prefix bytes replace Z80 and Z80180 instructions that do not offer a function. If an CPU assembler encounters one of these replaced instructions, it issues a warning message and assembles it as a standard NOP (`00h`). The CPU prefix bytes are indicated in Table 13.

Table 13. Z80 Instructions Replaced by Memory Mode Suffixes

Opcode Prefix (hex)	Previous Z80 and Z180 Instruction	New CPU Suffix
`40`	**LD** B,B	**.SIS**
`49`	**LD** C,C	**.LIS**
`52`	**LD** D,D	**.SIL**
`5B`	**LD** E,E	**.LIL**

For the traditional Z80 prefix bytes, the CPU does not allow an interrupt to occur in the time between fetching one of these prefix bytes and fetching the following instruction. The traditional Z80 prefix bytes are `CBh`, `DDh`, `EDh`, and `FDh`, which indicate opcodes that are not on the first page of the opcode map. The eZ80® MEMORY mode prefix bytes (`40h`, `59h`, `52h`, `5Bh`) must precede the traditional Z80 prefix bytes.

Persistent Memory Mode Changes in ADL and Z80 Modes

The CPU can only make persistent mode switches between ADL mode and Z80 mode as part of a special control transfer instruction (**CALL**, **JP**, **RST**, **RET, RETI,** or **RETN**), or as part of an interrupt or trap operation. The Program Counter (PC) is thus prevented from making an uncontrolled jump. When the memory mode is changed in any of these ways, it remains in its new state until another of these operations changes the mode back. Persistent mode changes are ideal for calling and executing a block of Z80-style code from within a higher-level ADL mode program. Memory mode switching, using interrupts, and traps are discussed in later sections of this manual.

The memory mode can be changed by adding a suffix to a **CALL**, **JP**, **RST**, or **RET, RETI,** or, **RETN** instruction. Tables 14 through 20 describe how each of these 4 instructions function. The individual instructions may perform additional operations that are not described here. These tables are focused only on the memory mode switching. For more detailed information, see eZ80® CPU Instruction Set Description, which starts on page 106.

Table 14. CALL Mmn Instruction

User Code	ADL Mode	Assembled Code	Operation
CALL mn	0	**CALL mn** assembles to `CD nn mm`	The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address PC[15:0] onto the SPS stack. The ADL mode bit remains cleared to 0. Load 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending program counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
CALL Mmn	1	**CALL Mmn** assembles to `CD nn mm MM`	The starting program counter is PC[23:0]. Push the 3-byte return address PC[23:0] onto the SPL stack. The ADL mode bit remains set to 1. Load 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending program counter is PC[23:0] = {MM, mm, nn}.
CALL.IS mn	0	**CALL.SIS mn** assembles to `40 CD nn mm`	The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte logical return address PC[15:0] onto the {MBASE, SPS} stack. Push a `02h` byte onto the SPL stack, indicating a call from Z80 mode, (because ADL = 0). The ADL mode bit remains cleared to 0. Load 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending program counter is {MBASE, PC[15:0]}.

Table 14. CALL Mmn Instruction (Continued)

User Code	ADL Mode	Assembled Code	Operation
CALL.IS mn	1	**CALL.LIS mn** assembles to `49 CD nn mm`	The starting program counter is PC[23:0]. Push the 2 LS bytes of the return address, PC[15:0], onto the {MBASE, SPS} stack. Push the MS byte of the return address, PC[23:16], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating a call from ADL mode (because ADL = 1). Reset the ADL mode bit to 0. Load 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending program counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
CALL.IL Mmn	0	**CALL.SIL Mmn** assembles to `52 CD nn mm MM`	The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte logical return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating a call from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Load the 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending program counter is PC[23:0] = {MM, mm, nn}.
CALL.IL Mmn	1	**CALL.LIL Mmn** assembles to `5B CD nn mm MM`	The starting program counter is PC[23:0]}. Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating a call from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Load a 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending program counter is PC[23:0] = {MM, mm, nn}.

Table 15. JP Mmn Instruction

User Code	ADL Mode	Assembled Code	Operation
JP mn	0	**JP mn** assembles to `C3 nn mm`	The starting program counter is {MBASE, PC[15:0]}. Write the 2-byte immediate value {mm, nn}, to PC[15:0]. The ADL mode bit remains cleared to 0. The ending program counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
JP.SIS mn	0	**JP.SIS mn** assembles to `40 C3 nn mm`	This operation is the same as the previous operation. The **.SIS** extension does not affect operation when beginning in Z80 mode.
JP.LIL Mmn	0	**JP.LIL mn** assembles to `5B C3 nn mm`	The starting program counter is {MBASE, PC[15:0]}. Write the 3-byte immediate value {MM, mm, nn}, to PC[23:0]. Set the ADL mode bit to 1. The ending program counter is PC[23:0] = {MM, mm, nn}.
JP.SIL Mmn	0	N/A	An illegal suffix for this instruction.
JP.LIS mn	0	N/A	An illegal suffix for this instruction.
JP Mmn	1	**JP Mmn** assembles to `C3 nn mm MM`	The starting program counter is PC[23:0]. Write the 3-byte immediate value {MM, mm, nn}, to PC[23:0]. The ADL mode bit remains set to 1. The ending program counter is PC[23:0] = {MM, mm, nn}.
JP.LIL Mmn	1	**JP.LIL Mmn** assembles to `5B C3 nn mm MM`	This operation is the same as the previous operation. The **.LIL** extension does not affect operation when beginning in ADL mode.
JP.SIS mn	1	**JP.SIS mn** assembles to `40 C3 nn mm`	The starting program counter is PC[23:0]. Write the 2-byte immediate value {mm, nn}, to PC[15:0]. Reset the ADL mode bit to 0. The ending program counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
JP.SIL Mmn	1	N/A	An illegal suffix for this instruction.
JP.LIS mn	1	N/A	An illegal suffix for this instruction.

Because the CPU core resets to Z80 MEMORY mode, a **JP.LIL Mmn** is recommended for use near the beginning of source programs that run primarily in ADL MEMORY mode.

Table 16. JP (rr) Instruction

User Code	ADL Mode	Assembled Code	Operation
JP (rr)	0	**JP (rr)** assembles to `E9` or `DD/FD E9`	The starting program counter is {MBASE, PC[15:0]}. Write the 2-byte value stored in **rr**[15:0] to PC[15:0]. The ADL mode bit remains cleared to 0. The ending program counter is {MBASE, PC[15:0]} = {MBASE, **rr**[15:0]}.
JP.S (rr)	0	**JP.SIS (rr)** assembles to `40 E9` or `40 DD/FD E9`	This operation is the same as the previous operation. The **.SIS** extension does not affect operation when beginning in Z80 mode.
JP.L (rr)	0	**JP.LIS (rr)** assembles to `49 E9` or `49 DD/FD E9`	The starting program counter is {MBASE, PC[15:0]}. Write the 3-byte value stored in **rr**[23:0] to PC[23:0]. Set the ADL mode bit to 1. The ending program counter is PC[23:0]=**rr**[23:0].
JP (rr)	1	**JP (rr)** assembles to `E9` or `DD/FD E9`	The starting program counter is PC[23:0]. Write the 3-byte value stored in **rr**[23:0] to PC[23:0]. The ADL mode bit remains set to 1. The ending program counter is PC[23:0]=**rr**[23:0].
JP.L (rr)	1	**JP.LIL (rr)** assembles to `5B E9` or `5B DD/FD E9`	This operation is the same as the previous operation. The **.LIL** extension does not affect operation when beginning in ADL mode.
JP.S (rr)	1	**JP.SIL (rr)** assembles to `52E9` or `52DD/FD E9`	The starting program counter is PC[23:0]. Write the 2-byte value stored in **rr**[15:0] to PC[15:0]. Reset ADL mode bit to 0. The ending program counter is {MBASE, PC[15:0]} = {MBASE, **rr**[15:0]}.

Table 17. RST n Instruction

User Code	ADL Mode	Assembled Code	Operation
RST n	0	**RST n** assembles to `CD nn`	The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the {MBASE,SPS} stack. The ADL mode bit remains cleared to 0. Write {`00h`, nn} to PC[15:0]. The ending program counter is {MBASE, PC[15:0]} = {MBASE, `00h`, nn}.
RST n	1	**RST n** assembles to `CD nn`	The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. The ADL mode bit remains set to 1. Write {`0000h`, nn} to PC[23:0]. The ending program counter is PC[23:0] = {`0000h`, nn}.
RST.S n	0	**RST.SIS n** assembles to `40 CD nn`	The starting program counter is {MBASE, PC[15:0]} Push the 2-byte return address, PC[15:0], onto the {MBASE, SPS} stack. Push a `02h` byte onto the SPL stack, indicating an interrupt from Z80 mode (ADL = 0). The ADL mode bit remains cleared to 0. Write {`00h`, nn} to PC[15:0].The ending program counter is {MBASE, PC[15:0]} = {MBASE, `00h`, nn}.
RST.S n	1	**RST.SIL n** assembles to `52 CD nn`	The starting program counter is PC[23:0]. Push the 2 LS bytes of the return address, PC[15:0], onto the {MBASE, SPS} stack. Push the MS byte of the return address, PC[23:16], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode (because ADL = 1). Reset ADL mode bit to 0. Write {`00h`, nn} to PC[15:0]. The ending program counter is {MBASE, PC[15:0]} = {MBASE, `00h`, nn}.

Table 17. RST n Instruction (Continued)

User Code	ADL Mode	Assembled Code	Operation
RST.L n	0	**RST.LIS n** assembles to `49 CD nn`	The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating an interrupt from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Write {`0000h`, nn} to PC[23:0]. The ending program counter is PC[23:0] = {`0000h`, nn}.
RST.L n	1	**RST.LIL n** assembles to `5B CD nn`	The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Write {`0000h`, nn} to PC[23:0]. The ending program counter is PC[23:0] = {`0000h`, nn}.

Table 18. RET Instruction

User Code	ADL Mode	Assembled Code	Operation
RET	0	**RET** assembles to `C9`	The starting program counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending program counter is {MBASE, PC[15:0]}.
RET	1	**RET** assembles to `C9`	The starting program counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending program counter is PC[23:0].
RET.S	0	—	An invalid suffix. **RET.L** must be used in all mixed-memory mode applications.

Table 18. RET Instruction (Continued)

User Code	ADL Mode	Assembled Code	Operation
RET.S	1	—	An invalid suffix. **RET.L** must be used in all mixed-memory mode applications.

RET.L	0	**RET.LIS** assembles to `49 C9`	The starting program counter is {MBASE, PC[15:0]}.
			Pop a byte from SPL into ADL to set memory mode (`03h`=ADL, `02h`=Z80).
			if ADL mode {
			Pop the upper byte of the return address from SPL into PC[23:16].
			Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0].
			The ending program counter is PC[23:0].
			}
			else Z80 mode {
			Pop a 2-byte return address from {MBASE,SPS} into PC[15:0].
			The ending program counter is {MBASE, PC[15:0]}.
			}

RET.L	1	**RET.LIL** assembles to `5B C9`	The starting program counter is PC[23:0]. Pop a byte from SPL into ADL to set memory mode (`03h`=ADL, `02h`=Z80).
			if ADL mode {
			Pop 3-byte return address from SPL into PC[23:0].
			The ending program counter is PC[23:0].
			}
			else Z80 mode {
			Pop a 2-byte return address from SPL into PC[15:0].
			The ending program counter is {MBASE, PC[15:0]}.
			}

Table 19. RETI Instruction

User Code	ADL Mode	Assembled Code	Operation
RETI	0	**RETI** assembles to `ED 4D`	The starting program counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending program counter is {MBASE, PC[15:0]}.
RETI	1	**RETI** assembles to `ED 4D`	The starting program counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending program counter is PC[23:0].
RETI.S	0	—	Because RETI.S is an invalid suffix, RETI.L must be used in all mixed-memory mode applications.

RETI.L	0	**RETI.LIS** assembles to `49 ED 4D`	The starting program counter is {MBASE, PC[15:0]}. Pop a byte from SPL into ADL to set memory mode (`03h`=ADL, `02h`=Z80).
			if ADL mode {
			Pop the upper byte of the return address from SPL into PC[23:16].
			Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0].
			The ending program counter is PC[23:0].
			}
			else Z80 mode {
			Pop a 2-byte return address from {MBASE,SPS} into PC[15:0]. The ending program counter is {MBASE, PC[15:0]}.
			}

Table 19. RETI Instruction (Continued)

User Code	ADL Mode	Assembled Code	Operation

RETI.L	1	**RETI.LIL** assembles to `5B ED 4D`	The starting program counter is PC[23:0]. Pop a byte from SPL into ADL to set memory mode (`03h`=ADL, `02h`=Z80).
			if ADL mode {
			Pop a 3-byte return address from SPL into PC[23:0].
			The ending program counter is PC[23:0].
			}
			else Z80 mode {
			Pop a 2-byte return address from SPL into PC[15:0].
			The ending program counter is {MBASE, PC[15:0]}.
			}

Table 20. RETN Instruction

User Code	ADL Mode	Assembled Code	Operation
RETN	0	**RETN** assembles to `ED 45`	The starting program counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending program counter is {MBASE, PC[15:0]}. IEF1 ← IEF2.
RETN	1	**RETN** assembles to `ED 45`	The starting program counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending program counter is PC[23:0]. IEF1 ← IEF2.
RETN.S	0	—	Because RETI.S is an invalid suffix, RETN.L must be used in all mixed-memory mode applications. IEF1 ← IEF2.

Table 20. RETN Instruction (Continued)

User Code	ADL Mode	Assembled Code	Operation

RETN.L	0	**RETN.LIS** assembles to `49 ED 45`	The starting program counter is {MBASE, PC[15:0]}. Pop a byte from SPL into ADL to set memory mode (`03h` = ADL, `02h` = Z80).
			if ADL mode {
			Pop the upper byte of the return address from SPL into PC[23:16].
			Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0]. The ending program counter is PC[23:0].
			}
			else Z80 mode {
			Pop a 2-byte return address from {MBASE,SPS} into PC[15:0]. The ending program counter is {MBASE, PC[15:0]}. IEF1 ← IEF2.
			}

RETN.L	1	**RETN.LIL** assembles to `5B ED 45`	The starting program counter is PC[23:0]. Pop a byte from SPL into ADL to set memory mode (`03h` = ADL, `02h` = Z80).
			if ADL mode {
			Pop 3-byte return address from SPL into PC[23:0]. The ending program counter is PC[23:0].
			}
			else Z80 mode {
			Pop a 2-byte return address from SPL into PC[15:0]. The ending program counter is {MBASE, PC[15:0]}. IEF1 ← IEF2.
			}

Mixed-Memory Mode Applications

The eZ80® CPU contains a control bit flag that affects operation of interrupts, illegal instruction traps and restart (**RST**) instructions. The Mixed-ADL (MADL) control bit must be set to 1 for all applications that run in both Z80 mode and ADL mode. The MADL control bit can be reset to 0 for all CPU applications that run exclusively in Z80 mode or exclusively in ADL mode. Default for the MADL control bit is reset to 0.

No application program can run exclusively in ADL mode, because the default for the CPU is to begin in Z80 mode. If a single **JP.LIL** instruction is used at or near the beginning of the source code to permanently change to ADL mode, this program is considered to operate exclusively in ADL mode.

The purpose of the MADL control bit is to force the CPU to monitor the memory mode when interrupts, traps or **RST** instructions occur. If the memory mode does not change, then the MADL control bit can be reset to 0.

When the MADL control bit is set to 1, the CPU pushes a byte onto the stack that contains the current memory mode whenever an interrupt, trap, or restart occurs. Even if the memory mode is not changed by the current interrupt, trap, or restart, the byte containing the memory mode bit is still pushed onto the stack. A `02h` byte is pushed onto the stack if the current code is operating in Z80 mode. A `03h` byte is pushed onto the stack if the current code is operating in ADL mode. The current memory mode is pushed onto the stack prior to setting the memory mode for the called service routine.

In addition, when the MADL control bit is set to 1 for mixed- memory mode applications, all interrupts begin in ADL mode.

For applications that run exclusively in a single memory mode (either Z80 or ADL mode), set the MADL control bit to 1. The CPU always handles interrupts, traps and restarts correctly if MADL is set to 1.

The MADL control bit is set to 1 by the **STMIX** instruction. The MADL control bit is reset to 0 by the **RSMIX** instruction.

MIXED MEMORY Mode Guidelines

Applications that include legacy code that runs in Z80 mode, and new code that runs in ADL mode, must follow certain rules to ensure proper operation:

1. Include a **STMIX** instruction in the device initialization procedure that sets MADL to 1, ensuring that interrupt service routines begin in a consistent memory mode (ADL mode).
2. End all interrupt service routines with a **RETI.L** or **RETN.L** instruction to ensure that the interrupted code's memory mode is popped from the SPL stack.
3. Use a suffixed **CALL** to access each block of code in the memory mode in which it was assembled or compiled. Suffixed **JP** instructions may also be used; however, suffixed **CALL** instructions are recommended, because the CPU keeps track of all the necessary memory modes when switching between blocks.
4. Any code block that may be called from either Z80 mode or ADL mode must be called with a suffix to save the calling code's memory mode on the SPL stack.
5. Any routine that may be called from either mode must return with a suffixed **RETI.L** instruction to restore the calling code's memory mode from the SPL stack.
6. If a calling code operating in one mode must pass stack-based operands/ arguments to a routine compiled or assembled for a different mode, it must use suffixed instructions to set up the operands/arguments. For **PUSH**, **.S** and **.L** suffixes control whether SPS or SPL is used and whether the operands/ arguments are stored as 2- or 3-byte values.

Note: In mixed-ADL applications, some of these rules may represent exceptions to the eZ80® CPU's design goal; that legacy code does not require modification to run on the eZ80® CPU. Assuming that legacy routines are not selectively converted to ADL mode and do not call newly-written routines, the only rule that could lead to such modification is Rule 5. If each legacy Z80 mode routine ends with a single **RET.L** at its end, this conversion is easy. Internal and conditional **RET**s require more careful review.

Interrupts

Interrupts allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an interrupt service routine (ISR). Usually this interrupt service routine is involved with the exchange of data, status information, or control information between the CPU and the interrupting peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

The CPU respond to two different interrupt types—maskable interrupts and nonmaskable interrupts. The nonmaskable interrupt (NMI) cannot be disabled by the programmer. An NMI request is always accepted when the peripheral device requests it. You can enable or disable maskable interrupts.

Interrupt Enable Flags (IEF1 and IEF2)

In the eZ80® CPU, there are two interrupt enable flags (IEF1 and IEF2) that are set or reset using the Enable Interrupt (**EI**) and Disable Interrupt (**DI**) instructions. When IEF1 is reset to 0 by a **DI** instruction, a maskable interrupt cannot be accepted by the CPU. When IEF1 is set to 1 by an **EI** instruction, a maskable interrupt is acknowledged by the CPU and executed.

The state of IEF1 is used to enable or inhibit interrupts, while IEF2 is used as a temporary storage location for IEF1. At reset, the CPU clears both IEF1 and IEF2 to 0, which disables the maskable interrupts. The maskable interrupts can be enabled using the **EI** instruction. No pending interrupt is accepted until the instruction that follows the **EI** instruction is executed. The single instruction delay occurs because **EI** is often followed by a return instruction, and because interrupts must not be allowed until the return is complete.

When a maskable interrupt is accepted by the CPU, both IEF1 and IEF2 are reset to the disabled state, thus inhibiting further interrupts until a new

EI instruction is executed. For all of the cases discussed previously in this section, IEF1 and IEF2 are always equal.

The purpose of IEF2 is to save the status of IEF1 when a nonmaskable interrupt occurs. When a nonmaskable interrupt is accepted, IEF1 is reset to prevent further interrupts until reenabled by the application code. The status of IEF1 is restored by executing the Return From Nonmaskable (**RETN**) instruction. During execution of a Return From Nonmaskable Interrupt, the CPU copies the contents of IEF2 back into IEF1. In addition, the **LD A,I** or **LD A,R** instructions copy the state of IEF2 into the Parity flag where it can be tested or stored.

Interrupts in Mixed Memory Mode Applications

For all mixed-memory mode applications, the MADL control bit must be set to 1 using the **STMIX** instruction. When the MADL is set to 1, all interrupt service routines (ISRs) begin in ADL mode. To explain, the ADL mode bit is set to 1 and full 24-bit linear addressing is used to access the ISRs. The ADL mode bit setting of the interrupted code is pushed onto the stack, using SPL, to allow the memory mode to return to the appropriate value after completion of the ISR. For mixed-memory mode applications, all ISRs must end with either a **RETI.L** for maskable interrupts or **RETN.L** for nonmaskable interrupts.

eZ80® CPU Response to a Nonmaskable Interrupt

The CPU always accepts a nonmaskable interrupt (NMI). The state of the Interrupt Enable flags (IEF1 and IEF2) have no effect on nonmaskable interrupt operation. CPU operation in response to an NMI is described in detail in Table 21.

Table 21. Nonmaskable Interrupt Operation

Current Memory Mode	ADL Mode Bit	MADL Control Bit	Operation
Z80 mode	0	0	**IEF2 ← IEF1** **IEF1 ← 0** The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the {MBASE,SPS} stack. The ADL mode bit remains cleared to 0. Write `0066h` to PC[15:0]. The ending program counter is {MBASE, PC[15:0]} = {MBASE, `0066h`}. The interrupt service routine must end with **RETN**.
ADL mode	1	0	**IEF2 ← IEF1** **IEF1 ← 0** The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. The ADL mode bit remains set to 1. Write `000066h` to PC[23:0]. The ending program counter is PC[23:0]=`000066h`. The interrupt service routine must end with **RETN**.
Z80 mode	0	1	**IEF2 ← IEF1** **IEF1 ← 0** The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating interrupting from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Write `000066h` to PC[23:0]. The ending program counter is PC[23:0]=`000066h`. The interrupt service routine must end with **RETN.L**.

Table 21. Nonmaskable Interrupt Operation (Continued)

Current Memory Mode	ADL Mode Bit	MADL Control Bit	Operation
ADL mode	1	1	**IEF2 ← IEF1** **IEF1 ← 0** The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Write `000066h` to PC[23:0]. The ending program counter is PC[23:0]= `000066h`. The interrupt service routine must end with **RETN.L**.

eZ80® CPU Response to a Maskable Interrupt

The eZ80® CPU is capable of responding to a maskable interrupt using one of three interrupt modes: Interrupt Mode 0, Interrupt Mode 1, or Interrupt Mode 2. The maskable interrupt mode is set by the IM0, IM1, and IM2 instructions. Not all products within the eZ80® family support all 3 of these interrupt modes. Refer to the *eZ80® and eZ80Acclaim!® product specifications* for information on supported interrupt modes.

Interrupt Mode 0

In Interrupt Mode 0, the interrupting device places the appropriate instruction onto the data bus during the interrupt acknowledge cycle. Interrupt Mode 0 is the default state upon reset of the CPU. Interrupt Mode 0 is also selected by execution of the **IM 0** instruction.

The instruction placed on the data bus must be a single byte restart instruction, **RST n**, with binary value `C7h`, `CFh`, `D7h`, `DFh`, `E7h`, `EFh`, `F7h`, or `FFh`, or a **CALL Mmn** instruction with binary value `CDh`. If any other binary value is placed on the data bus during the interrupt acknowledge cycle, the CPU treats the instruction as a **NOP**. The binary opcodes corresponding to the memory mode suffixes (**.SIS**, **.LIL**, **.SIL**, or **.LIS**) cannot be placed on the data bus by the interrupting peripheral.

Table 22. Interrupt Mode 0 Operation

Current Memory Mode	ADL Mode Bit	MADL Control Bit	Operation (if RST n or CALL Mmn is placed on the data bus)
Z80 mode	0	0	Read the **RST n** of **CALL mn** instruction placed on the data bus, D[7:0], by the interrupting peripheral. **IEF1 ← 0** **IEF2 ← 0** The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the {MBASE,SPS} stack. The ADL mode bit remains cleared to 0. Write {`00h`, nn} or {mm, nn} to PC[15:0]. The ending program counter is {MBASE, PC[15:0]}={MBASE, `00h`, nn} or {MBASE, mm, nn}. The interrupt service routine must end with **RETI**.
ADL mode	1	0	Read **RST n** or **CALL Mmn** instruction placed on the data bus, D[7:0], by the interrupting peripheral. **IEF1 ← 0** **IEF2 ← 0** The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. The ADL mode bit remains set to 1. Write {`0000h`, nn} or {MM, mm, nn} to PC[23:0]. The ending program counter is PC[23:0]={`0000h`, nn} or {MM, mm, nn}. The interrupt service routine must end with **RETI**.

Table 22. Interrupt Mode 0 Operation (Continued)

Current Memory Mode	ADL Mode Bit	MADL Control Bit	Operation (if RST n or CALL Mmn is placed on the data bus)
Z80 mode	0	1	Read **RST n** or **CALL Mmn** instruction placed on the data bus, D[7:0], by interrupting peripheral. **IEF1 ← 0** **IEF2 ← 0** The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating interrupting from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Write {`0000h`, nn} or {MM, mm, nn} to PC[23:0]. The ending program counter is PC[23:0] = {`0000h`, nn} or {MM, mm, nn}. The interrupt service routine must end with **RETI.L**
ADL mode	1	1	Read **RST n** or **CALL Mmn** instruction placed on the data bus, D[7:0], by interrupting peripheral. **IEF1 ← 0** **IEF2 ← 0** The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Write {`0000h`, nn} or {MM, mm, nn} to PC[23:0]. The ending program counter is PC[23:0] = {`0000h`, nn} or {MM, mm, nn}. The interrupt service routine must end with **RETI.L**

Interrupt Mode 1

In Interrupt Mode 1, the CPU responds to an interrupt by executing a restart to location `0038h` (**RST** `38h`). Interrupt Mode 1 is selected by executing a **IM 1** instruction.

Table 23. Interrupt Mode 1 Operation

Current Memory Mode	ADL Mode Bit	MADL Control Bit	Operation
Z80 mode	0	0	**IEF1 ← 0** **IEF2 ← 0** The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the {MBASE,SPS} stack. The ADL mode bit remains cleared to 0. Write `0038h` to PC[15:0]. The ending program counter is {MBASE, PC[15:0]} = {MBASE, `0038h`} The interrupt service routine must end with **RETI**.
ADL mode	1	0	**IEF1 ← 0** **IEF2 ← 0** The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. The ADL mode bit remains set to 1. Write `000038h` to PC[23:0]. The ending program counter is PC[23:0]=`000038h`. The interrupt service routine must end with **RETI**.
Z80 mode	0	1	**IEF1 ← 0** **IEF2 ← 0** The starting program counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating interrupting from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Write `000038h` to PC[23:0]. The ending program counter is PC[23:0]=`000038h`. The interrupt service routine must end with **RETI.L**.

Table 23. Interrupt Mode 1 Operation (Continued)

Current Memory Mode	ADL Mode Bit	MADL Control Bit	Operation
ADL mode	1	1	**IEF1 ← 0** **IEF2 ← 0** The starting program counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Write `000038h` to PC[23:0]. The ending program counter is PC[23:0] = `000038h`. The interrupt service routine must end with **RETI.L**

Interrupt Mode 2

In Interrupt Mode 2, when an interrupt is accepted, the interrupting device places the lower eight bits of the interrupt vector on the data bus, D[7:0], during the interrupt acknowledge cycle. Bit 0 of this byte must be 0. The middle byte of the interrupt vector address is set by the CPU's Interrupt Vector Register, I.

In applications that run Z80 mode code exclusively, the interrupt vector address is {MBASE, I[7:0], D[7:0]}. A 16-bit word is fetched from the interrupt vector address and loaded into the lower two bytes of the Program Counter, PC[15:0].

In mixed-memory mode applications or ADL mode applications, the interrupt vector address is { I[15:0], D[7:0]}. A 24-bit word is fetched from the interrupt vector address and loaded into the Program Counter, PC[23:0].

Table 24. Interrupt Mode 2 Operation

Memory Mode	ADL Bit	MADL Bit	Operation
Z80 Mode	0	0	Read the LSB of the interrupt vector placed on the data bus, D[7:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is effectively {MBASE, PC[15:0]}. • Push the 2-byte return address PC[15:0] onto the ({MBASE,SPS}) stack. • The ADL mode bit remains cleared to 0. • The interrupt vector address is located at { MBASE, I[7:0], D[7:0] }. • PC[15:0] ← ({ MBASE, I[7:0], D[7:0] }). • The ending Program Counter is effectively {MBASE, PC[15:0]} • The interrupt service routine must end with RETI.
ADL Mode	1	0	Read the LSB of the interrupt vector placed on the data bus, D[7:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is PC[23:0]. • Push the 3-byte return address, PC[23:0], onto the SPL stack. • The ADL mode bit remains set to 1. • The interrupt vector address is located at { I[15:0], D[7:0] }. • PC[23:0] ← ({I[15:0], D[7:0] }). • The ending Program Counter is { PC[23:0] }. • The interrupt service routine must end with RETI.

Table 24. Interrupt Mode 2 Operation (Continued)

Memory Mode	ADL Bit	MADL Bit	Operation
Z80 Mode	0	1	Read the LSB of the interrupt vector placed on the data bus, D[7:0], bus by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is effectively {MBASE, PC[15:0]}. • Push the 2-byte return address, PC[15:0], onto the SPL stack. • Push a 00h byte onto the SPL stack to indicate an interrupt from Z80 mode (because ADL = 0). • Set the ADL mode bit to 1. • The interrupt vector address is located at { I[15:0], D[7:0] }. • PC[23:0] ← ({ I[15:0], D[7:0] }). • The ending Program Counter is { PC[23:0] }. • The interrupt service routine must end with RETI.L
ADL Mode	1	1	Read the LSB of the interrupt vector placed on the data bus, D[7:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is PC[23:0]. • Push the 3-byte return address, PC[23:0], onto the SPL stack. • Push a 01h byte onto the SPL stack to indicate a restart from ADL mode (because ADL = 1). • The ADL mode bit remains set to 1. • The interrupt vector address is located at {00h, I[7:0], D[7:0]}. • PC[23:0] ← ({ I[15:0], D[7:0] }). • The ending Program Counter is { PC[23:0] }. • The interrupt service routine must end with RETI.L

Vectored Interrupts for On-Chip Peripherals

Vectored interrupts operate in the same manner as Mode 2 interrupts, irrespective of which interrupt mode is selected. In the case of the vectored interrupts, the CPU does not fetch the low-order interrupt vector address from the data bus, D[7:0]. Instead, the CPU fetches from the internal vectored interrupt bus, at address IVECT[8:0]. The internal vectored interrupt bus is used exclusively for on-chip peripherals.

In applications that run Z80 mode code exclusively, the interrupt vector address is {MBASE, I[7:1], IVECT[8:0]}. A 16-bit word is fetched from the interrupt vector address and loaded into the lower two bytes of the Program Counter, PC[15:0].

In mixed-memory or ADL mode applications, the interrupt vector address is {I[15:1], IVECT[8:0]}. A 24-bit word is fetched from the interrupt vector address and loaded into the Program Counter, PC[23:0].

Note: The eZ80190, eZ80L92, and eZ80F92/F93 devices only support an 8-bit I register, an 8-bit IVECT, and a 16-bit word fetch in ADL modes. Refer to the *eZ80® and eZ80Acclaim!® product specifications* for information on product specific vectored interrupt modes.

Table 25. Vectored Interrupt Operation

Memory Mode	ADL Bit	MADL Bit	Operation
Z80 Mode	0	0	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [8:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is effectively {MBASE, PC[15:0]}. • Push the 2-byte return address PC[15:0] onto the ({MBASE,SPS}) stack. • The ADL mode bit remains cleared to 0. • The interrupt vector address is located at { MBASE, I[7:1], IVECT[8:0] }. • PC[15:0] ← ({ MBASE, I[7:1], IVECT[8:0] }). • The ending Program Counter is effectively {MBASE, PC[15:0]} • The interrupt service routine must end with RETI.
ADL Mode	1	0	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [8:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is PC[23:0]. • Push the 3-byte return address, PC[23:0], onto the SPL stack. • The ADL mode bit remains set to 1. • The interrupt vector address is located at { I[15:1], IVECT[8:0] }. • PC[23:0] ← ({ I[15:1], IVECT[8:0] }). • The ending Program Counter is { PC[23:0] }. • The interrupt service routine must end with RETI.

Table 25. Vectored Interrupt Operation (Continued)

Memory Mode	ADL Bit	MADL Bit	Operation
Z80 Mode	0	1	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT[8:0], bus by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is effectively {MBASE, PC[15:0]}. • Push the 2-byte return address, PC[15:0], onto the SPL stack. • Push a 00h byte onto the SPL stack to indicate an interrupt from Z80 mode (because ADL = 0). • Set the ADL mode bit to 1. • The interrupt vector address is located at { I[15:1], IVECT[8:0] }. • PC[23:0] ← ({ I[15:1], IVECT[8:0] }). • The ending Program Counter is { PC[23:0] }. • The interrupt service routine must end with RETI.L
ADL Mode	1	1	Read the LSB of the interrupt vector placed on the internal vectored interrupt bus, IVECT [8:0], by the interrupting peripheral. • IEF1 ← 0 • IEF2 ← 0 • The starting Program Counter is PC[23:0]. • Push the 3-byte return address, PC[23:0], onto the SPL stack. • Push a 01h byte onto the SPL stack to indicate a restart from ADL mode (because ADL = 1). • The ADL mode bit remains set to 1. • The interrupt vector address is located at {I[15:1], IVECT[8:0]}. • PC[23:0] ← ({ I[15:1], IVECT[8:0] }). • The ending Program Counter is { PC[23:0] }. • The interrupt service routine must end with RETI.L

Illegal Instruction Traps

The eZ80® CPU instruction set does not cover all possible sequences of binary values. Binary values and sequences for which no operation is defined are illegal instructions. When an eZ80® processor fetches one of these illegal instructions, it performs a TRAP operation.

While not a true eZ80® instruction, a TRAP operation functions similar to an **RST** `00h` instruction. The function of the TRAP instruction is illustrated in the following code segment:

```
if ADL mode (ADL=1) {
  (SPL) ← PC[23:0]
  if MIXED MEMORY mode (MADL=1) {
  (SPL) ← 03h
  }
PC[23:0] ← 000000h
}
else Z80 mode (ADL=0){
  SPS ← PC[15:0]
  if MIXED MEMORY mode (MADL=1) {
  (SPL) ← 02h
}
PC[15:0] ← 0000h
```

Effectively, PC[23:0]={MBASE, PC[15:0]}.

The current program counter is pushed onto the stack (the stack is either SPL or SPS depending upon the current memory mode). In addition, if the program code is written for MIXED MEMORY mode (MADL=1), the current memory mode information is also pushed onto the stack.

The memory mode suffixes (**.SIS**, **.SIL**, **.LIS**, and **.LIL**) do not guarantee illegal instruction traps, even when used with instructions for which they have no meaning. For example, preceding a Complement Carry Flag instruction (CCF) with an .SIS suffix of opcode `40h` is allowed. The memory mode suffixes configure the CPU to act in a particular memory mode and fetch a particular number of bytes from the opcode stream, if necessary. Because the CCF instruction is not affected by the current memory mode and does not fetch any operands, there is no effect. The memory mode opcodes do not generate traps because they do not push into secondary pages of the opcode tables, which may contain undefined binary values.

Some products that employ the CPU can also contain a TRAP register for capturing the illegal binary value. Refer to the *eZ80® and eZ80Acclaim!® product specifications* for more information.

I/O Space

A separate I/O space may include both on- and off-chip peripheral devices. The eZ80® CPU is capable of addressing a 64 KB I/O space with 16-bit addresses. The memory and I/O space share the same 24-bit address and 8-bit data buses. However, the I/O peripherals are accessed using special I/O instructions including **IN**, **OUT**, and **TSTIO**. Whenever an I/O instruction is executed the upper byte of the 24-bit address bus is undefined.

Refer to the *eZ80® and eZ80Acclaim!® product specifications* for more information on using the I/O address space and the on-chip I/O peripherals.

Addressing Modes

The eZ80® CPU instruction set includes many different memory addressing modes. The memory address can be formed using several different methods, as outlined in the following text. The addressing modes supported are a function of each instruction.

Implied Register Addressing

Certain opcodes automatically imply a particular register to be used during execution. Implied register instructions include many arithmetic operations that inherently reference the accumulator (A), the Index registers (IX and IY), the Stack Pointer (SPS or SPL), or the general purpose working registers. Instructions using implied register addressing include **INC** A, **EXX,** and **CCF**.

Restart Addressing

The eZ80® CPU features eight special single-byte restart (**RST**) instructions that set the Program Counter (PC) to any of eight locations within the first 256 bytes of memory. In Z80 mode, the 16-bit program counter (PC) is set to one of the following values—`0000h`, `0008h`, `0010h`, `0018h`, `0020h`, `0028h`, `0030h`, or `0038h`. In Z80 mode, the MBASE register is unaffected by a **RST** instruction. Therefore the restart jumps to a location on the current Z80 page. In ADL mode, the 24-bit Program Counter (PC) is set to any of the following 24-bit addresses:

- `000000h`
- `000008h`
- `000010h`
- `000018h`
- `000020h`
- `000028h`
- `000030h`
- `000038h`

Register Indirect Addressing

The memory operand address is taken from one of the multibyte BC, DE or HL registers. Register indirect addressing is illustrated in Figure 6.

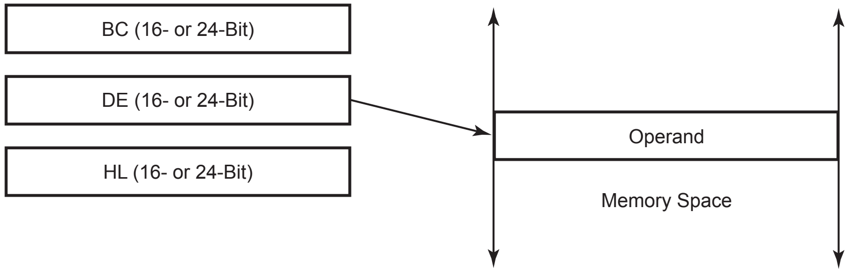


Figure 6. Register Indirect Addressing

Immediate Addressing

The memory operands immediately follows the instruction. The memory operand can be 8, 16, or 24 bits, depending on the instruction and the memory mode in use. Immediate addressing is illustrated in Figure 7.

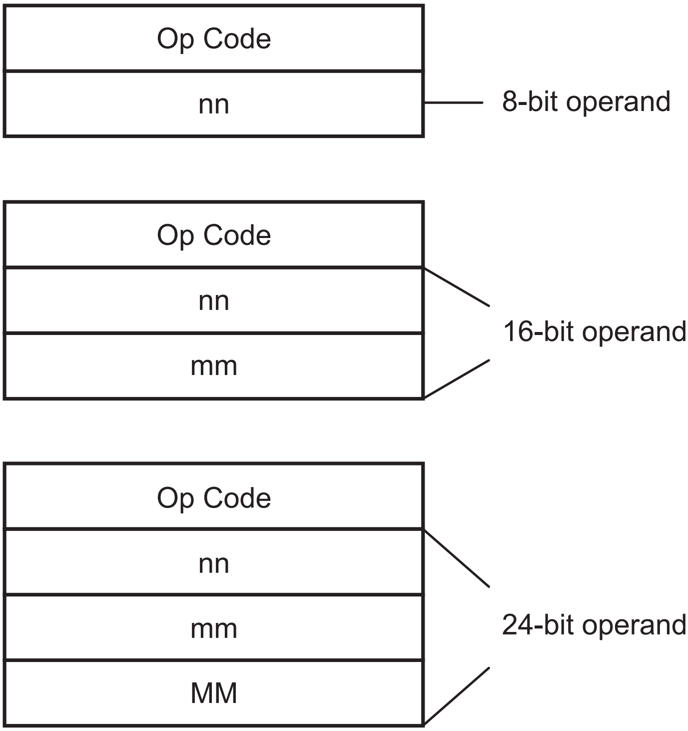


Figure 7. Immediate Addressing

Indexed Addressing

In this mode of addressing, a byte of data following the opcode contains a displacement to be added to one of the IX or IY Index registers. The displacement is a two's-complement value in the range +127 to –128. Figure 8 illustrates the indexed addressing.

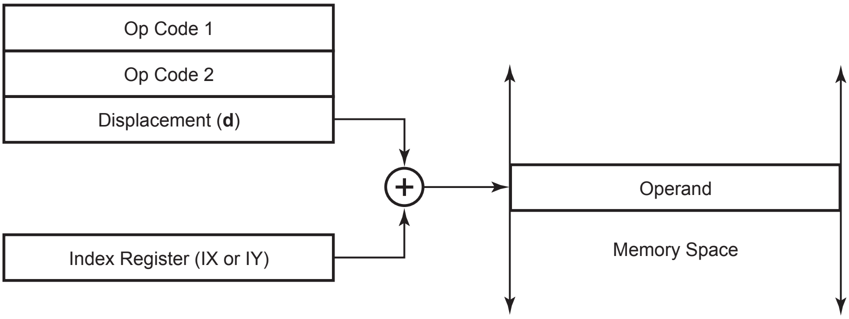


Figure 8. Indexed Addressing

Extended Addressing

The memory operand address is specified by two or three bytes following the opcode. Figure 9 illustrates the extended addressing.

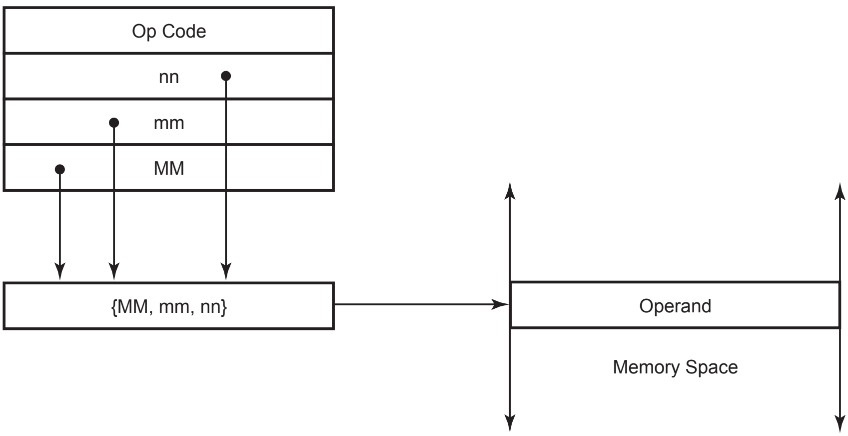


Figure 9. Extended Addressing

Relative Addressing

Relative addressing uses one byte of data following the opcode to specify a displacement from the existing program to which a program jump can occur. The displacement is a two's-complement number that is added to the address of the opcode following the instruction. The displacement can range from +127 to –128. Figure 10 illustrates the relative addressing.

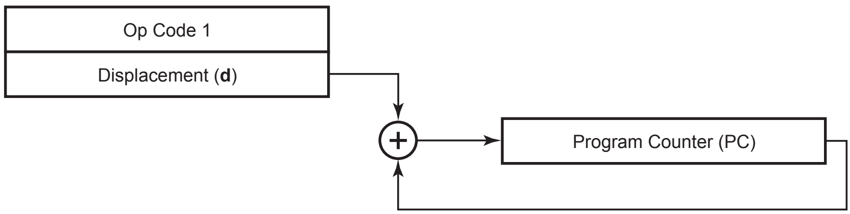


Figure 10. Relative Addressing

I/O Addressing

I/O addressing mode is used only by the I/O instructions **IN** and **OUT**. See the I/O Space section on page 63 for more information on I/O operations.

CPU Instruction Set

eZ80® CPU Assembly Language Programming Introduction

eZ80® CPU assembly language provides a means for writing an application program without considering the actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement contains labels, operations, operands, and comments.

Labels are assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the CPU.

An example segment of an assembly language source program is detailed in the following example.

Assembly Language Source Program Example

```
JP.LIL START         ;Everything after the semicolon is
                     ;a comment.
.ASSUME ADL=1        ;A compiler directive or pseudo-
                     ;op.
START:               ;A label called "START". The first
                     ;instruction in this example
                     ;causes program execution to jump
                     ;to the point within the program
                     ;where the JP.LIL (Jump) executes.
LD A, 3Ah            ;A Load (LD) instruction with two
                     ;operands. The accumulator
                     ;register, A, is the first operand
                     ;that indicates the destination
                     ;for this instruction. The
                     ;hexadecimal constant value 3Ah is
                     ;the second operand signifying the
                     ;source value for this
                     ;instruction.
```

eZ80® CPU assembly language is designed to minimize the number of different opcodes corresponding to the set of basic machine operations, in addition to providing a consistent description of instruction operands. The nomenclature is defined with special emphasis on mnemonic values and readability.

The movement of data is indicated by a single opcode, regardless whether the movement is between different registers or between registers and memory locations. For example, the first operand of an **LD** instruction is the destination of the operation and the second operand is the source of the operation. Thus, **LD A, B** indicates that the contents of the second operand, working register B, are to be transferred to the first operand, which is the accumulator, A. In the opcode descriptions, this operation is often represented as:

A ← B

Similarly, **LD C**, `3Fh` indicates that the constant `3Fh` is written to working register C:

C ← `3Fh`

Enclosing an operand in parentheses indicates a memory location addressed by the contents of the parentheses (i.e. an indirect memory access). For example, **LD BC**, **(HL)** indicates that the contents of the multibyte HL register are used as an address to a memory location. Multi-byte register BC is loaded with the data stored at the memory location pointed to by the contents of HL:

BC ← (HL)

Similarly, **LD (IX+6), C** indicates that the contents of register C are to be stored in the memory location addressed by the current value in the multibyte IX register plus 6:

(IX+6) ← C

eZ80® CPU Instruction Notations

The notations in the CPU instructions are defined in Table 26.

Table 26. Instruction Notations

Mnemonic	Definition
cc	Condition code C, NC, Z, NZ, P, M, PO, or PE—tests of single bits in Flags register
cc'	Condition code C, NC, Z, or NZ—tests of single bits in Flags register
d	An 8-bit two's complement displacement with value from –128 to 127
ir or **ir'**	8-bit CPU register IXH (IX[15:8]), IXL (IX[7:0]), IYH (IY[15:8]), or IYL (IY[7:0])
IX/Y	CPU Index Register IX or IY
(IX/Y+d)	A location in memory with address formed by the sum of the contents of the Index Register, IX or IY, and the two's-complement displacement **d**
Mmn	A 24-bit immediate data value
(Mmn)	A 24-bit value indicating a location in memory at this address
mn	A 16-bit immediate data value
(mn)	A 16-bit value indicating a location in memory at this address
n	8-bit immediate data value
r or **r'**	8-bit CPU register A, B, C, D, E, H, or L
rr	16- or 24-bit CPU register BC, DE, or HL
rxy	16- or 24-bit CPU register BC, DE, IX or IY
s	8-bit value
SP	Stack Pointer. Indicates either the Stack Pointer Short register (SPS) or the Stack Pointer Long register (SPL)
ss	8-, 16-, or 24-bit value, depending on instruction and context

eZ80® CPU Instruction Classes

Table 27. Arithmetic Instructions

Mnemonic	Instruction	Page(s)
ADC	Add with Carry	108–116
ADD	Add without Carry	117–128
CP	Compare with Accumulator	148–152
DAA	Decimal Adjust Accumulator	161
DEC	Decrement	164–171
INC	Increment	188–195
MLT	Multiply	295–296
NEG	Negate Accumulator	297
SBC	Subtract with Carry	396–404
SUB	Subtract without Carry	428–432

Table 28. Bit Manipulation Instructions

Mnemonic	Instruction	Page(s)
BIT	Bit Test	135–139
RES	Reset Bit	345–349
SET	Set Bit	406–409

Table 29. Block Transfer and Compare Instructions

Mnemonic	Instruction	Page(s)
CPD (CPDR)	Compare and Decrement (with Repeat)	154–155
CPI (CPIR)	Compare and Increment (with Repeat)	157–158
LDD (LDDR)	Load and Decrement (with Repeat)	285–286
LDI (LDIR)	Load and Increment (with Repeat)	288–289

Table 30. Exchange Instructions

Mnemonic	Instruction	Page(s)
EX	Exchange Registers	175–178
EXX	Exchange CPU Multibyte Register Banks	179

Table 31. Input/Output Instructions

Mnemonic	Instruction	Page(s)
IN	Input from I/O	183–184
IN0	Input from I/O on Page 0	186
IND (INDR)	Input from I/O and Decrement (with Repeat)	196
INDRX	Input from I/O and Decrement Memory Address with Stationary I/O Address	207
IND2 (IND2R)	Input from I/O and Decrement (with Repeat)	197–199
INDM (INDMR)	Input from I/O and Decrement (with Repeat)	201–203
INI (INIR)	Input from I/O and Increment (with Repeat)	209, 218
INIRX	Input from I/O and Increment Memory Address with Stationary I/O Address	220
INI2 (INI2R)	Input from I/O and Increment (with Repeat)	210–212
INIM (INIMR)	Input from I/O and Increment (with Repeat)	214–216
OTDM (OTDMR)	Output to I/O and Decrement (with Repeat)	307–308
OTDRX	Output to I/O and Decrement Memory Address with Stationary I/O Address	310

Table 31. Input/Output Instructions (Continued)

Mnemonic	Instruction	Page(s)
OTIM (OTIMR)	Output to I/O and Increment (with Repeat)	314–315
OTIRX	Output to I/O and Increment Memory Address with Stationary I/O Address	318
OUT	Output to I/O	320–322
OUT0	Output to I/0 on Page 0	323
OUTD (OTDR)	Output to I/O and Decrement (with Repeat)	325, 309
OUTD2 (OTD2R)	Output to I/O and Decrement (with Repeat)	326, 305
OUTI (OTIR)	Output to I/O and Increment (with Repeat)	327, 317
OUTI2 (OTI2R)	Output to I/O and Increment (with Repeat)	328, 312
TSTIO	Test I/O	438

Table 32. Load Instructions

Mnemonic	Instruction	Page(s)
LD	Load	232–284
LEA	Load Effective Address	291–294
PEA	Push Effective Address	329–331
POP	Pop	333–337
PUSH	Push	339–343

Table 33. Logical Instructions

Mnemonic	Instruction	Page(s)
AND	Logical AND	129–133
CPL	Complement Accumulator	160
OR	Logical OR	299–303
TST	Test Accumulator	434–436
XOR	Logical Exclusive OR	439–444

Table 34. Processor Control Instructions

Mnemonic	Instruction	Page
CCF	Complement Carry Flag	147
DI	Disable Interrupts	172
EI	Enable Interrupts	174
HALT	Halt	180
IM	Interrupt Mode	181
NOP	No Operation	298
RSMIX	Reset MIXED MEMORY Mode Flag	392
SCF	Set Carry Flag	405
SLP	Sleep	416
STMIX	Set MIXED MEMORY Mode Flag	427

Table 35. Program Control Instructions

Mnemonic	Instruction	Page(s)
CALL	Call Subroutine	144
CALL cc	Conditional Call Subroutine	141
DJNZ	Decrement and Jump if Nonzero	173
JP	Jump	224–228
JP cc	Conditional Jump	222
JR	Jump Relative	231
JR cc	Conditional Jump Relative	230
RET	Return	351
RET cc	Conditional Return	354
RETI	Return from Interrupt	357
RETN	Return from nonmaskable interrupt	360
RST	Restart	393

Table 36. Rotate and Shift Instructions

Mnemonic	Instruction	Page(s)
RL	Rotate Left	363–366
RLA	Rotate Left Accumulator	368
RLC	Rotate Left Circular	369–372
RLCA	Rotate Left Circular Accumulator	374
RLD	Rotate Left Decimal	375
RR	Rotate Right	377–381
RRA	Rotate Right Accumulator	383
RRC	Rotate Right Circular	384–387
RRCA	Rotate Right Circular Accumulator	389

Table 36. Rotate and Shift Instructions (Continued)

Mnemonic	Instruction	Page(s)
RRD	Rotate Right Decimal	390
SLA	Shift Left	411–414
SRA	Shift Right Arithmetic	417–420
SRL	Shift Right Logical	422–425

Instruction Summary

Table 37 describes each type or class of instruction, using the notation described in the preceding sections. In addressing modes where the same location acts as both the destination (Dest) and the source (Source), the information is centered between the Dest and Source columns (for example, the **DEC** instruction). The instruction summary table is sorted alphabetically by the assembly language mnemonics.

Table 37. Instruction Summary

	Address Mode		Opcode(s)	Flags Affected					
Instruction and Operation	**Dest**	**Source**	**(Hex)**	**S**	**Z**	**H**	**P/V**	**N**	**C**
ADC A,**s** A ← A+**s**+C		(HL)	`8E`	*	*	*	V	0	*
		ir	`DD/FD 8C–8D`						
		(IX/Y+d)	`DD/FD 8E dd`						
		n	`CE`						
		r	`88–8F`						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
ADC HL,**ss** HL ← HL+**ss**+C		**rr**	`ED 4A-6A`	*	*	*	V	0	*
		SP	`ED 7A`						
ADD A,**s** A ← A+**s**		(HL)	`86`	*	*	*	V	0	*
		ir	`DD/FD 84-85`						
		(**IX/Y**+**d**)	`DD/FD 86 dd`						
		n	`C6`						
		r	`80-87`						
ADD HL,**ss** HL ← HL+**ss**		**rr**	`09-29`	—	—	*	—	0	*
		SP	`39`						
ADD IX/Y,**ss** **IX/Y** ← IX/y+**ss**		**rxy**	`DD/FD 09-29`	—	—	*	—	0	*
		SP	`DD 39`						
AND A,**s** A ← A **AND s**		(HL)	`A6`	*	*	1	P	0	0
		ir	`DD/FD A4-A5`						
		(**IX/Y**+**d**)	`DD/FD A6 dd`						
		n	`E6`						
		r	`A0-A7`						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
BIT b,s Z ← ~**s**[**b**]		(HL)	`CB 46-7E`	X	*	1	X	0	—
		(IX/Y+d)	`DD/FD CB dd 46-7E`						
		r	`CB 40-7F`						

CALL cc,Mmn
if **cc** {
(SP) ← PC
PC ← **Mmn**
}
			`C4-FC`	—	—	—	—	—	—

CALL Mmn **(SP)** ← PC PC ← **Mmn**			`CD`	—	—	—	—	—	—
CCF C ← ~C			`3F`	—	—	*	—	0	*
CP A,**s** A−**s**		(HL)	`BE`	*	*	*	V	1	*
		ir	`DD/FD BC-BD`						
		(IX/Y+d)	`DD/FD BE dd`						
		n	`FE`						
		r	`B8-BF`						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
CPD A–(HL) HL ← HL−1 BC ← BC−1			`ED A9`	*	*	*	*	1	—
CPDR repeat { A–(HL) HL ← HL−1 BC ← BC−1 } while (~Z and BC ≠ 0)			`ED B9`	*	*	*	*	1	—
CPI A–(HL) HL ← HL+1 BC ← BC−1			`ED A1`	*	*	*	*	1	—
CPIR repeat { A–(HL) HL ← HL+1 BC ← BC−1 } while (~Z and BC ≠ 0)			`ED B1`	*	*	*	*	1	—
CPL A ← ~A			`2F`	—	—	1	—	1	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
DAA A ← decimal adjust (A)			`27`	*	*	*	P	—	*
DEC ss **ss** ← **ss**−1		(HL)	`35`	*	*	*	V	1	—
		ir	`DD/FD 25-2D`	*	*	*	V	1	—
		IX/Y	`DD/FD 2B`	—	—	—	—	—	—
		(**IX/Y**+**d**)	`DD/FD 35 dd`	*	*	*	V	1	—
		r	`05-3D`	*	*	*	V	1	—
		rr	`0B-2B`	—	—	—	—	—	—
		SP	`3B`	—	—	—	—	—	—
DI IEF1,2 ← 0			`F3`	—	—	—	—	—	—

DJNZ d
B ← B−1
if B ≠ 0 {
PC ← PC+**d**
}
			`10 dd`	—	—	—	—	—	—

EI IEF1,2 ← 1			`FB`	—	—	—	—	—	—
EX AF,AF' AF ↔ AF'			`08`	*	*	*	*	*	*

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
EX DE,HL DE ↔ HL			`EB`	—	—	—	—	—	—
EX (SP),ss **(SP) ↔ ss**		HL	`E3`	—	—	—	—	—	—
		IX/Y	`DD/FD E3`						
EXX BC ↔ BC' DE ↔ DE' HL ↔ HL'			`D9`	—	—	—	—	—	—
HALT			`76`	—	—	—	—	—	—
IM n			`ED 46-5E`	—	—	—	—	—	—
IN A,(**n**) A ←({`00h`, A, **n**)})			`DB`	—	—	—	—	—	—
IN r,(BC) also **IN r**,(C) **r** ← ({`00h`, BC[15:0]})			`ED 40-78`	*	*	0	P	0	–
IN0 r,(**n**) **r** ← ({`0000h`, **n**})			`ED 00-38`	*	*	0	P	0	–

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
INC ss ss ← ss+1		(HL)	`34`	*	*	*	V	1	–
		ir	`DD/FD 24-2C`	*	*	*	V	1	–
		IX/Y	`DD/FD 23`	—	—	—	—	—	—
		(IX/Y+d)	`DD/FD 34 dd`	*	*	*	V	1	–
		r	`04-3C`	*	*	*	V	1	–
		rr	`03-23`	—	—	—	—	—	—
		SP	`33`	—	—	—	—	—	—
IND (HL) ← ({`00h`, BC[15:0]}) B ← B–1 HL ← HL–1			`ED AA`	—	*	—	—	*	—
IND2 (HL) ← ({`00h`, BC[15:0]}) B ← B–1 C ← C–1 HL ← HL–1			`ED 8C`	—	*	—	—	*	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
IND2R repeat { (HL) ← ({`00h`, DE[15:0]}) BC ← BC–1 DE ← DE–1 HL ← HL–1 } while BC ≠ 0			`ED 9C`	—	1	—	—	*	—
INDM (HL) ← ({`0000h`, C}) B ← B–1 C ← C–1 HL ← HL–1			`ED 8A`	X	*	X	X	*	X
INDMR repeat { (HL) ← ({`0000h`, C}) B ← B–1 C ← C–1 HL ← HL–1 } while B ≠ 0			`ED 9A`	—	1	—	—	*	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
INDR repeat { (HL) ← ({00h, BC[15:0]}) B ← B–1 HL ← HL–1 } while B ≠ 0			ED BA	—	1	—	—	*	—
INDRX repeat { (HL) ← ({00h, DE[15:0]}) BC ← BC–1 HL ← HL–1 } while BC ≠ 0			ED CA	—	1	—	—	*	—
INI (HL) ← ({00h, BC[15:0]}) B ← B–1 HL ← HL+1			ED A2	—	*	—	—	*	—
INI2 (HL) ← ({00h, BC[15:0]}) B ← B–1 C ← C+1 HL ← HL+1			ED 84	—	*	—	—	*	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
INI2R repeat { (HL) ← ({`00h`, DE[15:0]}) BC ← BC−1 DE ← DE+1 HL ← HL+1 } while BC ≠ 0			`ED 94`	—	1	—	—	*	—
INIM (HL) ← ({`0000h`, C}) B ← B−1 C ← C+1 HL ← HL+1			`ED 82`	X	*	X	X	*	X
INIMR repeat { (HL) ← ({`0000h`, C}) B ← B−1 C ← C+1 HL ← HL+1 } while B ≠ 0			`ED 92`	—	1	—	—	*	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
INIR repeat { (HL) ← ({`00h`, BC[15:0]}) B ← B–1 HL ← HL+1 } while B ≠ 0			`ED B2`	—	1	—	—	*	—
INIRX repeat { (HL) ← ({`00h`, DE[15:0]}) BC ← BC–1 HL ← HL+1 } while BC ≠ 0			`ED C2`	—	1	—	—	*	—

JP cc,Mmn
if **cc** {
PC ← **Mmn** {
if **.S** {ADL ← 0}
else if **.L** {ADL ← 1}
}
			`C2-FA`	—	—	—	—	—	—

JP (ss) PC ← **ss** if **.S** {ADL ← 0} else if **.L** {ADL ← 1}		(HL)	`E9`	—	—	—	—	—	—
		(IX/Y)	`DD/FD E9`						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
JP Mmn PC ← **Mmn** { if **.S** {ADL ← 0} else if **.L** {ADL ← 1}			C3	—	—	—	—	—	—
JR cc',d if **cc'** {PC ← PC+**d**}			20–38	—	—	—	—	—	—
JR d PC ← PC+**d**			18	—	—	—	—	—	—
LD A,**s** A ← **s**		I[7:0]	ED 57	*	*	0	IEF2	0	—
		(IX/Y+d)	DD/FD 7E	—	—	—	—	—	—
		MB	ED 6E	—	—	—	—	—	—
		(Mmn)	3A	—	—	—	—	—	—
		R	ED 5F	*	*	0	IEF2	0	—
		(rr)	0A, 1A, 7E	—	—	—	—	—	—
LD HL,I HL ← I			ED D7	—	—	—	—	—	—
LD (HL),**ss** (HL) ← **ss**		**IX/Y**	ED 3E-3F	—	—	—	—	—	—
		n	36						
		r	70-77						
		rr	ED 0F-2F						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
LD I,A I[7:0] ← A			ED 47	—	—	—	—	—	—
LD I,HL I ← HL			ED C7	—	—	—	—	—	—
LD ir, **s** **ir** ← **s**		**ir'**	DD/FD 64-6D	—	—	—	—	—	—
		n	DD/FD 26-2E						
		r	DD/FD 60-67						
LD IX/Y, **ss** **IX/Y** ← **ss**		(HL)	ED 31-7	—	—	—	—	—	—
		(IX/Y+d)	DD/FD 31-37						
		Mmn	DD/FD 21						
		(Mmn)	DD/FD 2A						
LD (**IX/Y**+**d**), **ss** (**IX/Y**+**d**) ← **ss**		**IX/Y**	DD/FD 3E-3F	—	—	—	—	—	—
		n	DD/FD 36						
		r	DD/FD 70-77						
		rr	DD/FD 0F-2F						
LD MB,A if ADL mode {MBASE ← A}			ED 6D	—	—	—	—	—	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
LD (Mmn), ss **(Mmn) ← ss**		A	32	—	—	—	—	—	—
		IX/Y	DD/FD 22						
		rr	ED 43–63						
		SP	ED 73						
LD R, A R ← A		A	ED 4F	—	—	—	—	—	—
LD r, s **r ← s**		(HL)	46-7E	—	—	—	—	—	—
		ir	DDFD 44-7D						
		(IX/Y+d)	DD/FD 46-7E						
		n	06-3E						
		r'	41-7F						
LD rr, ss **rr ← ss**		(HL)	ED 07-27	—	—	—	—	—	—
		(IX/Y+d)	DD/FD 07-27						
		Mmn	01-21						
		(Mmn)	ED 4B-6B						
LD (rr), A **(rr)** ← A		A	02, 12, 77	—	—	—	—	—	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
LD SP, ss **SP ← ss**		HL	`F9`	—	—	—	—	—	—
		IX/Y	`DD/FD F9`						
		Mmn	`31`						
		(Mmn)	`ED 7B`						
LDD (DE) ← (HL) DE ← DE–1 HL ← HL–1 BC ← BC–1			`ED A8`	—	—	0	*	0	—
LDDR repeat { (DE) ← (HL) DE ← DE–1 HL ← HL–1 BC ← BC–1 } while BC ≠ 0			`ED B8`	—	—	0	*	0	—
LDI (DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC–1			`ED A0`	—	—	0	*	0	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
LDIR repeat { (DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC–1 } while BC ≠ 0			`ED B0`	—	—	0	*	0	—
LEA IX/Y, IX+**d** **IX/Y** ← IX+**d**		IX+**d**	`ED 32-55`	—	—	—	—	—	—
LEA IX/Y, IY+**d** **IX/Y** ← IY+**d**		IY+**d**	`ED 33-54`	—	—	—	—	—	—
LEA rr, IX+**d** **rr** ← IX+**d**		IX+**d**	`ED 02-22`	—	—	—	—	—	—
LEA rr, IY+**d** **rr** ← IY+**d**		IY+**d**	`ED 03-23`	—	—	—	—	—	—
MLT ss **ss**[15:0] ← **ss**[15:8] X **ss**[7:0]	**rr**		`ED 4C-6C`	—	—	—	—	—	—
	SP		`ED 7C`						
NEG A ← 0–A			`ED 44`	*	*	*	V	1	*
NOP			`00`	—	—	—	—	—	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
OR A,s A ← A **OR s**		(HL)	`B6`	*	*	0	P	0	0
		ir	`DD/FD B4–B5`						
		(IX/Y+d)	`DD/FD B6 dd`						
		n	`F6`						
		r	`B0–B7`						
OTD2R repeat { ({`00h`, DE[15:0]} ← (HL)) BC ← BC–1 DE ← DE–1 HL ← HL–1 } while BC ≠ 0			`ED BC`	—	1	—	—	*	—
OTDM ({`0000h`, C}) ← (HL) B← B–1 C ← C–1 HL ← HL–1			`ED 8B`	X	*	X	X	*	X

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
OTDMR repeat { ({0000h, C}) ← (HL) B ← B–1 C ← C–1 HL ← HL–1 } while B ≠ 0			ED 9B	X	1	X	X	*	X
OTDR repeat { ({00h, BC[15:0]}) ← (HL) B ← B–1 HL ← HL–1 } while B ≠ 0			ED BB	—	1	—	—	*	—
OTDRX repeat { ({00h, DE[15:0]}) ← (HL) BC ← BC–1 HL ← HL–1 } while BC ≠ 0			ED CB	—	1	—	—	*	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
OTI2R repeat { ({00h, DE[15:0]}) ← (HL) BC ← BC−1 DE ← DE+1 HL ← HL+1 } while BC ≠ 0			`ED B4`	—	1	—	—	*	—
OTIM ({0000h, C}) ← (HL) B← B−1 C ← C+1 HL ← HL+1			`ED 83`	X	*	X	X	*	X
OTIMR repeat { ({0000h, C}) ← (HL) B ← B−1 C ← C+1 HL ← HL+1 } while B ≠ 0			`ED 93`	X	1	X	X	*	X

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
OTIR repeat { ({00h, BC[15:0]}) ← (HL) B ← B–1 HL ← HL+1 } while B ≠ 0			ED B3	—	1	—	—	*	—
OTIRX repeat { ({00h, DE[15:0]}) ← (HL) BC ← BC–1 HL ← HL+1 } while BC ≠ 0			ED C3	—	1	—	—	*	—
OUT (BC),**r** also **OUT** (C),**r** ({00h, BC[15:0]}) ← **r**			ED 41–79	—	—	—	—	—	—
OUT (**n**),A ({00h, A, **n**}) ← A			D3	—	—	—	—	—	—
OUT0 (**n**),**r** ({0000h, **n**}) ← **r**			ED 01–39	—	—	—	—	—	—
OUTD ({00h, BC[15:0]}) ← (HL) B ← B–1 HL ← HL–1			ED AB	—	*	—	—	*	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
OUTD2 ({00h, BC[15:0]}) ← (HL) B ← B–1 C ← C–1 HL ← HL–1			ED AC	—	*	—	—	*	—
OUTI ({00h, BC[15:0]}) ← (HL) B ← B–1 HL ← HL+1			ED A3	—	*	—	—	*	—
OUTI2 ({00h, BC[15:0]}) ← (HL) B ← B–1 C ← C+1 HL ← HL+1			ED A4	—	*	—	—	*	—

PEA IX+**d**
if ADL mode {
(SPL) ← IX+**d**
SPL ← SPL–3
}
else Z80 mode {
SPS ← IX+**d**
SPS ← SPS–2
}
			ED 65	—	—	—	—	—	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode		Opcode(s)	Flags Affected					
	Dest	**Source**	**(Hex)**	**S**	**Z**	**H**	**P/V**	**N**	**C**

PEA IY+**d**
if ADL mode {
(SPL) ← IY+**d**
SPL ← SPL–3
}
else Z80 mode {
SPS ← IY+**d**
SPS ← SPS–2
}
			`ED 66`	—	—	—	—	—	—

POP ss
if ADL mode{
ss ← (SPL)
SPL ← SPL+3
}
else Z80 mode {
ss ← {MBASE, SPS}
SPS ← SPS+2
}
	AF		`F1`	F ← (SPL) or (SPS)					

	IX/Y		`DD/FD E1`	—	—	—	—	—	—
	rr		`C1-E1`	—	—	—	—	—	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

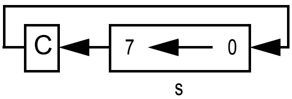
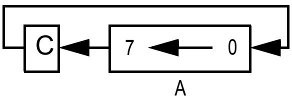
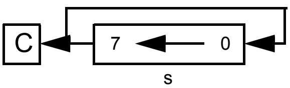
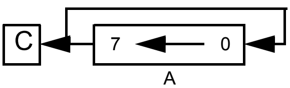
Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C

PUSH ss
if ADL mode {
(SPL) ← **ss**
SPL ← SPL–3
}
else Z80 mode{
SPS ← **ss**
SPS ← SPS–2
}
		AF	`F5`	—	—	—	—	—	—

		IX/Y	`DD/FD E5`						
		rr	`C5-E5`						
RES b,s s[**b**] ← 0	(HL)		`CB 86-BE`	—	—	—	—	—	—
	(**IX/Y**+**d**)		`DD/FD CB dd 86-BE`						
	r		`CB 80-BF`						
RET PC ← **(SP)**			`C9`	—	—	—	—	—	—
RET cc if **cc** {PC ← **(SP)**}			`C0-F8`	—	—	—	—	—	—
RETI PC ← **(SP)**			`ED 4D`	—	—	—	—	—	—
RETN Same as **RET**, with addition of IEF1 ← IEF2			`ED 45`	—	—	—	—	—	—

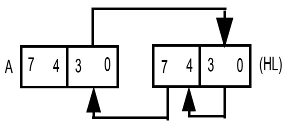
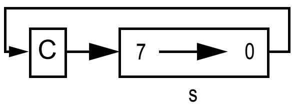
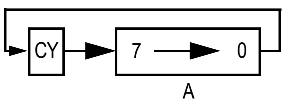
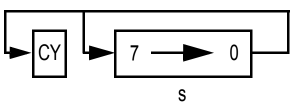
Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
RL s 	(HL)		CB 16	*	*	0	P	0	*
	(IX/Y+d)		DD/FD CB dd 16						
	r		CB 10–17						
RLA 	A		17	—	—	0	—	0	*
RLC s 	(HL)		CB 06	*	*	0	P	0	*
	(IX/Y+d)		DD/FD CB dd 06						
	r		CB 00–07						
RLCA 	A		07	—	—	0	—	0	*

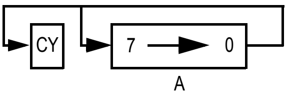
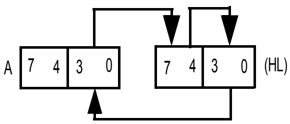
Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
RLD A[3:0] ← (HL)[7:4] (HL)[7:4] ← (HL)[3:0] (HL)[3:0] ← A[3:0] 			`ED 6F`	*	*	0	P	0	—
RR s 	(HL)		`CB 1E`	*	*	0	P	0	*
	(IX/Y+d)		`DD/FD CB dd 1E`						
	r		`CB 18–1F`						
RRA 	A		`1F`	—	—	0	—	0	*
RRC s 	(HL)		`CB 1E`	*	*	0	P	0	*
	(IX/Y+d)		`DD/FD CB dd 1E`						
	r		`CB 08–0F`						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

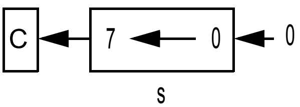
Table 37. Instruction Summary (Continued)

	Address Mode		Opcode(s)	Flags Affected					
Instruction and Operation	**Dest**	**Source**	**(Hex)**	**S**	**Z**	**H**	**P/V**	**N**	**C**
RRCA 			`0F`	—	—	0	—	0	*
RRD A[3:0] ← (HL)[3:0] (HL)[3:0] ← (HL)[7:4] (HL)[7:4] ← A[3:0] 			`ED 67`	*	*	0	P	0	—
RSMIX MADL ← 0			`ED 7E`	—	—	—	—	—	—

RST n
(SP) ← PC
if MADL=1 {
    (SP) ← ADL
}
PC ← {`0000h`,**n**}
			`C7-FF`	—	—	—	—	—	—

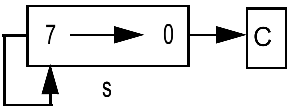
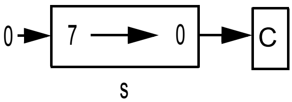
Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
SBC A, **s** A ← A–**s**–C		(HL)	`9E`	*	*	*	V	1	*
		ir	`DD/FD 9C–9D`						
		(IX/Y+d)	`DD/FD 9E dd`						
		n	`DE`						
		r	`98–9F`						
SBC HL, **ss** HL ← HL–**ss**–C		**rr**	`ED 42–62`	*	*	*	V	1	*
		SP	`ED 72`						
SCF C ← 1			`37`	—	—	0	—	0	1
SET b, s **s**[**b**] ← 1		(HL)	`CB C6–FE`	—	—	—	—	—	—
		(IX/Y+d)	`DD/FD CB dd C6–FE`						
		r	`CB C0–FF`						
SLA s 		(HL)	`CB 26`	*	*	0	P	0	*
		(IX/Y+d)	`DD/FD CB dd 26`						
		r	`CB 20–27`						
SLP			`ED 76`	—	—	—	—	—	—

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	S	Z	H	P/V	N	C
SRA s	(HL)		CB 2E	*	*	0	P	0	*
	**(IX/Y+d)**		DD/FD CB dd 2E						
	r		CB 28–2F						
SRL s	(HL)		CB 3E	*	*	0	P	0	*
	**(IX/Y+d)**		DD/FD CB dd 3E						
	r		CB 38–3F						
STMIX MADL ← 1			ED 7D	—	—	—	—	—	—
SUB A,**s** A ← A–**s**		(HL)	96	*	*	*	V	1	*
		ir	DD/FD 94–95						
		(IX/Y+d)	DD/FD 96 dd						
		n	D6						
		r	90–97						
TST A,**s** A **AND s**		(HL)	ED 34	*	*	1	P	0	0
		n	ED 64						
		r	ED 04–3C						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

Table 37. Instruction Summary (Continued)

Instruction and Operation	Address Mode Dest	Address Mode Source	Opcode(s) (Hex)	Flags Affected S	Z	H	P/V	N	C
TSTIO n {0000h, C} **AND n**			ED 74	*	*	1	P	0	0
XOR A,s A ← A **XOR s**		(HL)	AE	*	*	0	P	0	0
		ir	DD/FD AC–AD						
		(IX/Y+d)	DD/FD AE dd						
		n	EE						
		r	A8–AF						

Note: *This flag value is a function of the result of the affected operation.
— = No Change.
0 = Set to 0.
1 = Set to 1.
V = Set to 1 if overflow occurs.
X = Undetermined.
P = Set to the parity of the result (0 if odd parity, 1 if even parity).
IEF2 = The value of Interrupt Enable Flag 2.

eZ80® CPU Instruction Set Description

The following pages provide detailed descriptions of the assembly language instructions available with the eZ80® CPU. Some CPU-based products may not support all instructions, registers, operating modes, etc. Refer to the *eZ80® and eZ80Acclaim!® product specifications* for information on CPU usage. The instruction set descriptions on the following pages are organized alphabetically by mnemonic.

eZ80® CPU Instruction Cycle Times

The instruction execution cycle time information provided for each of the following CPU instructions refers to the bus cycles required to execute the instruction. This cycle time information appears in the Attributes tables under the heading **Cycle**. The number of clock cycles required to execute the instruction is a function of the number of bus cycles, the number of wait states in use, and whether or not conditional operations are performed.

ADC A, (HL)
ADD with Carry

Operation

A ← A+(HL)+C

Description

The (HL) operand is an 8-bit value retrieved from the memory location specified by the contents of the multibyte register HL. This 8-bit value and the Carry Flag (C) are added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	A,(HL)	X	2	`8E`
ADC.S	A,(HL)	1	3	`52, 8E`
ADC.L	A,(HL)	0	3	`49, 8E`

ADC A, ir

ADD with Carry

Operation

A ← A+**ir**+C

Description

The **ir** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The **ir** operand and the Carry Flag (C) are added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	A,IXH	X	2	`DD, 8C`
ADC	A,IXL	X	2	`DD, 8D`
ADC	A,IYH	X	2	`FD, 8C`
ADC	A,IYL	X	2	`FD, 8D`

ADC A, (IX/Y+d)

ADD with Carry

Operation

A ← A+(**IX/Y**+**d**)+C

Description

(**IX/Y**+**d**) is an 8-bit value stored in the memory location specified by the Index Register, IX or IY, offset by the two's-complement displacement **d**. This 8-bit value and the Carry Flag (C) are added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	A,(IX+**d**)	X	4	`DD, 8E, dd`
ADC.S	A,(IX+**d**)	1	5	`52, DD, 8E, dd`
ADC.L	A,(IX+**d**)	0	5	`49, DD, 8E, dd`
ADC	A,(IY+**d**)	X	4	`FD, 8E, dd`
ADC.S	A,(IY+**d**)	1	5	`52, FD, 8E, dd`
ADC.L	A,(IY+**d**)	0	5	`49, FD, 8E, dd`

ADC A, n
ADD with Carry

Operation

A ← A+**n**+C

Description

The 8-bit immediate value **n** and the Carry Flag (C) are added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	A,**n**	X	2	`CE, nn`

ADC A, r
ADD with Carry

Operation

A ← A+**r**+C

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand and the Carry Flag (C) are added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes in Table 38.

Table 38. Register and `jj` Opcodes for ADC A, r Instruction (hex)

Register	`jj`
A	`8F`
B	`88`
C	`89`
D	`8A`
E	`8B`
H	`8C`
L	`8D`

ADC HL, rr

ADD with Carry

Operation

HL ← HL+**rr**+C

Description

The **rr** operand is any of the multibyte registers BC, DE, or HL. The **rr** operand and the Carry Flag (C in the F register) are added to the contents of the HL register. The result is stored in the HL register.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 11; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from MSB; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	HL,**ss**	X	2	`ED, kk`
ADC.S	HL,**ss**	1	3	`52, ED, kk`
ADC.L	HL,**ss**	0	3	`49, ED, kk`

`kk` identifies the BC, DE, or HL register and is assembled into one of the opcodes in Table 39.

Table 39. Register and kk Opcodes for ADC HL, rr instruction (hex)

Register	kk
BC	4A
DE	5A
HL	6A

ADC HL, SP

ADD with Carry

Operation

HL ← HL+**SP**+C

Description

The Stack Pointer and the Carry Flag (C in the F register) are added to the contents of the HL register. The result is stored in the HL register. In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, SPS is used for **SP**.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 11; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from MSB; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADC	HL,**SP**	X	2	`ED, 7A`
ADC.S	HL,**SP**	1	3	`52, ED, 7A`
ADC.L	HL,**SP**	0	3	`49, ED, 7A`

ADD A, (HL)
ADD without Carry

Operation

A ← A+(HL)

Description

The (HL) operand is an 8-bit value retrieved from the memory location specified by the contents of the multibyte register HL. This 8-bit value is added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	A,(HL)	X	2	86
ADD.S	A,(HL)	1	3	52, 86
ADD.L	A,(HL)	0	3	49, 86

ADD A, ir

ADD without Carry

Operation

A ← A+**ir**

Description

The **ir** operand is any of IXH, IXL, IYH, or IYL. The **ir** operand is added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	A,IXH	X	2	`DD, 84`
ADD	A,IXL	X	2	`DD, 85`
ADD	A,IYH	X	2	`FD, 84`
ADD	A,IYL	X	2	`FD, 85`

ADD A, (IX/Y+d)

ADD without Carry

Operation

A ← A+(**IX/Y**+**d**)

Description

The (**IX/Y**+**d**) operand is an 8-bit value retrieved from the memory location specified by the contents of the Index Register, IX or IY, offset by the two's complement displacement **d**. This 8-bit value is added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	A,(IX+**d**)	X	4	`DD, 86, dd`
ADD.S	A,(IX+**d**)	1	5	`52, DD, 86, dd`
ADD.L	A,(IX+**d**)	0	5	`49, DD, 86, dd`
ADD	A,(IY+**d**)	X	4	`FD, 86, dd`
ADD.S	A,(IY+**d**)	1	5	`52, FD, 86, dd`
ADD.L	A,(IY+**d**)	0	5	`49, FD, 86, dd`

ADD A, n

ADD without Carry

Operation

A ← A+**n**

Description

The 8-bit immediate value **n** is added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	A,**n**	X	2	`C6, nn`

ADD A, r

ADD without Carry

Operation

A ← A+**r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand is added to the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Reset.
C	Set if carry from bit 7; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes in Table 40.

Table 40. Register and `jj` Opcodes for ADD A, r Instruction (hex)

Register	`jj`
A	87
B	80
C	81
D	82
E	83
H	84
L	85

ADD HL, rr

ADD without Carry

Operation

HL ← HL+**rr**

Description

The **rr** operand is any of the multibyte registers BC, DE, or HL. The CPU adds the contents of the **rr** register to the contents of the HL register, and stores the results in the HL register.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Set if carry from bit 11; reset otherwise.
P/V	Not affected.
N	Reset.
C	Set if carry from MSB; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	HL,**rr**	X	1	`kk`
ADD.S	HL,**rr**	1	2	`52, kk`
ADD.L	HL,**rr**	0	2	`49, kk`

`kk` identifies the BC, DE, or HL register and is assembled into one of the opcodes in Table 41.

Table 41. Register and `kk` Opcodes for ADD HL, rr Instruction (hex)

Register	`kk`
BC	09
DE	19
HL	29

ADD HL, SP

ADD without Carry

Operation

HL ← HL+**SP**

Description

The CPU adds the contents of the multibyte Stack Pointer (**SP**) register to the contents of the HL register, and stores the results in the HL register. In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, SPS is used for **SP**.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Set if carry from bit 11; reset otherwise.
P/V	Not affected.
N	Reset.
C	Set if carry from MSB; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	HL,**SP**	X	1	`39`
ADD.S	HL,**SP**	1	2	`52, 39`
ADD.L	HL,**SP**	0	2	`49, 39`

ADD IX/Y, rxy
ADD without Carry

Operation

IX/Y ← IX/Y+rxy

Description

The **rxy** operand is any of the multibyte BC, DE, or **IX/Y** registers. The CPU adds the contents of the multibyte register **rxy** to the contents of the Index Register, IX or IY, and stores the results in the Index Register, IX or IY.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Set if carry from bit 11; reset otherwise.
P/V	Not affected.
N	Reset.
C	Set if carry from MSB; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	IX,**rxy**	X	2	`DD, kk`
ADD.S	IX,**rxy**	1	3	`52, DD, kk`
ADD.L	IX,**rxy**	0	3	`49, DD, kk`
ADD	IY,**rxy**	X	2	`FD, kk`
ADD.S	IY,**rxy**	1	3	`52, FD, kk`
ADD.L	IY,**rxy**	0	3	`49, FD, kk`

`kk` identifies the BC, DE, or **IX/Y** register and is assembled into one of the opcodes in Table 42.

Table 42. Register and `kk` Opcodes for ADD IX/Y, rxy Instruction (hex)

Register	kk
BC	09
DE	19
IX/IY	29 (destination is the same as the source) **IX ← IX+IX or IY ← IY+IY**

ADD IX/Y, SP

ADD without Carry

Operation

IX/Y ← IX/Y+SP

Description

The CPU adds the contents of the multibyte Stack Pointer register (**SP**) to the contents of the Index Register, IX or IY, and stores the results in the Index Register, IX or IY. In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, SPS is used for **SP**.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Set if carry from bit 11; reset otherwise.
P/V	Not affected.
N	Reset.
C	Set if carry from MSB; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
ADD	IX,**SP**	X	2	`DD, 39`
ADD.S	IX,**SP**	1	3	`52, DD, 39`
ADD.L	IX,**SP**	0	3	`49, DD, 39`
ADD	IY,**SP**	X	2	`FD, 39`
ADD.S	IY,**SP**	1	3	`52, FD, 39`
ADD.L	IY,**SP**	0	3	`49, FD, 39`

AND A, (HL)
Logical AND

Operation

A ← A **AND** (HL)

Description

The (HL) operand is the 8-bit value stored at the memory location indicated by the contents of the multibyte HL register. This 8-bit value is bitwise ANDed with the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
AND	A,(HL)	X	2	`A6`
AND.S	A,(HL)	1	3	`52, A6`
AND.L	A,(HL)	0	3	`49, A6`

AND A, ir
Logical AND

Operation

A ← A **AND ir**

Description

The **ir** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The **ir** operand is bitwise ANDed with the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
AND	A,IXH	X	2	`DD, A4`
AND	A,IXL	X	2	`DD, A5`
AND	A,IYH	X	2	`FD, A4`
AND	A,IYL	X	2	`FD, A5`

AND A, (IX/Y+d)

Logical AND

Operation

A ← A **AND** (**IX/Y**+**d**)

Description

The (**IX/Y**+**d**) operand is the 8-bit value stored in the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value is bitwise ANDed with the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
AND	A,(IX+**d**)	X	4	`DD, A6, dd`
AND.S	A,(IX+**d**)	1	5	`52, DD, A6, dd`
AND.L	A,(IX+**d**)	0	5	`49, DD, A6, dd`
AND	A,(IY+**d**)	X	4	`FD, A6, dd`
AND.S	A,(IY+**d**)	1	5	`52, FD, A6, dd`
AND.L	A,(IY+**d**)	0	5	`49, FD, A6, dd`

AND A, n
Logical AND

Operation

A ← A **AND n**

Description

The 8-bit immediate value **n** is bitwise ANDed with the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
AND	A,**n**	X	2	`E6, nn`

AND A, r

Logical AND

Operation

A ← A **AND r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand is bitwise ANDed with the contents of the accumulator, A. The result is stored in the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
AND	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes in Table 43.

Table 43. Register and `jj` Opcodes for AND A, r Instruction (hex)

Register	jj
A	A7
B	A0
C	A1
D	A2
E	A3
H	A4
L	A5

Bit b, (HL)

Bit Test

Operation

Z ← ~(HL)[**b**]

Description

The (HL) operand is an 8-bit value stored at the memory location specified by the contents of the multibyte register HL. This instruction tests bit **b** of this 8-bit value and sets the 0 Flag (Z) if the bit is 0. The Z Flag is reset if bit **b** of operand (HL) is a one.

Condition Bits Affected

S	Undefined.
Z	Set if bit **b** is 0; reset otherwise.
H	Set.
P/V	Undefined.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
BIT	**b**,(HL)	X	3	`CB, kk`
BIT.S	**b**,(HL)	1	4	`52, CB, kk`
BIT.L	**b**,(HL)	0	4	`49, CB, kk`

`kk`=binary code `01 bbb 110`, where `bbb` identifies the bit tested and assembled into the object code, as indicated in Table 44.

Table 44. Bit tested, bb values, and `kk` Opcode for Bit B, (HL) Instruction (hex)

Bit	**bb**	**kk**
0	000	46
1	001	4E
2	010	56
3	011	5E
4	100	66
5	101	6E
6	110	76
7	111	7E

Bit b, (IX/Y+d)

Bit Test

Operation

Z ← ~(**IX/Y**+**d**)[**b**]

Description

The (**IX/Y**+**d**) operand is an 8-bit value stored at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This instruction tests bit **b** of this 8-bit value and sets the 0 Flag (Z) if the bit is 0. The Z Flag is reset if bit **b** of operand (HL) is a one.

Condition Bits Affected

S	Undefined.
Z	Set if bit **b** is 0; reset otherwise.
H	Set.
P/V	Undefined.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
BIT	**b**,(IX+**d**)	X	5	`DD, CB, dd, kk`
BIT.S	**b**,(IX+**d**)	1	6	`52, DD, CB, dd, kk`
BIT.L	**b**,(IX+**d**)	0	6	`49, DD, CB, dd, kk`
BIT	**b**,(IY+**d**)	X	5	`FD, CB, dd, kk`
BIT.S	**b**,(IY+**d**)	1	6	`52, FD, CB, dd, kk`
BIT.L	**b**,(IY+**d**)	0	6	`49, FD, CB, dd, kk`

kk=binary code 01 bbb 110, where bbb identifies the bit tested and assembled into the object code, as indicated in Table 45.

Table 45. Bit test, bb, and kk Opcodes for Bit B, (IX/Y+d) Instruction (hex)

Register	bb	kk
0	000	46
1	001	4E
2	010	56
3	011	5E
4	100	66
5	101	6E
6	110	76
7	111	7E

Bit b, r

Bit Test

Operation

Z ← ~r[b]

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. This instruction tests bit **b** in the specified register and sets the 0 Flag (Z) if the bit is 0. The Z Flag is reset if bit **b** of register **r** is a one.

Condition Bits Affected

S	Undefined.
Z	Set if bit **b** is 0; reset otherwise.
H	Set.
P/V	Undefined.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
BIT	**b,r**	X	2	`CB, jj`

`jj`=binary code `01 bbb rrr`; where `rrr` identifies the A, B, C, D, E, H, or L register and `bbb` identifies the bit tested and assembled into the object code, as indicated in Table 46.

Table 46. Register, bbb, and rrr Opcodes for Bit b, r Instruction (hex)

Bit Tested	bbb	Register	rrr
0	000	A	111
1	001	B	000
2	010	C	001
3	011	D	010
4	100	E	011
5	101	H	100
6	110	L	101
7	111		

CALL cc, Mmn

Conditional CALL Subroutine

Operation

if **cc** {
  (SP) ← PC
  PC ← **Mmn**
}

Description

If condition **cc** is true (1), the return address is pushed onto the stack. The return address is the address of the instruction immediately following this **CALL** instruction. The Program Counter (PC) is loaded with the **Mmn** operand, and execution continues at the new PC address. The **Mmn** operand is a 16- or 24-bit address, depending on the instruction suffix and the ADL mode. Table 47 offers detailed information.

Table 47. Conditional Operations for CALL cc, Mmn Instruction

ADL	Suffix	Operation if condition cc is true (1)
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Push a 2-byte return address, PC15:0], onto the SPS stack. The ADL mode bit remains cleared to 0. Load a 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
1	None	The starting Program Counter is PC[23:0]. Push the 3-byte return address, PC[23:0],onto the SPL stack. The ADL mode bit remains set to 1. Load a 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending Program Counter is PC[23:0] = {MM, mm, nn}.

Table 47. Conditional Operations for CALL cc, Mmn Instruction

0	**.IS**	The starting Program Counter is {MBASE, PC[15:0]}. Push the 2-byte logical return address, PC[15:0], onto the {MBASE, SPS} stack. Push a `02h` byte onto the SPL stack, indicating a call from Z80 mode (because ADL = 0). The ADL mode bit remains cleared to 0. Load a 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}.
1	**.IS**	The starting Program Counter is PC[23:0]. Push the 2 LS bytes of the return address, PC[15:0], onto the {MBASE, SPS} stack. Push the MS byte of the return address, PC[23:16], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating a call from ADL mode (because ADL = 1). Reset ADL mode bit to 0. Load a 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
0	**.IL**	The starting Program Counter is {MBASE, PC[15:0]}. Push the 2-byte logical return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating a call from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Load the 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending Program Counter is PC[23:0] = {MM, mm, nn}.
1	**.IL**	The starting Program Counter is PC[23:0]} Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating a call from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Load a 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending Program Counter is PC[23:0] = {MM, mm, nn}.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CALL	**cc,mn**	0	3/6	`kk, nn, mm`
CALL	**cc,Mmn**	1	4/7	`kk, nn, mm, MM`
CALL.IS	**cc,mn**	0	4/7	`40, kk, nn, mm`
CALL.IS	**cc,mn**	1	4/8	`49, kk, nn, mm`
CALL.IL	**cc,Mmn**	0	5/8	`52, kk, nn, mm, MM`
CALL.IL	**cc,Mmn**	1	5/9	`5B, kk, nn, mm, MM`

The opcode (`kk`) depends on the condition code being tested. According to the relevant condition code, the opcode is assembled as indicated in Table 48.

Table 48. Opcode Assembly for CALL cc, Mmn Instruction (hex)

Condition	Relevant Flag	Opcode
NZ (non 0)	Z	`C4`
Z (0)	Z	`CC`
NC (no carry)	C	`D4`
C (carry)	C	`DC`
PO (parity odd)	P/V	`E4`
PE (parity even)	P/V	`EC`
P (sign positive)	S	`F4`
M (sign negative/minus)	S	`FC`

CALL Mmn

CALL Subroutine

Operation

(SP) ← PC
PC ← **Mmn**

Description

The return address is pushed onto the stack. The return address is the address of the instruction immediately following this **CALL** instruction. Then the Program Counter (PC) is loaded with the **Mmn** operand and execution continues at the new PC address. The **Mmn** operand is a 16- or 24-bit address, depending on the instruction suffix and the ADL mode. Table 49 offers more detailed information.

Table 49. Detail of the CALL Mmn Instruction

ADL	Suffix	Operation
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Push a 2-byte return address, PC15:0], onto the SPS stack. The ADL mode bit remains cleared to 0. Load a 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
1	None	The starting Program Counter is PC[23:0]. Push the 3-byte return address, PC[23:0],onto the SPL stack. The ADL mode bit remains set to 1. Load a 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending Program Counter is PC[23:0] = {MM, mm, nn}.

Table 49. Detail of the CALL Mmn Instruction

0	**.IS**	The starting Program Counter is {MBASE, PC[15:0]} Push the 2-byte logical return address, PC[15:0], onto the {MBASE, SPS} stack. Push a `02h` byte onto the SPL stack, indicating a call from Z80 mode (because ADL = 0). The ADL mode bit remains cleared to 0. Load a 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}.
1	**.IS**	The starting Program Counter is PC[23:0]. Push the 2 LS bytes of the return address, PC[15:0], onto the {MBASE, SPS} stack. Push the MS byte of the return address, PC[23:16], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating a call from ADL mode (because ADL = 1). Reset ADL mode bit to 0. Load a 2-byte logical address {mm, nn} from the instruction into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
0	**.IL**	The starting Program Counter is {MBASE, PC[15:0]}. Push the 2-byte logical return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating a call from Z80 mode (because ADL = 0). Set the ADL mode bit to 1. Load the 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending Program Counter is PC[23:0] = {MM, mm, nn}.
1	**.IL**	The starting Program Counter is PC[23:0]} Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating a call from ADL mode (because ADL = 1). The ADL mode bit remains set to 1. Load a 3-byte address {MM, mm, nn} from the instruction into PC[23:0]. The ending Program Counter is PC[23:0] = {MM, mm, nn}.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CALL	**mn**	0	5	`CD, nn, mm`
CALL	**Mmn**	1	7	`CD, nn, mm, MM`
CALL.IS	**mn**	0	7	`40, CD, nn, mm`
CALL.IS	**mn**	1	8	`49, CD, nn, mm`
CALL.IL	**Mmn**	0	8	`52, CD, nn, mm, MM`
CALL.IL	**Mmn**	1	9	`5B, CD, nn, mm, MM`

CCF

Complement Carry Flag

Operation

C ← ~C

Description

The Carry Flag bit (C) in the F register is inverted.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Previous carry is copied.
P/V	Not affected.
N	Reset.
C	Set if carry was cleared to 0 before operation; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CCF	—	X	1	3F

CP A, (HL)

Compare with Accumulator

Operation

A–(HL)

Description

The (HL) operand is an 8-bit value stored at the memory location specified by the contents of the multibyte register HL. This 8-bit value is compared with (subtracted from) the contents of the accumulator, A. The execution of this instruction does not affect the contents of the accumulator or the (HL) operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=(HL); reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow: reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CP	A,(HL)	X	2	BE
CP.S	A,(HL)	1	3	52, BE
CP.L	A,(HL)	0	3	49, BE

CP A, ir

Compare with Accumulator

Operation

A–**ir**

Description

The **ir** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The **ir** operand is compared with (subtracted from) the contents of the accumulator, A. The execution of this instruction does not affect the contents of the accumulator or the **ir** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=**ir**; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow: reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CP	A,IXH	X	2	`DD, BC`
CP	A,IXL	X	2	`DD, BD`
CP	A,IYH	X	2	`FD, BC`
CP	A,IYL	X	2	`FD, BD`

CP A, (IX/Y+d)

Compare with Accumulator

Operation

A–(**IX/Y**+**d**)

Description

The (**IX/Y**+**d**) operand is the 8-bit value stored in the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value is compared with (subtracted from) the contents of the accumulator, A. The execution of this instruction does not affect the contents of the accumulator or the (**IX/Y**+**d**) operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=(**IX/Y**+**d**); reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow: reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CP	A,(IX+**d**)	X	4	`DD, BE, dd`
CP.S	A,(IX+**d**)	1	5	`52, DD, BE, dd`
CP.L	A,(IX+**d**)	0	5	`49, DD, BE, dd`
CP	A,(IY+**d**)	X	4	`FD, BE, dd`
CP.S	A,(IY+**d**)	1	5	`52, FD, BE, dd`
CP.L	A,(IY+**d**)	0	5	`49, FD, BE, dd`

CP A, n

Compare with Accumulator

Operation

A–**n**

Description

The 8-bit immediate value **n** is compared with (subtracted from) the contents of the accumulator, A. The execution of this instruction does not affect the contents of the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=**n**; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow: reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CP	A,**n**	X	2	`FE nn`

CP A, r

Compare with Accumulator

Operation

A–**r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU compares the **r** operand to the contents of the accumulator, A, and outputs the difference. The execution of this instruction does not affect the contents of the accumulator or the **r** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=**r**; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow: reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CP	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 50.

Table 50. Register and `jj` Opcodes for CP A, r Instruction (hex)

Register	jj
A	BF
B	B8
C	B9
D	BA
E	BB
H	BC
L	BD

CPD

Compare and Decrement

Operation

A–(HL)
HL ← HL−1
BC ← BC−1

Description

The CPU compares the contents of the accumulator, A, to the memory location that the HL register points to, and outputs the difference. This instruction does not affect the contents of the reference memory location or the accumulator. The HL and BC registers decrement.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=(HL); reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if BC−1 ≠ 0; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CPD	—	X	3	`ED, A9`
CPD.S	—	1	4	`52, ED, A9`
CPD.L	—	0	4	`49, ED, A9`

CPDR

Compare and Decrement with Repeat

Operation

repeat {
  A–(HL)
  HL ← HL – 1
  BC ← BC – 1
} while (~Z and BC ≠ 0)

Description

The CPU compares the contents of the accumulator, A, to the memory location that the HL register points to and outputs the difference. This instruction does not affect the contents of the reference memory location or the accumulator. The HL and BC registers decrement. This operation is repeated until one of the following two conditions is met:

1. A=(HL), which sets the 0 Flag (Z).
2. BC is decremented to 0, which resets the P/V Flag.

In Z80 mode, the BC register is 16 bits, which allows the CPDR instruction to repeat a maximum of 65536 (64K) times. In ADL mode, the BC register is 24 bits, which allows the CPDR instruction to repeat a maximum of 16,777,216 (16M) times.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=(HL); reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if BC – 1 ≠ 0; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CPDR	—	X	1 + 3 * BC	`ED, B9`
CPDR.S	—	1	2 + 3 * BC	`52, ED, B9`
CPDR.L	—	0	2 + 3 * BC	`49, ED, B9`

CPI
Compare and Increment

Operation

A–(HL)
HL ← HL+1
BC ← BC−1

Description

The CPU compares the contents of the accumulator, A, to the memory location that the HL register points to and outputs the difference. This instruction does not affect the contents of the reference memory location or the accumulator. The HL register increments, while the BC register decrements.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=(HL); reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if BC−1 ≠ 0; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CPI	—	X	3	`ED, A1`
CPI.S	—	1	4	`52, ED, A1`
CPI.L	—	0	4	`49, ED, A1`

CPIR

Compare and Increment with Repeat

Operation

repeat {
  A–(HL)
  HL ← HL+1
  BC ← BC−1
} while (~Z and BC ≠ 0)

Description

The CPU compares the contents of the accumulator, A, to the memory location that the HL register points to and outputs the difference. This instruction does not affect the contents of the reference memory location or the accumulator. The HL register increments, while the BC register decrements. This operation is repeated until one of the following two condition is met:

1. A=(HL), which sets the 0 Flag (Z).
2. BC is decremented to 0, which resets the P/V Flag.

In Z80 mode, the BC register is 16 bits, which allows the CPIR instruction to repeat a maximum of 65536 (64K) times. In ADL mode, the BC register is 24 bits, which allows the CPIR instruction to repeat a maximum of 16,777,216 (16M) times.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if A=(HL); reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if BC−1 ≠ 0; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CPIR	—	X	1 + 3 * BC	`ED, B1`
CPIR.S	—	1	2 + 3 * BC	`52, ED, B1`
CPIR.L	—	0	2 + 3 * BC	`49, ED, B1`

CPL
Complement Accumulator

Operation

A ← ~A

Description

All bits in the accumulator, A, are inverted (1's complemented).

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Set.
P/V	Not affected.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
CPL	—	X	1	2F

DAA

Decimal Adjust Accumulator

Operation

A ← Decimal Adjust (A)

Description

This instruction conditionally adjusts the accumulator, A, following addition and subtraction operations on binary-coded-decimal (BCD) values. For addition (**ADD**, **ADC**, **INC**) or subtraction (**SUB**, **SBC**, **DEC**, **NEG**), Table 51 indicates the operation performed by the **DAA** instruction.

Table 51. Operations of the DAA Instruction

Operation	C Before DAA	Hex Value in Upper Digit (Bits 7:4)	H Before DAA	Hex Value in Lower Digit (Bits 3:0)	Number Added to Byte	C After DAA	H After DAA
ADD, **ADC**, or **INC**	0	0–9	0	0–9	00	0	0
	0	0–8	0	A–F	06	0	1
	0	0–9	1	0–3	06	0	0
	0	A–F	0	0–9	60	1	0
	0	9–F	0	A–F	66	1	1
	0	A–F	1	0–3	66	1	0
	1	0–2	0	0–9	60	1	0
	1	0–2	0	A–F	66	1	1
	1	0–3	1	0–3	66	1	0
SUB, **SBC**, **DEC**, or **NEG**	0	0–9	0	0–9	00	0	0
	0	0–8	1	6–F	FA	0	0
	1	7–F	0	0–9	A0	1	0
	1	6–F	1	6–F	9A	1	0

Condition Bits Affected

S	Set if the msb of the result is 1 after the operation; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	See Table 51 on page 161.
P/V	Set if the result is even parity after the operation; reset otherwise.
N	Not affected.
C	See Table 51 on page 161.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DAA	—	X	1	27

Example

If an addition operation is performed between 15 (BCD) and 27 (BCD), simple decimal arithmetic yields the following result:

$$\begin{array}{r} 15 \\ \underline{+27} \\ 42 \end{array}$$

However, when the binary representations are added in the accumulator according to standard binary arithmetic, the resulting sum is an invalid BCD value:

$$\begin{array}{rl} 0001\ 0101 & =15\ (\text{BCD}) \\ \underline{+0010\ 0111} & =\underline{27\ (\text{BCD})} \\ 0011\ 1100 & =3\text{C (invalid BCD value)} \end{array}$$

The **DAA** instruction adjusts the result so that the correct BCD representation is obtained:

0011 1100	=	3C(invalid BCD value)
+0000 0110	=	06 (BCD)
0100 0010	=	42 (BCD result)

Before operating, the **DAA** instruction checks the Carry Flag (C) and the Half-Carry Flag (H) to determine if a decimal adjustment is required as a result of the preceding BCD arithmetic operation.

DEC (HL)

Decrement

Operation

(HL) ← (HL)–1

Description:

The (HL) operand is an 8-bit value stored at the memory location specified by the contents of the multibyte register HL. This 8-bit value is decremented by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set is borrowed from bit 4; reset otherwise.
P/V	Set if operand was `80h` before operation; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	(HL)	X	4	`35`
DEC.S	(HL)	1	5	`52, 35`
DEC.L	(HL)	0	5	`49, 35`

DEC ir

Decrement

Operation

ir ← **ir**−1

Description:

The **ir** operand is any of 8-bit CPU registers IXH, IXL, IYH, or IYL. The value contained in the specified register is decremented by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set is borrowed from bit 4; reset otherwise.
P/V	Set if operand was `80h` before operation; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	IXH	X	2	`DD, 25`
DEC	IXL	X	2	`DD, 2D`
DEC	IXH	X	2	`FD, 25`
DEC	IXL	X	2	`FD, 2D`

DEC IX/Y
Decrement

Operation

IX/Y ← IX/Y−1

Description

The value contained in the specified Index Register, IX or IY, is decremented by 1.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	IX	X	2	`DD, 2B`
DEC.S	IX	1	3	`52, DD, 2B`
DEC.L	IX	0	3	`49, DD, 2B`
DEC	IY	X	2	`FD, 2B`
DEC.S	IY	1	3	`52, FD, 2B`
DEC.L	IY	0	3	`49, FD, 2B`

DEC (IX/Y+d)
Decrement

Operation

(**IX/Y**+**d**) ← (**IX/Y**+**d**)–1

Description:

The (**IX/Y**+**d**) operand is the 8-bit value stored in the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value contained in the specified register is decremented by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set is borrowed from bit 4; reset otherwise.
P/V	Set if operand was `80h` before operation; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	(IX+**d**)	X	6	`DD, 35, dd`
DEC.S	(IX+**d**)	1	7	`52, DD, 35, dd`
DEC.L	(IX+**d**)	0	7	`49, DD, 35, dd`
DEC	(IY+**d**)	X	6	`FD, 35, dd`
DEC.S	(IY+**d**)	1	7	`52, FD, 35, dd`
DEC.L	(IY+**d**)	0	7	`49, FD, 35, dd`

DEC r

Decrement

Operation

r ← **r**–1

Description:

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The value contained in the specified register is decremented by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set is borrowed from bit 4; reset otherwise.
P/V	Set if operand was `80h` before operation; reset otherwise.
N	Set.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 52.

Table 52. Register and `jj` Opcodes for DEC r Instruction (hex)

Register	`jj`
A	`3D`
B	`05`
C	`0D`
D	`15`
E	`1D`
H	`25`
L	`2D`

DEC rr

Decrement

Operation

rr ← **rr**–1

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The value contained in the specified register is decremented by 1.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	**rr**	X	1	`kk`
DEC.S	**rr**	1	2	`52, kk`
DEC.L	**rr**	0	2	`49, kk`

`kk` identifies the BC, DE, or HL register and is assembled into one of the opcodes indicated in Table 53.

Table 53. Register and `kk` Opcodes for DEC rr Instruction (hex)

Register	kk
BC	`0B`
DE	`1B`
HL	`2B`

DEC SP
Decrement

Operation

SP ← **SP**–1

Description

The value contained in the Stack Pointer (**SP**) register is decremented by 1. In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, the SPS is used for **SP**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DEC	**SP**	X	1	3B
DEC.S	**SP**	1	2	52, 3B
DEC.L	**SP**	0	2	49, 3B

DI

Disable Interrupt

Operation

IEF1 ← 0
IEF2 ← 0

Description:

This instruction disables the maskable interrupts by resetting the interrupt enable flags (IEF1 and IEF2).

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DI	—	X	1	F3

DJNZ d

Decrement B Jump not 0

Operation

$B \leftarrow B-1$
if $B \neq 0$ {
    $PC \leftarrow PC+\mathbf{d}$
}

Description

The B register decrements by 1. If the resultant value in register B is not 0, the two's-complement displacement **d** is added to the value of the Program Counter. The jump is measured from the address of the instruction opcode following this instruction.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
DJNZ	**d**	X	2/4	`10, dd`

EI
Enable Interrupt

Operation

IEF1 ← 1
IEF2 ← 1

Description

This instruction sets the interrupt enable flags (IEF1 and IEF2) to a logical 1, which allows any maskable interrupt to be recognized.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
EI	—	X	1	FB

EX AF, AF’

Exchange AF and AF’

Operation

A ↔ A’
F ↔ F’

Description

The CPU exchanges the contents of the accumulator, A, and the Flag register, F, with the contents of the alternate accumulator, A’, and alternate Flag register, F’, respectively.

Condition Bits Affected

All condition bits are replaced with the values from the alternate Flag register, F’.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
EX	AF,AF’	X	1	08

EX DE, HL

Exchange DE with HL

Operation

DE ↔ HL

Description

The CPU exchanges the contents of the DE register with the contents of the HL register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
EX	DE,HL	X	1	EB

EX (SP), HL

Exchange Stack and HL Register

Operation

```
if ADL = 1 {
    (SPL) ↔ HL[7:0]
    (SPL+1) ↔ HL[15:8]
    (SPL+2) ↔ HL[23:16]
}
else if ADL = 0 {
    SPS ↔ HL[7:0]
    (SPS+1) ↔ HL[15:8]
}
```

Description

The CPU exchanges the contents of the multibyte CPU register HL with the contents of the memory location specified by the Stack Pointer (**SP**). In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, SPS is used for **SP**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
EX	**(SP)**,HL	0/1	5/7	E3
EX.S	**(SP)**,HL	1	6	52, E3
EX.L	**(SP)**,HL	0	8	49, E3

EX (SP), IX/Y

Exchange Stack and Index Register

Operation

if ADL=1 {
    (SPL) ↔ **IX/Y**[7:0]
    (SPL+1) ↔ **IX/Y**[15:8]
    (SPL+2) ↔ **IX/Y**[23:16]
}
else if ADL=0 {
    SPS ↔ **IX/Y**[7:0]
    (SPS+1) ↔ **IX/Y**[15:8]
}

Description

The CPU exchanges the contents of the multibyte Index Register, IX or IY, with the memory location specified by the Stack Pointer (**SP**). In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, SPS is used for **SP**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
EX	**(SP)**,IX	0/1	6/8	`DD, E3`
EX.S	**(SP)**,IX	1	7	`52, DD, E3`
EX.L	**(SP)**,IX	0	9	`49, DD, E3`
EX	**(SP)**,IY	0/1	6/8	`FD, E3`
EX.S	**(SP)**,IY	1	7	`52, FD, E3`
EX.L	**(SP)**,IY	0	9	`49, FD, E3`

EXX

Exchange Working Register Set with Alternate Register Set

Operation

BC ↔ BC’
DE ↔ DE’
HL ↔ HL’

Description

The CPU exchanges the contents of the primary working registers BC, DE, and HL with the alternate working registers BC’, DE’, and HL’, respectively.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
EXX	—	X	1	D9

HALT
Halt

Operation

while **HALT** {
  NOP
}

Description

The **HALT** instruction suspends CPU operation until a subsequent interrupt or reset is received. While in HALT mode, the CPU executes NOPs. The Program Counter, PC, stops incrementing while in HALT mode.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
HALT	—	X	1	76

IM n

Set Interrupt Mode

Operation

Select the appropriate interrupt mode from Interrupt Mode 0, Interrupt Mode 1, and Interrupt Mode 2.

Description

The **n** operand is any of the following interrupt modes:

- Interrupt Mode 0—in this mode, the interrupting device inserts an instruction on the data bus during an interrupt acknowledge cycle.
- Interrupt Mode 1—in this mode, the CPU responds to an interrupt by executing a restart to location `000038h`.
- Interrupt Mode 2—in this mode, the interrupting device places the low-order address of the interrupt vector on the data bus during an interrupt acknowledge cycle. The I register provides the high-order byte of the interrupt vector table address. The 16-bit value at the address {I, DATA[7:0]} is the starting address of the interrupt service routine.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IM	0	X	2	`ED, 46`
IM	1	X	2	`ED, 56`
IM	2	X	2	`ED, 5E`

Attributes

Not all eZ80® products support the three interrupt modes. Refer to the individual product specification for information on supported interrupt modes.

IN A, (n)

Input from I/O

Operation

A ← ({UU, A, **n**})

Description

The **n** operand is placed on the lower byte of the address bus, ADDR[7:0]; the contents of the accumulator, A, are placed on the middle byte of the address bus, ADDR[15:8]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The byte at I/O address {UU, A, **n**} is written to the accumulator.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IN	(**n**)	X	3	`DB, nn`

IN r, (BC)—also IN r, (C) for Z80 compatibility

Input from I/O

Operation

r ← ({UU, BC[15:0]})

Description

The CPU places the contents of the 16-bit BC multibyte register onto the lower two bytes of the address bus at ADDR[15:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The byte located at I/O address {UU, BC[15:0]} is written to the specified register **r** (A, B, C, D, E, H, or L).

Condition Bits Affected

S	Set if byte is negative; reset otherwise.
Z	Set if byte is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IN	**r**, (BC)	X	3	`ED, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 54.

Table 54. Register and `jj` Opcodes for IN r, (BC) and IN r, (C) Instructions (hex)

Register	`jj`
A	78
B	40
C	48
D	50
E	58
H	60
L	68

IN0 r, (n)

Input from I/O

Operation

r ← ({UU, 00h, **n**})

Description

The **n** operand is placed on the lower byte of the address bus, ADDR[7:0], while the High byte of the address bus, ADDR[15:8], is forced to 0. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses.The byte at this I/O address is written to the specified register **r** (A, B, C, D, E, H, or L).

Condition Bits Affected

S	Set if byte is negative; reset otherwise.
Z	Set if byte is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IN0	**r,(n)**	X	4	`ED, jj, nn`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 55

Table 55. Register and jj Opcodes for IN0 r, (n) Instruction (hex)

Register	jj
A	38
B	00
C	08
D	10
E	18
H	20
L	28

INC (HL)
Increment

Operation

(HL) ← (HL)+1

Description

The (HL) operand is an 8-bit value stored at the memory location specified by the contents of the multibyte register HL. This 8-bit value increments by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3.
P/V	Set if operand was `7Fh` before operation; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	(HL)	X	4	`34`
INC.S	(HL)	1	5	`52, 34`
INC.L	(HL)	0	5	`49, 34`

INC ir
Increment

Operation

ir ← **ir**+1

Description

The **ir** operand is any of the 8-bit CPU registers IXH, IXL, IYH, IYL. The contents of the specified register increment by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3.
P/V	Set if operand was `7Fh` before operation; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	IXH	X	2	`DD, 24`
INC	IXL	X	2	`DD, 2C`
INC	IYH	X	2	`FD, 24`
INC	IYL	X	2	`FD, 2C`

INC IX/Y

Increment

Operation

IX/Y ← **IX/Y**+1

Description

The CPU increments the contents of the specified Index Register, IX or IY, by 1. In Z80 mode, or when the **.S** suffix is employed, **IX/Y**[23:16] ← `00h`.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	IX	X	2	`DD, 23`
INC.S	IX	1	3	`52, DD, 23`
INC.L	IX	0	3	`49, DD, 23`
INC	IY	X	2	`FD, 23`
INC.S	IY	1	3	`52, FD, 23`
INC.L	IY	0	3	`49, FD, 23`

INC (IX/Y+d)

Increment

Operation

(**IX/Y**+**d**) ← (**IX/Y**+**d**)+1

Description

The (**IX/Y**+**d**) operand is an 8-bit register at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU increments the contents of this 8-bit register by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3.
P/V	Set if operand was `7Fh` before operation.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	(IX+**d**)	X	6	`DD, 34, dd`
INC.S	(IX+**d**)	1	7	`52, DD, 34, dd`
INC.L	(IX+**d**)	0	7	`49, DD, 34, dd`
INC	(IY+**d**)	X	6	`FD, 34, dd`
INC.S	(IY+**d**)	1	7	`52, FD, 34, dd`
INC.L	(IY+**d**)	0	7	`49, FD, 34, dd`

INC r

Increment

Operation

r ← **r**+1

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU increments the contents of the specified register **r** by 1.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if carry from bit 3.
P/V	Set if operand was `7Fh` before operation; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 56.

Table 56. Register and `jj` Opcodes for INC r Instruction (hex)

Register	`jj`
A	3C
B	04
C	0C
D	14
E	1C
H	24
L	2C

INC rr

Increment

Operation

rr ← **rr**+1

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The CPU increments the contents of the specified register by 1. In Z80 mode, or when the **.S** suffix is employed, **rr**[23:16] ← 00h.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	**rr**	X	1	kk
INC.S	**rr**	1	2	52, kk
INC.L	**rr**	0	2	49, kk

kk identifies the BC, DE, or HL register and is assembled into one of the opcodes indicated in Table 57.

Table 57. Register and kk Opcodes for INC rr Instruction (hex)

Register	kk
BC	03
DE	13
HL	23

INC SP

Increment

Operation

SP ← **SP**+1

Description

The CPU increments the contents of the Stack Pointer register (**SP**) by 1. In ADL mode, or when the **.L** suffix is employed, SPL is used for **SP**. In Z80 mode, or when the **.S** suffix is employed, SPS is used for **SP**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INC	**SP**	X	1	33
INC.S	**SP**	1	2	52, 33
INC.L	**SP**	0	2	49, 33

IND

Input from I/O and Decrement

Operation

(HL) ← ({UU, BC[15:0]})
B ← B−1
HL ← HL−1

Description

The CPU places the contents of BC[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at this I/O address into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. Next, the CPU decrements the B and HL registers and sets the Z Flag to 1 if the B register is decremented to 0.

Condition Bits Affected

S	Not affected.
Z	Set if B-1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IND	—	X	5	`ED, AA`
IND.S	—	1	6	`52, ED, AA`
IND.L	—	0	6	`49, ED, AA`

IND2

Input from I/O and Decrement

Operation

(HL) ← ({UU, BC[15:0]})
B ← B–1
C ← C–1
HL ← HL–1

Description

The CPU places the contents of BC[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16], is undefined for I/O addresses. The CPU reads the byte located at this I/O address into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. Next, the CPU decrements the B, C, and HL registers, and sets the Z Flag to 1 if the B register is decremented to 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IND2	—	x	5	`ED, 8C`
IND2.S	—	1	6	`52, ED, 8C`
IND2.L	—	0	6	`49, ED, 8C`

IND2R

Input from I/O and Decrement with Repeat

Operation

repeat {
  (HL) ← ({UU, DE[15:0]})
  BC ← BC–1
  DE ← DE–1
  HL ← HL–1
} while BC ≠ 0

Description

The CPU places the contents of DE[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16], is undefined for I/O addresses. The CPU reads the byte at this I/O address into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. Next, the CPU decrements the BC, DE, and HL registers, and sets the Z Flag to 1 if the BC register is decremented to 0. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if BC–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
IND2R	—	X	2 + 3 * BC	`ED, 9C`
IND2R.S	—	1	3 + 3 * BC	`52, ED, 9C`
IND2R.L	—	0	3 + 3 * BC	`49, ED, 9C`

Note

This instruction operates differently in the eZ80190 devicedevice. In the eZ80190, operation is:

repeat {
  (HL) ← ({UU, BC[15:0]})
  B ← B–1
  C ← C–1
  HL ← HL–1
} while B ≠ 0

INDM

Input from I/O and Decrement

Operation

(HL) ← ({UU, `00h`, C})
B ← B–1
C ← C–1
HL ← HL–1

Description

The CPU places the contents of register C onto the lower byte of the address bus, ADDR[7:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at this I/O address into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. Next, the CPU decrements the B, C, and HL registers, and sets the Z Flag to 1 if the B register is decremented to 0.

Condition Bits Affected

S	Undefined.
Z	Set if B–1=0; reset otherwise.
H	Undefined.
P/V	Undefined.
N	Set if msb of data is a logical 1; reset otherwise.
C	Undefined.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INDM	—	X	5	`ED, 8A`
INDM.S	—	1	6	`52, ED, 8A`
INDM.L	—	0	6	`49, ED, 8A`

INDMR

Input from I/O and Decrement with Repeat

Operation

repeat {
  (HL) ← ({UU, `00h`,C})
  B ← B–1
  C ← C–1
  HL ← HL–1
} while B ≠ 0

Description

The CPU places the contents of register C onto the lower byte of the address bus, ADDR[7:0], and places a 0 onto the High byte of the address bus, ADDR[15:8]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at I/O address {UU, `00h`, C} into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The CPU decrements the B, C, and HL registers, and sets the Z Flag to 1 if the B register is decremented to 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INDMR	—	X	2 + 3 * B	`ED, 9A`
INDMR.S	—	1	3 + 3 * B	`52, ED, 9A`
INDMR.L	—	0	3 + 3 * B	`49, ED, 9A`

INDR

Input from I/O and Decrement with Repeat

Operation

repeat {
  (HL) ← ({UU, BC[15:0]})
  B ← B–1
  HL ← HL–1
} while B ≠ 0

Description

The CPU places the contents of BC[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at I/O address {UU, BC[15:0]} into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. Next, the CPU decrements the B and HL registers, and sets the Z Flag to 1 if the B register is decremented to 0. The instruction repeats until the B register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set of B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INDR	—	X	2 + 3 * B	ED, BA
INDR.S	—	1	3 + 3 * B	52, ED, BA
INDR.L	—	0	3 + 3 * B	49, ED, BA

INDRX

Input from I/O and Decrement Memory Address with Stationary I/O Address

Operation

repeat {
    (HL) ← ({UU, DE[15:0]})
    BC ← BC – 1
    HL ← HL – 1
} while BC ≠ 0

Description

The CPU places the contents of register DE onto the lower byte of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16], is undefined for I/O addresses. The CPU reads the byte at this I/O address, {UU, DE[15:0]}, into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The BC and HL registers decrement. Next, the CPU sets the Z Flag to 1 if the BC register decrements to 0. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set of BC – 1 = 0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if Bit 7 of data = 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INDRX	—	X	2 + 3 * BC	`ED, CA`
INDRX.S	—	1	3 + 3 * BC	`52, ED, CA`
INDRX.L	—	0	3 + 3 * BC	`49, ED, CA`

Note

This instruction is not supported on the eZ80190 device.

INI

Input from I/O and Increment

Operation

(HL) ← ({UU, BC[15:0]})
B ← B–1
HL ← HL+1

Description

The CPU places the contents of BC[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at I/O address {UU, BC[15:0]} into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The B register decrements and the HL register increments. Next, the CPU sets the Z Flag to 1 if the B register decrements to 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INI	—	X	5	`ED, A2`
INI.S	—	1	6	`52, ED, A2`
INI.L	—	0	6	`49, ED, A2`

INI2

Input from I/O and Increment

Operation

(HL) ← ({UU, BC[15:0]})
B ← B–1
C ← C+1
HL ← HL+1

Description

The CPU places the contents of BC[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at this I/O address into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The B register decrements. The C and HL registers increment. Next, the CPU sets the Z Flag to 1 if the B register decrements to 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INI2	—	—	5	`ED, 84`
INI2.S	—	—	6	`52, ED, 84`
INI2.L	—	—	6	`49, ED, 84`

INI2R

Input from I/O and Increment with Repeat

Operation

repeat {
    (HL) ← ({UU, DE[15:0]})
    BC ← BC−1
    DE ← DE+1
    HL ← HL+1
} while BC ≠ 0

Description

The CPU places the contents of DE[15:0] onto the lower two bytes of the address bus, ADDR[15:0], and places a 0 onto the upper byte of the address bus, ADDR[23:16]. The CPU reads the byte at this I/O address into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The BC register decrements. The DE and HL registers increment. Next, the CPU sets the Z Flag to 1 if the BC register decrements to 0. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if BC−1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INI2R	—	X	2 + 3 * BC	`ED, 94`
INI2R.S	—	1	3 + 3 * BC	`52, ED, 94`
INI2R.L	—	0	3 + 3 * BC	`49, ED, 94`

Note

This instruction operates differently in the eZ80190 device. In the eZ80190, operation is:

repeat {
    (HL) ← ({UU, BC[15:0]})
    B ← B−1
    C ← C+1
    HL ← HL+1
} while B ≠ 0

INIM

Input from I/O and Increment

Operation

(HL) ← ({UU, `00h`, C})
B ← B–1
C ← C+1
HL ← HL+1

Description

The CPU places the contents of register C onto the lower byte of the address bus, ADDR[7:0], and places a 0 onto the High byte of the address bus, ADDR[15:8]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at I/O address {UU, `00h`, C} into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The B register decrements. The C and HL registers increment. The Z Flag is set to 1 if the B register decrements to 0.

Condition Bits Affected

S	Undefined.
Z	Set if B–1=0; reset otherwise.
H	Undefined.
P/V	Undefined.
N	Set if msb of data is a logical 1; reset otherwise.
C	Undefined.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INIM	—	X	5	`ED, 82`
INIM.S	—	1	6	`52, ED, 82`
INIM.L	—	0	6	`49, ED, 82`

INIMR

Input from I/O and Increment with Repeat

Operation

repeat {
  (HL) ← ({UU, `00h`, C})
  B ← B–1
  C ← C+1
  HL ← HL+1
} while B ≠ 0

Description

The CPU places the contents of register C onto the lower byte of the address bus, ADDR[7:0], and places a 0 onto the High byte of the address bus, ADDR[15:8]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at I/O address {UU, `00h`, C} into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The B register decrements. The C and HL registers increment. Next, the CPU sets the Z Flag to 1 if the B register decrements to 0. The instruction repeats until the B register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INIMR	—	X	2 + 3 * B	ED, 92
INIMR.S	—	1	3 + 3 * B	52, ED, 92
INIMR.L	—	0	3 + 3 * B	49, ED, 92

INIR

Input from I/O and Increment with Repeat

Operation

```
repeat {
    (HL) ← ({UU, BC[15:0]})
    B ← B–1
    HL ← HL+1
} while B ≠ 0
```

Description

The CPU places the contents of BC[15:0] onto the lower two bytes of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU reads the byte located at I/O address {UU, BC[15:0]} into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The B register decrements and the HL register increments. Next, the CPU sets the Z Flag to 1 if the B register decrements to 0. The instruction repeats until the B register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is a logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INIR	—	X	2 + 3 * B	ED, B2
INIR.S	—	1	3 + 3 * B	52, ED, B2
INIR.L	—	0	3 + 3 * B	49, ED, B2

INIRX

Input from I/O and Increment Memory Address with Stationary I/O Address

Operation

repeat {
    (HL) ← ({UU, DE[15:0]})
    BC ← BC−1
    HL ← HL+1
} while BC ≠ 0

Description

The CPU places the contents of register DE onto the lower byte of the address bus, ADDR[15:0]. The upper byte of the address bus, ADDR[23:16], is undefined for I/O addresses. The CPU reads the byte at this I/O address, {UU, DE[15:0]}, into CPU memory. The CPU next places the contents of HL onto the address bus and writes the byte to the memory address specified by the HL register. The BC register decrements. The HL register increments. Next, the CPU sets the Z Flag to 1 if the BC register decrements to 0. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set of BC−1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if Bit 7 of data = 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
INIRX	—	X	2 + 3 * BC	`ED, C2`
INIRX.S	—	1	3 + 3 * BC	`52, ED, C2`
INIRX.L	—	0	3 + 3 * BC	`49, ED, C2`

Note

This instruction is not supported on the eZ80190 device.

JP cc, Mmn

Conditional Jump

Operation

if **cc** {
PC ← **Mmn**
}

Description

If the condition is true (a logical 1), the Program Counter, PC, is loaded with the instruction operand. When assembled, the first byte after the opcode is the low-order byte of the operand. Table 58 provides more detailed information on this instruction, particularly when switching between ADL and Z80 modes. The information in Table 58 is only valid if the condition is true.

Table 58. JP cc, Mmn Instruction Detail

ADL	Suffix	Operation (if condition cc is true)
0	None or **.SIS**	The starting Program Counter is {MBASE, PC[15:0]}. Write the 2-byte immediate value {mm, nn}, to PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
1	None or **.LIL**	The starting Program Counter is PC[23:0]. Write the 3-byte immediate value {MM, mm, nn}, to PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0] = {MM, mm, nn}.
0	**.LIL**	The starting Program Counter is {MBASE, PC[15:0]}. Write the 3-byte immediate value {MM, mm, nn}, to PC[23:0]. Set the ADL mode bit to 1. The ending Program Counter is PC[23:0] = {MM, mm, nn}.
1	**.SIS**	The starting Program Counter is PC[23:0]. Write the 2-byte immediate value {mm, nn}, to PC[15:0]. Reset the ADL mode bit to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
X	**.SIL**	An illegal suffix for this instruction.
X	**.LIS**	An illegal suffix for this instruction.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
JP	**cc,mn**	0	3/4	`kk, nn, mm`
JP	**cc,Mmn**	1	4/5	`kk, nn, mm, MM`
JP.SIS	**cc,mn**	X	4/5	`40, kk, nn, mm`
JP.LIL	**cc,Mmn**	X	5/6	`5B, kk, nn, mm, MM`

Condition	Relevant Flag	Opcode (hex)
NZ (non 0)	**Z**	`C2`
Z (0)	**Z**	`CA`
NC (no carry)	**C**	`D2`
C (carry)	**C**	`DA`
PO (parity odd)	**P/V**	`E2`
PE (parity even)	**P/V**	`EA`
P (sign positive)	**S**	`F2`
M (sign negative/minus)	**S**	`FA`

JP (HL)

Jump Indirect

Operation

PC ← HL

Description

The Program Counter is loaded with the contents of the multibyte CPU register HL. Table 59 provides more detailed information on this instruction, particularly when switching between ADL and Z80 modes.

Table 59. JP (HL) Instruction Detail

ADL	Suffix	Operation
0	None or **.S**	The starting Program Counter is {MBASE, PC[15:0]}. Write the 2-byte value stored in HL[15:0] to PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, HL[15:0]}.
1	None or .L	The starting Program Counter is PC[23:0]. Write the 3-byte value stored in HL[23:0] to PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0]=HL[23:0].
0	.L	The starting Program Counter is {MBASE, PC[15:0]}. Write the 3-byte value stored in HL[23:0] to PC[23:0]. Set the ADL mode bit to 1. The ending Program Counter is PC[23:0]=HL[23:0].
1	**.S**	The starting Program Counter is PC[23:0]. Write the 2-byte value stored in HL[15:0] to PC[15:0]. Reset ADL mode bit to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, HL[15:0]}.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
JP	(HL)	0/1	3	`E9`
JP.S	(HL)	1	4	`52, E9`
JP.L	(HL)	0	4	`49, E9`

JP (IX/Y)

Jump Indirect

Operation

PC ← **IX/Y**

Description

The Program Counter is loaded with the contents of the specified Index Register, IX or IY. Table 60 provides more detailed information on this instruction, particularly when switching between ADL and Z80 modes.

Table 60. JP (IX/Y) Instruction Detail

ADL	Suffix	Operation
0	None or **.S**	The starting Program Counter is {MBASE, PC[15:0]}. Write the 2-byte value stored in **IX/Y**[15:0] to PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, **IX/Y**[15:0]}.
1	None or .L	The starting Program Counter is PC[23:0]. Write the 3-byte value stored in **IX/Y**[23:0] to PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0]=**IX/Y**[23:0].
0	.L	The starting Program Counter is {MBASE, PC[15:0]}. Write the 3-byte value stored in **IX/Y**[23:0] to PC[23:0]. Set the ADL mode bit to 1. The ending Program Counter is PC[23:0]=**IX/Y**[23:0].
1	**.S**	The starting Program Counter is PC[23:0]. Write the 2-byte value stored in **IX/Y**[15:0] to PC[15:0]. Reset ADL mode bit to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, **IX/Y**[15:0]}.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
JP	(IX)	1	4	`DD, E9`
JP.S	(IX)	X	5	`40, DD, E9`
JP.L	(IX)	X	5	`5B, DD, E9`
JP	(IY)	1	4	`FD, E9`
JP.S	(IY)	X	5	`40, FD, E9`
JP.L	(IY)	X	5	`5B, FD, E9`

JP Mmn

Jump

Operation

PC ← **Mmn**

Description

The Program Counter is loaded with the instruction operand. When assembled, the first byte after the opcode is the low-order byte of the operand. Table 61 provides more detailed information on this instruction, particularly when switching between ADL and Z80 modes.

Table 61. JP Mmn Instruction Detail

ADL	Suffix	Operation
0	None or **.SIS**	The starting Program Counter is {MBASE, PC[15:0]}. Write the 2-byte immediate value {mm, nn}, to PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
1	None or **.LIL**	The starting Program Counter is PC[23:0]. Write the 3-byte immediate value {MM, mm, nn}, to PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0] = {MM, mm, nn}.
0	**.LIL**	The starting Program Counter is {MBASE, PC[15:0]}. Write the 3-byte immediate value {MM, mm, nn}, to PC[23:0]. Set the ADL mode bit to 1. The ending Program Counter is PC[23:0] = {MM, mm, nn}.
1	**.SIS**	The starting Program Counter is PC[23:0]. Write the 2-byte immediate value {mm, nn}, to PC[15:0]. Reset the ADL mode bit to 0. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, mm, nn}.
X	**.SIL**	An illegal suffix for this instruction.
X	**.LIS**	An illegal suffix for this instruction.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
JP	**mn**	0	4	`C3, nn, mm`
JP	**Mmn**	1	5	`C3, nn, mm, MM`
JP.SIS	**mn**	X	5	`40, C3, nn, mm`
JP.LIL	**Mmn**	X	6	`5B, C3, nn, mm, MM`

JR cc’, d

Conditional Jump Relative

Operation

if **cc’** {
  PC ← PC+**d**
}

Description

If the condition **cc’** (NZ, Z, NC or C) is true (a logical 1), then the two’s-complement displacement **d** is added to the Program Counter. The jump is measured from the address of the byte following the instruction.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
JR	**cc’,d**	X	2/3	`kk, dd`

The opcode `kk` depends on the condition code being tested. According to the relevant condition code, the opcode is assembled as indicated in Table 62.

Table 62. Opcode Assembly for JR cc’, d Instruction

Condition	Relevant Flag	Opcode (hex)
NZ (non 0)	Z	`20`
Z (0)	Z	`28`
NC (no carry)	C	`30`
C (carry)	C	`38`

JR d

Jump Relative

Operation

PC ← PC+**d**

Description

The two's-complement displacement **d** is added to the Program Counter. The jump is measured from the address of the byte following the instruction.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
JR	**d**	X	3	`18, dd`

LD A, I
Load Accumulator

Operation

A ← I[7:0]

Description

The CPU writes the contents of the lower byte of the Interrupt Vector register, I[7:0], to the accumulator, A.

Condition Bits Affected

S	Set if the I register is negative; reset otherwise.
Z	Set if the I register is 0; reset otherwise.
H	Reset.
P/V	Contains contents of IEF2.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A, I	X	2	`ED, 57`

LD A, (IX/Y+d)

Load Accumulator

Operation

A ← (**IX/Y**+**d**)

Description

The CPU writes the contents of the memory location specified by the contents of the IX or IY register offset by the two's-complement displacement, **d**, to the accumulator, A.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A,(IX+**d**)	X	4	`DD, 7E, dd`
LD.S	A,(IX+**d**)	1	5	`52, DD, 7E, dd`
LD.L	A,(IX+**d**)	0	5	`49, DD, 7E, dd`
LD	A,(IY+**d**)	X	4	`FD, 7E, dd`
LD.S	A,(IY+**d**)	1	5	`52, FD, 7E, dd`
LD.L	A,(IY+**d**)	0	5	`49, FD, 7E, dd`

LD A, MB
Load Accumulator

Operation

A ← MBASE

Description

The CPU writes the contents of the Memory Base register, MBASE, to the accumulator, A. In Z80 mode, no operation occurs (two-cycle NOP).

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A,MB	1	2	`ED, 6E`

LD A, (Mmn)

Load Accumulator

Operation

A ← **(Mmn)**

Description

The CPU writes the contents of the specified memory location, **Mmn**, to the accumulator, A.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A,**(mn)**	0	4	`3A, nn, mm`
LD	A,**(Mmn)**	1	5	`3A, nn, mm, MM`
LD.LIL	A,**(Mmn)**	0	6	`5B, 3A, nn, mm, MM`
LD.SIS	A,**(mn)**	1	5	`40, 3A, nn, mm`

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

LD A, R
Load Accumulator

Operation

A ← R

Description

The CPU writes the contents of the Refresh Counter register, R, to the accumulator, A.

Condition Bits Affected

S	Set if the R register is negative; reset otherwise.
Z	Set if the R register is 0; reset otherwise.
H	Reset.
P/V	Contains contents of IEF2.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A,I	X	2	`ED, 5F`

LD A, (rr)

Load Accumulator

Operation

A ← **(rr)**

Description

The **rr** operand is any of BC, DE, or HL. The CPU writes the contents of the memory location specified by the multibyte register to the accumulator, A.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A,(BC)	X	2	0A
LD.S	A,(BC)	1	3	52, 0A
LD.L	A,(BC)	0	3	49, 0A
LD	A,(DE)	X	2	1A
LD.S	A,(DE)	1	3	52, 1A
LD.L	A,(DE)	0	3	49, 1A
LD	A,(HL)	X	2	7E
LD.S	A,(HL)	1	3	52, 7E
LD.L	A,(HL)	0	3	40, 7E

LD HL, I

Load Register

Operation

HL ← I

Description

The CPU writes the contents of the 16-bit Interrupt Vector register, I, to the multibyte register, HL.

Condition Bits Affected

S	Set if the I register is negative; reset otherwise.
Z	Set if the I register is 0; reset otherwise.
H	Reset.
P/V	Contains contents of IEF2.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	HL, I	X	2	`ED, D7`

Note

This instruction is not supported on the eZ80190, eZ80L92, or eZ80F92/F93 devices.

LD (HL), IX/Y

Load Indirect

Operation

(HL) ← **IX/Y**

Description

The CPU writes the contents of the multibyte Index Register IX or IY to the memory location specified by the contents of the multibyte HL register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(HL),IX	0/1	4/5	`ED, 3F`
LD.S	(HL),IX	1	5	`52, ED, 3F`
LD.L	(HL),IX	0	6	`49, ED, 3F`
LD	(HL),IY	0/1	4/5	`ED, 3E`
LD.S	(HL),IY	1	5	`52, ED, 3E`
LD.L	(HL),IY	0	6	`49, ED, 3E`

LD (HL), n
Load Indirect

Operation

(HL) ← **n**

Description

The 8-bit immediate **n** operand is written to the memory location specified by the contents of the multibyte CPU register HL.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(HL),**n**	X	3	`36, nn`
LD.S	(HL),**n**	1	4	`52, 36, nn`
LD.L	(HL),**n**	0	4	`49, 36, nn`

LD (HL), r
Load Indirect

Operation

(HL) ← **r**

Description

The **r** operand is any of A, B, C, D, E, H, L. The CPU stores the contents of the specified register into the memory location specified by the contents of the multibyte HL register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(HL),**r**	X	2	`jj`
LD.S	(HL),**r**	1	3	`52, jj`
LD.L	(HL),**r**	0	3	`49, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 63.

Table 63. Register and `jj` Opcodes for LD (HL), r Instruction (hex)

Register	`jj`	Register	`jj`
A	`77`	E	`73`
B	`70`	H	`74`
C	`71`	L	`75`
D	`72`		

LD (HL), rr

Load Indirect

Operation

(HL) ← **rr**

Description

The **rr** operand is any of the multibyte registers BC, DE, or HL. The CPU writes the contents of the multibyte register **rr** to the memory location specified by the contents of the multibyte HL register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(HL),BC	0/1	4/5	`ED, 0F`
LD.S	(HL),BC	1	5	`52, ED, 0F`
LD.L	(HL),BC	0	6	`49, ED, 0F`
LD	(HL),DE	0/1	4/5	`ED, 1F,`
LD.S	(HL),DE	1	5	`52, ED, 1F`
LD.L	(HL),DE	0	6	`49, ED, 1F`
LD	(HL),HL	0/1	4/5	`ED, 2F`
LD.S	(HL),HL	1	5	`52, ED, 2F`
LD.L	(HL),HL	0	6	`49, ED, 2F`

LD I, HL

Load Interrupt Vector

Operation

I ← HL

Description

The CPU writes the contents of the accumulator, HL, to the 16-bit Interrupt Vector register, I.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	I,HL	X	2	`ED, C7`

Note

This instruction is not supported on the eZ80190, eZ80L92, or eZ80F92/F93 devices.

LD I, A

Load Interrupt Vector

Operation

I[7:0] ← A

Description

The CPU writes the contents of the accumulator, A, to the lower byte of the Interrupt Vector register, I[7:0].

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	I,A	X	2	`ED, 47`

LD ir, ir'

Load

Operation

ir ← ir'

Description

The **ir** and **ir'** operands are any of the 8-bit CPU registers IXH, IXL, IYH, or IYL. The CPU writes the contents of the specified register **ir'** to the selected register **ir**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IXH,IXH	X	2	`DD, 64`
LD	IXH,IXL	X	2	`DD, 65`
LD	IXL,IXH	X	2	`DD, 6C`
LD	IXL,IXL	X	2	`DD, 6D`
LD	IYH,IYH	X	2	`FD, 64`
LD	IYH,IYL	X	2	`FD, 65`
LD	IYL,IYH	X	2	`FD, 6C`
LD	IYL,IYL	X	2	`FD, 6D`

LD ir, n

Load

Operation

ir ← n

Description

The **ir** operand is any of the 8-bit CPU registers IXH, IXL, IYH, or IYL. The 8-bit immediate value **n** is written to the specified register **ir**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IXH,**n**	X	2	`DD, 26`
LD	IXL,**n**	X	2	`DD, 2E`
LD	IYH,**n**	X	2	`FD, 26`
LD	IYL,**n**	X	2	`FD, 2E`

LD ir, r
Load

Operation

ir ← **r**

Description

The **ir** operand is any of the 8-bit CPU registers IXH, IXL, IYH, or IYL. The **r** operand is any of the 8-bit CPU registers A, B, C, D, or E. The CPU writes the contents of the specified register **r** to the selected register **ir**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IXH,**r**	X	2	`DD, jj`
LD	IXL,**r**	X	2	`DD, kk`
LD	IYH,**r**	X	2	`FD, jj`
LD	IYL,**r**	X	2	`FD, kk`

`jj` identifies the A, B, C, D, or E register and is assembled into one of the opcodes indicated in Table 64.

Table 64. Register and jj Opcodes for LD ir, r Instruction (hex)

Register	jj
A	67
B	60
C	61
D	62
E	63

kk identifies the A, B, C, D, or E register and is assembled into one of the opcodes indicated in Table 65.

Table 65. Register and kk Opcodes for LD ir, r Instruction (hex)

Register	kk
A	6F
B	68
C	69
D	6A
E	6B

LD IX/Y, (HL)
Load Index Register

Operation

IX/Y ← (HL)

Description

The CPU writes the contents of the memory location specified by the HL register to the multibyte Index Register, IX or IY.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IX,(HL)	0/1	4/5	`ED, 37`
LD.S	IX,(HL)	1	5	`52, ED, 37`
LD.L	IX,(HL)	0	6	`49, ED, 37`
LD	IY,(HL)	0/1	4/5	`ED, 31`
LD.S	IY,(HL)	1	5	`52, ED, 31`
LD.L	IY,(HL)	0	6	`49, ED, 31`

LD IX/Y, (IX/Y+d)
Load Index Register

Operation

IX/Y ← (**IX/Y**+**d**)

Description

The CPU writes the contents of the memory location, specified by the contents of the IX or IY register offset by the two's-complement displacement **d**, to the multibyte Index Register, IX or IY.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IX,(IX+**d**)	0/1	5/6	`DD, 37, dd`
LD.S	IX,(IX+**d**)	1	6	`52, DD, 37, dd`
LD.L	IX,(IX+**d**)	0	7	`49, DD, 37, dd`
LD	IY,(IX+**d**)	0/1	5/6	`DD, 31, dd`
LD.S	IY,(IX+**d**)	1	6	`52, DD, 31, dd`
LD.L	IY,(IX+**d**)	0	7	`49, DD, 31, dd`
LD	IX,(IY+**d**)	0/1	5/6	`FD, 31, dd`
LD.S	IX,(IY+**d**)	1	6	`52, FD, 31, dd`
LD.L	IX,(IY+**d**)	0	7	`49, FD, 31, dd`
LD	IY,(IY+**d**)	0/1	5/6	`FD, 37, dd`
LD.S	IY,(IY+**d**)	1	6	`52, FD, 37, dd`
LD.L	IY,(IY+**d**)	0	7	`49, FD, 37, dd`

LD IX/Y, Mmn

Load Index Register

Operation

IX/Y ← Mmn

Description

The immediate operand, **Mmn**, is written to the specified multibyte Index Register, IX or IY.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IX,**mn**	0	4	`DD, 21, nn, mm`
LD	IX,**Mmn**	1	5	`DD, 21, nn, mm, MM`
LD.LIL	IX,**Mmn**	0	6	`5B, DD, 21, nn, mm, MM`
LD.SIS	IX,**mn**	1	5	`40, DD, 21, nn, mm`
LD	IY,**mn**	0	4	`FD, 21, nn, mm`
LD	IY,**Mmn**	1	5	`FD, 21, nn, mm, MM`
LD.LIL	IY,**Mmn**	0	6	`5B, FD, 21, nn, mm, MM`
LD.SIS	IY,**mn**	1	5	`40, FD, 21, nn, mm`

LD IX/Y, (Mmn)

Load Index Register

Operation

IX/Y ← (Mmn)

Description

The 16- or 24-bit operand **(Mmn)** specifies a location in memory. The 16- or 24-bit value stored at this location in memory is written to the specified multibyte Index Register, IX or IY.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	IX,**(mn)**	0	6	`DD, 2A, nn, mm`
LD	IX,**(Mmn)**	1	8	`DD, 2A, nn, mm, MM`
LD.LIL	IX,**(Mmn)**	0	9	`5B, DD, 2A, nn, mm, MM`
LD.SIS	IX,**(mn)**	1	7	`40, DD, 2A, nn, mm`
LD	IY,**(mn)**	0	6	`FD, 2A, nn, mm`
LD	IY,**(Mmn)**	1	8	`FD, 2A, nn, mm, MM`
LD.LIL	IY,**(Mmn)**	0	9	`5B, FD, 2A, nn, mm, MM`
LD.SIS	IY,**(mn)**	1	7	`40, FD, 2A, nn, mm`

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

LD (IX/Y+d), IX/Y

Load Indirect with Offset

Operation

(**IX/Y**+**d**) ← **IX/Y**

Description

The CPU writes the contents of the Index Register, IX or IY, to the memory location specified by the contents of the multibyte Index Register, IX or IY, offset by the two's-complement displacement **d**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(IX+**d**),IX	0/1	5/6	`DD, 3F, dd`
LD.S	(IX+**d**),IX	1	6	`52, DD, 3F, dd`
LD.L	(IX+**d**),IX	0	7	`49, DD, 3F, dd`
LD	(IX+**d**),IY	0/1	5/6	`DD, 3E, dd`
LD.S	(IX+**d**),IY	1	6	`52, DD, 3E, dd`
LD.L	(IX+**d**),IY	0	7	`49, DD, 3E, dd`
LD	(IY+**d**),IX	0/1	5/6	`FD, 3E, dd`
LD.S	(IY+**d**),IX	1	6	`52, FD, 3E, dd`
LD.L	(IY+**d**),IX	0	7	`49, FD, 3E, dd`
LD	(IY+**d**),IY	0/1	5/6	`FD, 3F, dd`
LD.S	(IY+**d**),IY	1	6	`52, FD, 3F, dd`
LD.L	(IY+**d**),IY	0	7	`49, FD, 3F, dd`

LD (IX/Y+d), n

Load Indirect with Offset

Operation

(IX/Y+d) ← **n**

Description

The 8-bit immediate value **n** is written to the memory location specified by the contents of the multibyte Index Register, IX or IY, offset by the two's-complement displacement **d**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(IX+**d**),**n**	X	5	`DD, 36, dd, nn`
LD.S	(IX+**d**),**n**	1	6	`52, DD, 36, dd, nn`
LD.L	(IX+**d**),**n**	0	6	`49, DD, 36, dd, nn`
LD	(IY+**d**),**n**	X	5	`FD, 36, dd, nn`
LD.S	(IY+**d**),**n**	1	6	`52, FD, 36, dd, nn`
LD.L	(IY+**d**),**n**	0	6	`49, FD, 36, dd, nn`

LD (IX/Y+d), r

Load Indirect with Offset

Operation

(**IX**/**Y**+**d**) ← **r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU writes the contents of the **r** register to the memory location specified by the contents of the multibyte Index Register, IX or IY, offset by the two's-complement displacement **d**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(IX+**d**),**r**	X	4	`DD, jj, dd`
LD.S	(IX+**d**),**r**	1	5	`52, DD, jj, dd`
LD.L	(IX+**d**),**r**	0	5	`49, DD, jj, dd`
LD	(IY+**d**),**r**	X	4	`FD, jj, dd`
LD.S	(IY+**d**),**r**	1	5	`52, FD, jj, dd`
LD.L	(IY+**d**),**r**	0	5	`49, FD, jj, dd`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 66.

Table 66. Register and `jj` Opcodes for LD (IX/Y+d), r Instruction (hex)

Register	jj
A	77
B	70
C	71
D	72
E	73
H	74
L	75

LD (IX/Y+d), rr

Load Indirect with Offset

Operation

(**IX**/**Y**+**d**) ← **rr**

Description

The **rr** operand is any of the multibyte registers BC, DE, or HL. The CPU writes the contents of the multibyte **rr** register to the memory location specified by the contents of the multibyte Index Register, IX or IY, offset by the two's-complement displacement **d**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(IX+**d**),BC	0/1	5/6	`DD, 0F, dd`
LD.S	(IX+**d**),BC	1	6	`52, DD, 0F, dd`
LD.L	(IX+**d**),BC	0	7	`49, DD, 0F, dd`
LD	(IX+**d**),DE	0/1	5/6	`DD, 1F, dd`
LD.S	(IX+**d**),DE	1	6	`52, DD, 1F, dd`
LD.L	(IX+**d**),DE	0	7	`49, DD, 1F, dd`
LD	(IX+**d**),HL	0/1	5/6	`DD, 2F, dd`
LD.S	(IX+**d**),HL	1	6	`52, DD, 2F, dd`
LD.L	(IX+**d**),HL	0	7	`49, DD, 2F, dd`
LD	(IY+**d**),BC	0/1	5/6	`FD, 0F, dd`
LD.S	(IY+**d**),BC	1	6	`52, FD, 0F, dd`
LD.L	(IY+**d**),BC	0	7	`49, FD, 0F, dd`

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(IY+**d**),DE	0/1	5/6	FD, 1F, dd
LD.S	(IY+**d**),DE	1	6	52, FD, 1F, dd
LD.L	(IY+**d**),DE	0	7	49, FD, 1F, dd
LD	(IY+**d**),HL	0/1	5/6	FD, 2F, dd
LD.S	(IY+**d**),HL	1	6	52, FD, 2F, dd
LD.L	(IY+**d**),HL	0	7	49, FD, 2F, dd

LD MB, A
Load MBASE

Operation

MBASE ← A

Description

In ADL mode (ADL mode bit=1), the CPU writes the contents of the accumulator, A, to the MBASE register. otherwise., no operation occurs (two-cycle NOP).

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	MB,A	X	2	ED, 6D

LD (Mmn), A

Load Indirect

Operation

(Mmn) ← A

Description

The CPU stores the contents of the accumulator, A, into the memory location specified by **Mmn**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**(mn)**,A	0	4	`32, nn, mm`
LD	**(Mmn)**,A	1	5	`32, nn, mm, MM`
LD.IS	**(mn)**,A	1	5	`40, 32, nn, mm`
LD.IL	**(Mmn)**,A	0	5	`5B, 32, nn, mm, MM`

LD (Mmn), IX/Y

Load Indirect

Operation

(Mmn) ← IX/Y

Description

The CPU stores the contents of the multibyte Index Register, IX or IY, in the memory location specified by 16- or 24-bit constant **Mmn**.

Condition Bits Affected

None

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**(mn)**,IX	0	6	`DD, 22, nn, mm`
LD	**(Mmn)**,IX	1	8	`DD, 22, nn, mm, MM`
LD.SIS	**(mn)**,IX	1	7	`40, DD, 22, nn, mm`
LD.LIL	**(Mmn)**,IX	0	9	`5B, DD, 22, nn, mm, MM`
LD	**(mn)**,IY	0	6	`FD, 22, nn, mm`
LD	**(Mmn)**,IY	1	8	`FD, 22, nn, mm, MM`
LD.SIS	**(mn)**,IY	1	7	`40, FD, 22, nn, mm`
LD.LIL	**(Mmn)**,IY	0	9	`5B, FD, 22, nn, mm, MM`

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

LD (Mmn), rr

Load Indirect

Operation

(Mmn) ← rr

Description

The **rr** operand is any of the multibyte registers BC, DE, or HL. The CPU stores the contents of the multibyte register **rr** in the memory location specified by **Mmn**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**(mn)**,BC	0	6	`ED, 43, nn, mm`
LD	**(Mmn)**,BC	1	8	`ED, 43, nn, mm, MM`
LD.SIS	**(mn)**,BC	1	7	`40, ED, 43, nn, mm`
LD.LIL	**(Mmn)**,BC	0	9	`5B, ED, 43, nn, mm, MM`
LD	**(mn)**,DE	0	6	`ED, 53, nn, mm`
LD	**(Mmn)**,DE	1	8	`ED, 53, nn, mm, MM`
LD.SIS	**(mn)**,DE	1	7	`40, ED, 53, nn, mm`
LD.LIL	**(Mmn)**,DE	0	9	`5B, ED, 53, nn, mm, MM`
LD	**(mn)**,HL	0	5	`22, nn, mm`
LD	**(Mmn)**,HL	1	7	`22, nn, mm, MM`
LD.SIS	**(mn)**,HL	1	6	`40, 22, nn, mm`
LD.LIL	**(Mmn)**,HL	0	8	`5B, 22, nn, mm, MM`

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

LD (Mmn), SP

Load Indirect

Operation

(Mmn) ← SP

Description

The CPU stores the contents of the multibyte Stack Pointer Register **SP** in the memory location specified by **Mmn**. In ADL mode, if **SP** is chosen, Stack Pointer Long (SPL) is the source. In Z80 mode, if **SP** is chosen, Stack Pointer Short (SPS) is the source.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**(mn)**,**SP**	0	6	`ED, 73, nn, mm`
LD	**(Mmn)**,**SP**	1	8	`ED, 73, nn, mm, MM`
LD.SIS	**(mn)**,**SP**	1	7	`40, ED, 73, nn, mm`
LD.LIL	**(Mmn)**,**SP**	0	9	`5B, ED, 73, nn, mm, MM`

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

LD R, A
Load Refresh Counter

Operation

R ← A

Description

The CPU writes the contents of the accumulator, A, to the Refresh Counter register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	R, A	X	2	`ED, 4F`

LD r, (HL)

Load Register

Operation

r ← (HL)

Description

The **r** operand is any of A, B, C, D, E, H, or L. The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte CPU register HL. This 8-bit value is written to the specified **r** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**r**,(HL)	X	2	`jj`
LD.S	**r**,(HL)	1	3	`52, jj`
LD.L	**r**,(HL)	0	3	`49, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 67.

Table 67. Register and `jj` Opcodes for LD r, (HL) Instruction (hex)

Register	`jj`	Register	`jj`
A	`7E`	E	`5E`
B	`46`	H	`66`
C	`4E`	L	`6E`
D	`56`		

LD r, ir

Load Register

Operation

r ← ir

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, or E. The **ir** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The CPU writes the contents of the specified **ir** register to the selected **r** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	A,IXH	X	2	`DD, 7C`
LD	A,IXL	X	2	`DD, 7D`
LD	A,IYH	X	2	`FD, 7C`
LD	A,IYL	X	2	`FD, 7D`
LD	B,IXH	X	2	`DD, 44`
LD	B,IXL	X	2	`DD, 45`
LD	B,IYH	X	2	`FD, 44`
LD	B,IYL	X	2	`FD, 45`
LD	C,IXH	X	2	`DD, 4C`
LD	C,IXL	X	2	`DD, 4D`
LD	C,IYH	X	2	`FD, 4C`
LD	C,IYL	X	2	`FD, 4D`
LD	D,IXH	X	2	`DD, 54`

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	D,IXL	X	2	`DD, 55`
LD	D,IYH	X	2	`FD, 54`
LD	D,IYL	X	2	`FD, 55`
LD	E,IXH	X	2	`DD, 5C`
LD	E,IXL	X	2	`DD, 5D`
LD	E,IYH	X	2	`FD, 5C`
LD	E,IYL	X	2	`FD, 5D`

LD r, (IX/Y+d)

Load Register

Operation

r ← (**IX/Y+d**)

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The (**IX/Y+d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value is written to the specified **r** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**r**,(IX+**d**)	X	4	`DD, jj, dd`
LD.S	**r**,(IX+**d**)	1	5	`52, DD, jj, dd`
LD.L	**r**,(IX+**d**)	0	5	`49, DD, jj, dd`
LD	**r**,(IY+**d**)	X	4	`FD, jj, dd`
LD.S	**r**,(IY+**d**)	1	5	`52, FD, jj, dd`
LD.L	**r**,(IY+**d**)	0	5	`49, FD, jj, dd`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 68.

Table 68. Register and `jj` Opcodes for LD r, (IX/Y+d) Instruction (hex)

Register	jj
A	7E
B	46
C	4E
D	56
E	5E
H	66
L	6E

LD r, n

Load Register

Operation

r ← n

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The 8-bit immediate operand **n** is written to the specified **r** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**r,n**	X	2	`jj, nn`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 69.

Table 69. Register and `jj` Opcodes for LD r, n Instruction (hex)

Register	`jj`
A	`3E`
B	`06`
C	`0E`
D	`16`
E	`1E`
H	`26`
L	`2E`

LD r, r'
Load Register

Operation

r ← r'

Description

The **r** and **r'** operands are any of A, B, C, D, E, H, or L. The CPU writes the contents of the **r'** register to the **r** register. The **r'** register described here should not be confused with the registers in the alternate working register set.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**r,r'**	X	1	`jj`

`jj`=binary code `01 ddd sss` where `ddd` identifies the destination A, B, C, D, E, H, or L register and `sss` identifies the source A, B, C, D, E, H, or L register assembled in the object code, as indicated in Table 70.

Table 70. Register and `jj` Opcodes for LD r, r' Instruction (hex)

Register	`jj` (`ddd` or `sss`)
A	`111`
B	`000`
C	`001`

Table 70. Register and `jj` Opcodes for LD r, r' Instruction (hex)

Register	`jj` (`ddd` or `sss`)
D	`010`
E	`011`
H	`100`
L	`101`

LD rr, (HL)

Load Register

Operation

rr ← (HL)

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The CPU writes the contents of the memory location specified by the HL register to the multibyte **rr** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	BC,(HL)	0/1	4/5	`ED, 07`
LD.S	BC,(HL)	1	5	`52, ED, 07`
LD.L	BC,(HL)	0	6	`49, ED, 07`
LD	DE,(HL)	0/1	4/5	`ED, 17`
LD.S	DE,(HL)	1	5	`52, ED, 17`
LD.L	DE,(HL)	0	6	`49, ED, 17`
LD	HL,(HL)	0/1	4/5	`ED, 27`
LD.S	HL,(HL)	1	5	`52, ED, 27`
LD.L	HL,(HL)	0	6	`49, ED, 27`

LD rr, (IX/Y+d)

Load Register

Operation

rr ← (**IX/Y**+**d**)

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The CPU writes the contents of the memory location, specified by the contents of the IX or IY register offset by the two's-complement displacement **d**, to the multibyte **rr** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	BC,(IX+**d**)	0/1	5/6	`DD, 07, dd`
LD.S	BC,(IX+**d**)	1	6	`52, DD, 07, dd`
LD.L	BC,(IX+**d**)	0	7	`49, DD, 07, dd`
LD	DE,(IX+**d**)	0/1	5/6	`DD, 17, dd`
LD.S	DE,(IX+**d**)	1	6	`52, DD, 17, dd`
LD.L	DE,(IX+**d**)	0	7	`49, DD, 17, dd`
LD	HL,(IX+**d**)	0/1	5/6	`DD, 27, dd`
LD.S	HL,(IX+**d**)	1	6	`52, DD, 27, dd`
LD.L	HL,(IX+**d**)	0	7	`49, DD, 27, dd`
LD	BC,(IY+**d**)	0/1	5/6	`FD, 07, dd`
LD.S	BC,(IY+**d**)	1	6	`52, FD, 07, dd`
LD.L	BC,(IY+**d**)	0	7	`49, FD, 07, dd`

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	DE,(IY+**d**)	0/1	4/5	`FD, 17, dd`
LD.S	DE,(IY+**d**)	1	5	`52, FD, 17, dd`
LD.L	DE,(IY+**d**)	0	6	`49, FD, 17, dd`
LD	HL,(IY+**d**)	0/1	4/5	`FD, 27, dd`
LD.S	HL,(IY+**d**)	1	5	`52, FD, 27, dd`
LD.L	HL,(IY+**d**)	0	6	`49, FD, 27, dd`

LD rr, Mmn

Load Register

Operation

rr ← Mmn

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The immediate operand, **Mmn**, is written to the multibyte **rr** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**ss,mn**	0	3	`kk, nn, mm`
LD	**ss,Mmn**	1	4	`kk, nn, mm, MM`
LD.LIL	**ss,Mmn**	0	5	`5B, kk, nn, mm, MM`
LD.SIS	**ss,mn**	1	4	`40, kk, nn, mm`

`kk` identifies the BC, DE, HL, or SPI register and is assembled into one of the opcodes indicated in Table 71.

Table 71. Register and `kk` Opcodes for LD rr, Mmn Instruction (hex)

Register	kk
BC	`01`
DE	`11`
HL	`21`

LD rr, (Mmn)

Load Register

Operation

rr ← (Mmn)

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The 16- or 24-bit operand **(Mmn)** specifies a location in memory. The 16- or 24-bit value stored at this location in memory is written to the multibyte **rr** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**rr,(mn)**	0	6	`ED, kk, nn, mm`
LD	**rr,(Mmn)**	1	8	`ED, kk, nn, mm, MM`
LD.LIL	**rr,(Mmn)**	0	9	`5B, ED, kk, nn, mm, MM`
LD.SIS	**rr,(mn)**	1	7	`40, ED, kk, nn, mm`
LD	HL, (**mn**)	0	5	`2A, nn, mm`
LD	HL, (M**mn**)	1	7	`2A, nn, mm, MM`
LD.LIL	HL, (**Mmn**)	0	8	`5B, 2A, nn, mm, MM`
LD.SIS	HL, (**mn**)	1	6	`40, 2A, nn, mm`

`kk` identifies the BC or DE register and is assembled into one of the opcodes indicated in Table 72.

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

Table 72. Register and `kk` Opcodes for LD rr, (Mmn) Instruction (hex)

Register	kk
BC	4B
DE	5B

LD (rr), A
Load Indirect

Operation

(rr) ← A

Description

The **rr** operand is any of the multibyte registers BC, DE, or HL. The CPU stores the contents of the accumulator, A, in the memory location specified by the contents of the multibyte register **rr**.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	(BC),A	X	2	02
LD.S	(BC),A	1	3	52, 02
LD.L	(BC),A	0	3	49, 02
LD	(DE),A	X	2	12
LD.S	(DE),A	1	3	52, 12
LD.L	(DE),A	0	3	49, 12
LD	(HL),A	X	2	77
LD.S	(HL),A	1	3	52, 77
LD.L	(HL),A	0	3	49, 77

LD SP, HL

Load Stack Pointer

Operation

SP ← HL

Description

The CPU writes the contents of the multibyte CPU register HL to the Stack Pointer. In ADL mode, or when the **.L** suffix is employed, the destination is Stack Pointer Long (SPL). In Z80 mode, or when the **.S** suffix is employed, the destination is Stack Pointer Short (SPS).

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**SP**,HL	x	1	`F9`
LD.S	**SP**,HL	1	2	`52, F9`
LD.L	**SP**,HL	0	2	`49, F9`

LD SP, IX/Y

Load Stack Pointer

Operation

SP ← IX/Y

Description

The CPU writes the contents of the specified multibyte Index Register, IX or IY, to the Stack Pointer. In ADL mode, or when the **.L** suffix is employed, the destination is Stack Pointer Long (SPL). In Z80 mode, or when the **.S** suffix is employed, the destination is Stack Pointer Short (SPS).

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**SP**,IX	X	2	`DD, F9`
LD.S	**SP**,IX	1	3	`52, DD, F9`
LD.L	**SP**,IX	0	3	`49, DD, F9`
LD	**SP**,IY	X	2	`FD, F9`
LD.S	**SP**,IY	1	3	`52, FD, F9`
LD.L	**SP**,IY	0	3	`49, FD, F9`

LD SP, Mmn

Load Stack Pointer

Operation

SP ← Mmn

Description

The immediate operand, **Mmn**, is written to the multibyte Stack Pointer register (**SP**). In ADL mode, or when the **.L** suffix is employed, Stack Pointer Long (SPL) is the destination. In Z80 mode, or when the **.S** suffix is employed, Stack Pointer Short (SPS) is the destination.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**SP,mn**	0	3	`31, nn, mm`
LD	**SP,Mmn**	1	4	`31, nn, mm, MM`
LD.LIL	**SP,Mmn**	0	5	`5B, 31, nn, mm, MM`
LD.SIS	**SP,mn**	1	4	`40, 31, nn, mm`

LD SP, (Mmn)

Load Stack Pointer

Operation

SP ← (Mmn)

Description

The 16- or 24-bit operand **(Mmn)** specifies a location in memory. The 16- or 24-bit value stored at this memory location is written to the multibyte Stack Pointer register (**SP**). In ADL mode, or when the **.L** suffix is employed, Stack Pointer Long (SPL) is the destination. In Z80 mode, or when the **.S** suffix is employed, Stack Pointer Short (SPS) is the destination.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LD	**SP,(mn)**	0	5	`ED, 7B, nn, mm`
LD	**SP,(Mmn)**	1	6	`ED, 7B, nn, mm, MM`
LD.LIL	**SP,(Mmn)**	0	7	`5B, ED, 7B, nn, mm, MM`
LD.SIS	**SP,(mn)**	1	6	`40, ED, 7B, nn, mm`

Note: ZiLOG recommends against using the **.SIL** and **.LIS** suffixes with this instruction. The **.SIL** instruction fetches a 24-bit value, **Mmn**. However, this instruction ignores the upper byte and uses address {MBASE, mm, nn} instead. The **.LIS** instruction fetches a 16-bit value, **mn**. However, the **.LIS** instruction does not use the MBASE value. Instead, it uses address {00, mm, nn}.

LDD

Load and Decrement

Operation

(DE) ← (HL)
BC ← BC – 1
DE ← DE – 1
HL ← HL – 1

Description

The CPU writes the contents of the memory location with an address contained in the multibyte register HL to the memory location with the address contained in the multibyte register DE. The BC, DE, and HL registers decrement.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Reset if BC – 1 = 0; set otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LDD	—	X	5	`ED, A8`
LDD.S	—	1	6	`52, ED, A8`
LDD.L	—	0	6	`49, ED, A8`

LDDR

Load and Decrement with Repeat

Operation

repeat {
  (DE) ← (HL)
  BC ← BC−1
  DE ← DE−1
  HL ← HL−1
} while (BC ≠ 0)

Description

The CPU writes the contents of the memory location with address contained in the multibyte register HL to the memory location with address contained in the multibyte register DE. The BC, DE, and HL registers decrement. This operation is repeated until BC decrements to 0.

In Z80 mode, the BC register is 16 bits, which allows the LDDR instruction to repeat a maximum of 65536 (64K) times. In ADL mode, the BC register is 24 bits, which allows the LDDR instruction to repeat a maximum of 16,777,216 (16M) times.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Reset if BC−1=0; set otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LDDR	—	X	2 + 3 * BC	`ED, B8`
LDDR.S	—	1	3 + 3 * BC	`52, ED, B8`
LDDR.L	—	0	3 + 3 * BC	`49, ED, B8`

LDI

Load and Increment

Operation

(DE) ← (HL)
BC ← BC−1
DE ← DE+1
HL ← HL+1

Description

The CPU writes the contents of the memory location with address contained in the multibyte register HL to the memory location with address contained in the multibyte register DE. The BC register decrements. The DE and HL registers increment.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Reset if BC−1=0; set otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LDI	—	X	5	`ED, A0`
LDI.S	—	1	6	`52, ED, A0`
LDI.L	—	0	6	`49, ED, A0`

LDIR

Load and Increment with Repeat

Operation

repeat {
(DE) ← (HL)
BC ← BC−1
DE ← DE+1
HL ← HL+1
} while (BC ≠ 0)

Description

The CPU writes the contents of the memory location with the address contained in the multibyte register HL to the memory location with the address contained in the multibyte register DE. The BC register decrements, and the DE and HL registers increment. This operation is repeated until BC decrements to 0.

In Z80 mode, the BC register is 16 bits, which allows the CPDR instruction a maximum of 65536 (64K) times. In ADL mode, the BC register is 24 bits, which allows the CPDR instruction to repeat a maximum of 16,777,216 (16M) times.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Reset if BC−1=0; set otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LDIR	—	X	2 + 3 * BC	ED, B0
LDIR.S	—	1	3 + 3 * BC	52, ED, B0
LDIR.L	—	0	3 + 3 * BC	49, ED, B0

LEA IX/Y, IX+d

Load Effective Address

Operation

IX/Y ← IX+**d**

Description

The CPU adds the contents of the IX register to the signed displacement **d** and writes the sum to the specified multibyte Index Register, IX or IY.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LEA	IX,IX+**d**	X	3	`ED, 32, dd`
LEA.S	IX,IX+**d**	1	4	`52, ED, 32, dd`
LEA.L	IX,IX+**d**	0	4	`49, ED, 32, dd`
LEA	IY,IX+**d**	X	3	`ED, 55, dd`
LEA.S	IY,IX+**d**	1	4	`52, ED, 55, dd`
LEA.L	IY,IX+**d**	0	4	`49, ED, 55, dd`

LEA IX/Y, IY+d

Load Effective Address

Operation

IX/Y ← IY+**d**

Description

The CPU adds the contents of the IY register to the two's-complement displacement **d** and writes the sum to the specified multibyte Index Register, IX or IY.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LEA	IX,IY+**d**	X	3	`ED, 54, dd`
LEA.S	IX,IY+**d**	1	4	`52, ED, 54, dd`
LEA.L	IX,IY+**d**	0	4	`49, ED, 54, dd`
LEA	IY,IY+**d**	X	3	`ED, 33, dd`
LEA.S	IY,IY+**d**	1	4	`52, ED, 33, dd`
LEA.L	IY,IY+**d**	0	4	`49, ED, 33, dd`

LEA rr, IX+d

Load Effective Address

Operation

rr ← IX+**d**

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The CPU adds the contents of the IX register to the signed displacement **d** and writes the sum to the multibyte **rr** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LEA	**rr**,IX+**d**	X	3	`ED, kk, dd`
LEA.S	**rr**,IX+**d**	1	4	`52, ED, kk, dd`
LEA.L	**rr**,IX+**d**	0	4	`49, ED, kk, dd`

`kk` identifies either the BC, DE, or HL multibyte register and is assembled into one of the opcodes indicated in Table 73.

Table 73. Register and `kk` Opcodes for LEA rr, IX+d Instruction (hex)

Register	`kk`
BC	`02`
DE	`12`
HL	`22`

LEA rr, IY+d
Load Effective Address

Operation

rr ← IY+**d**

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The CPU adds the contents of the IY register to the signed displacement **d** and writes the sum to the multibyte **rr** register.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
LEA	**rr**,IY+**d**	X	3	`ED, kk, dd`
LEA.S	**rr**,IY+**d**	1	4	`52, ED, kk, dd`
LEA.L	**rr**,IY+**d**	0	4	`49, ED, kk, dd`

`kk` identifies either the BC, DE, or HL multibyte register and is assembled into one of the opcodes indicated in Table 74.

Table 74. Register and `kk` Opcodes for LEA rr, IY+d Instruction (hex)

Register	`kk`
BC	`03`
DE	`13`
HL	`23`

MLT rr

Multiply Register

Operation

rr[15:0] ← **rr**[15:8] x **rr**[7:0]

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The **MLT** instruction performs an 8-bit by 8-bit multiply operation. The **rr** operand Low byte is multiplied by the **rr** operand High byte. The 16-bit product is written back into the 16-bit **rr** register pair. The MLT instruction performs an 8-bit by 8-bit multiply operation with a 16-bit result, regardless of the ADL mode.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
MLT	BC	X	6	`ED 4C`
MLT	DE	X	6	`ED 5C`
MLT	HL	X	6	`ED 6C`

MLT SP

Multiply Stack Pointer

Operation

SP[15:0] ← **SP**[15:8] x **SP**[7:0]

Description

The **MLT SP** instruction performs an 8-bit by 8-bit multiply operation using the Stack Pointer (**SP**). The **SP** Low byte is multiplied by the **SP** High byte. The 16-bit product is written back into the **SP** register. This operation is an 8-bit by 8-bit operation with a 16-bit result, regardless of the ADL mode. In ADL mode, or if the **.L** suffix is employed, the 24-bit Stack Pointer Long (SPL) is used. In Z80 mode, or if the **.S** suffix is employed, the 16-bit Stack Pointer Short (SPS) is used.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
MLT	**SP**	X	6	`ED 7C`
MLT.L	**SP**	0	6	`49 ED 7C`
MLT.S	**SP**	1	6	`52 ED 7C`

NEG

Negate Accumulator

Operation

A ← 0–A

Description

The contents of the accumulator, A, are negated (two's-complemented) and are identical to a subtraction of the accumulator from 0.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; result otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if accumulator contained `80h` before operation; reset otherwise.
N	Set.
C	Set if accumulator was not `00h` before operation; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
NEG		X	2	`EE, 44`

NOP
No Operation

Operation

No operation.

Description

The CPU performs no operation during execution of this instruction.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
NOP		X	1	00

OR A, (HL)
Logical OR

Operation

A ← A **OR** (HL)

Description

The (HL) operand is the 8-bit value located at the memory location specified by the contents of the multibyte CPU register HL. This 8-bit value is logically ORed to the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OR	A, (HL)	X	2	`B6`
OR.S	A, (HL)	1	3	`52, B6`
OR.L	A, (HL)	0	3	`49, B6`

OR A, ir
Logical OR

Operation

A ← A **OR ir**

Description

The **rr** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The **rr** operand is logically ORed to the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OR	A, IXH	X	2	`DD B4`
OR	A, IXL	X	2	`DD B5`
OR	A, IYH	X	2	`FD B4`
OR	A, IYL	X	2	`FD B5`

OR A, (IX/Y+d)

Logical OR

Operation

A ← A **OR** (**IX/Y**+**d**)

Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value is logically ORed to the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OR	A, (IX+**d**)	**X**	**4**	`DD, B6, dd`
OR.S	A, (IX+**d**)	**1**	**5**	`52, DD, B6, dd`
OR.L	A, (IX+**d**)	**0**	**5**	`49, DD, B6, dd`
OR	A, (IY+**d**)	**X**	**4**	`FD, B6, dd`
OR.S	A, (IY+**d**)	**1**	**5**	`52, FD, B6, dd`
OR.L	A, (IY+**d**)	**0**	**5**	`49, FD, B6, dd`

OR A, n
Logical OR

Operation

A ← A **OR n**

Description

The 8-bit immediate value **n** is logically ORed to the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OR	A, **n**	X	2	`F6, nn`

OR A, r
Logical OR

Operation

A ← A **OR r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand is logically ORed to the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OR	A, **r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 75.

Table 75. Register and `jj` Opcodes for OR A, r Instruction (hex)

Register	`jj`
A	B7
B	B0
C	B1
D	B2
E	B3
H	B4
L	B5

OTD2R

Output to I/O and Decrement with Repeat

Operation

repeat {
    ({UU, DE[15:0]}) ← (HL)
    BC ← BC–1
    DE ← DE–1
    HL ← HL–1
} while BC ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. This byte is output to I/O address {UU, DE[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The BC, DE, and HL registers are decremented. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if BC–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTD2R	—	X	2 + 3 * B	`ED, BC`
OTD2R.S	—	1	3 + 3 * B	`52, ED, BC`
OTD2R.L	—	0	3 + 3 * B	`49, ED, BC`

Note

This instruction operates differently in the eZ80190 device. In the eZ80190, operation is:

repeat {
  ({UU, BC[15:0]}) ← (HL)
  B ← B−1
  C ← C−1
  HL ← HL−1
} while B ≠ 0

OTDM

Output to I/O and Decrement

Operation

({UU, `00h`,C}) ← (HL)
B ← B–1
C ← C–1
HL ← HL–1

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to the I/O address specified by the C register with the High byte of the address, ADDR[15:8], forced to 0. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B, C, and HL registers are decremented.

Condition Bits Affected

S	Undefined.
Z	Set if B–1=0; reset otherwise.
H	Undefined.
P/V	Undefined.
N	Set if msb of data is logical 1; reset otherwise.
C	Undefined.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTDM	—	X	5	`ED, 8B`
OTDM.S	—	1	6	`52, ED, 8B`
OTDM.L	—	0	6	`49, ED, 8B`

OTDMR

Output to I/O and Decrement

Operation

repeat {
  ({UU, `00h`, C}) ← (HL)
  B ← B–1
  C ← C–1
  HL ← HL–1
} while B ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to the I/O address specified by the C register with the High byte of the address, ADDR[15:8], forced to 0. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B, C, and HL registers are decremented. The instruction repeats until register B equals 0.

Condition Bits Affected

S	Undefined.
Z	Set if B–1=0; reset otherwise.
H	Undefined.
P/V	Undefined.
N	Set if msb of data is logical 1; reset otherwise.
C	Undefined.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTDMR	—	X	2 + 3 * B	`ED, 9B`
OTDMR.S	—	1	3 + 3 * B	`52, ED, 9B`
OTDMR.L	—	0	3 + 3 * B	`49, ED, 9B`

OTDR

Output to I/O and Decrement

Operation

repeat {
 ({UU, BC[15:0]}) ← (HL)
 B ← B–1
 HL ← HL–1
} while B ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B and HL registers are decremented. The instruction repeats until register B equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTDR	—	X	2 + 3 * B	`ED, BB`
OTDR.S	—	1	3 + 3 * B	`52, ED, BB`
OTDR.L	—	0	3 + 3 * B	`49, ED, BB`

OTDRX

Output to I/O and Decrement Memory Address with Stationary I/O Address

Operation

repeat {
  {UU, DE[15:0]} ← (HL)
  BC ← BC−1
  HL ← HL−1
} while BC ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to the I/O address {UU, DE[15:0]}. The upper byte of I/O addresses is undefined. The BC and HL registers decrement. The Z Flag is set to 1 if the BC register decrements to 0. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set of BC−1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTDRX	—	X	2 + 3 * BC	`ED, CB`
OTDRX.S	—	1	3 + 3 * BC	`52, ED, CB`
OTDRX.L	—	0	3 + 3 * BC	`49, ED, CB`

Note

This instruction is not supported on the eZ80190 device.

OTI2R

Output to I/O and Increment with Repeat

Operation

repeat {
  ({UU, DE[15:0]}) ← (HL)
  BC ← BC−1
  DE ← DE+1
  HL ← HL+1
} while BC ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, DE[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The BC register decrements. The DE and HL registers increment. The instruction repeats until register BC equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if BC−1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTI2R	—	X	2 + 3 * B	`ED, B4`
OTI2R.S	—	1	3 + 3 * B	`52, ED, B4`
OTI2R.L	—	0	3 + 3 * B	`49, ED, B4`

Note

This instruction operates differently in the eZ80190 device product. In the eZ80190, operation is:

repeat {
  ({UU, BC[15:0]}) ← (HL)
  B ← B–1
  C ← C+1
  HL ← HL+1
} while B ≠ 0

OTIM

Output to I/O and Increment

Operation

({UU, `00h`,C}) ← (HL)
B ← B–1
C ← C+1
HL ← HL+1

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to the I/O address specified by the C register with the High byte of the address, ADDR[15:8], forced to 0. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B register decrements. The C and HL registers increment.

Condition Bits Affected

S Undefined.
Z Set if B–1=0; reset otherwise.
H Undefined.
P/V Undefined.
N Set if msb of data is logical 1; reset otherwise.
C Undefined.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTIM	—	X	5	`ED, 83`
OTIM.S	—	1	6	`52, ED, 83`
OTIM.L	—	0	6	`49, ED, 83`

OTIMR

Output to I/O and Increment

Operation

repeat {
  ({UU, `00h`, C}) ← (HL)
  B ← B−1
  C ← C+1
  HL ← HL+1
} while B ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to the I/O address specified by the C register with the High byte of the address, ADDR[15:8], forced to 0. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B register decrements. The C and HL registers increment. The instruction repeats until the B register equals 0.

Condition Bits Affected

S	Undefined.
Z	Set if B−1=0; reset otherwise.
H	Undefined.
P/V	Undefined.
N	Set if msb of data is logical 1; reset otherwise.
C	Undefined.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTIMR	—	X	2 + 3 * B	`ED, 93`
OTIMR.S	—	1	3 + 3 * B	`52, ED, 93`
OTIMR.L	—	0	3 + 3 * B	`49, ED, 93`

OTIR

Output to I/O and Increment

Operation

repeat {
    ({UU, BC[15:0]}) ← (HL)
    B ← B−1
    HL ← HL+1
} while B ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B register decrements and the HL register increments. The instruction repeats until the B register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set if B−1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTIR	—	X	2 + 3 * B	`ED, B3`
OTIR.S	—	1	3 + 3 * B	`52, ED, B3`
OTIR.L	—	0	3 + 3 * B	`49, ED, B3`

OTIRX

Output to I/O and Increment Memory Address with Stationary I/O Address

Operation

repeat {
  {UU, DE[15:0]} ← (HL)
  BC ← BC−1
  HL ← HL+1
} while BC ≠ 0

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next loads the contents of this byte to the I/O address {UU, DE[15:0]}. The upper byte of I/O addresses is undefined. The BC register decrements. The HL register increments. The Z Flag is set to 1 if the BC register decrements to 0. The instruction repeats until the BC register equals 0.

Condition Bits Affected

S	Not affected.
Z	Set of BC−1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OTIRX	—	X	2 + 3 * BC	`ED, C3`
OTIRX.S	—	1	3 + 3 * BC	`52, ED, C3`
OTIRX.L	—	0	3 + 3 * BC	`49, ED, C3`

Note

This instruction is not supported on the eZ80190 device.

OUT (BC), r—also OUT (C), r for Z80 compatibility

Output to I/O

Operation

({UU, BC[15:0]}) ← **r**

Description

The **r** operand is any of the A, B, C, D, E, H, and L registers. The CPU outputs the contents of this byte of the specified register to the I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUT	(BC),**r**	X	3	`ED, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 76.

Table 76. Register and `jj` Opcodes for OUT (BC), r and OUT (C), r Instructions (hex)

Register	`jj`
A	`79`
B	`41`
C	`49`

Table 76. Register and `jj` Opcodes for OUT (BC), r and OUT (C), r Instructions (hex) (Continued)

Register	`jj`
D	51
E	59
H	61
L	69

OUT (n), A

Output to I/O

Operation

({UU, A, **n**}) ← A

Description

The **n** operand is placed on the lower byte of the address bus, ADDR[7:0]. The CPU places the contents of the accumulator, A, onto the middle byte of the address bus, ADDR[15:8]. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU next outputs the contents of the accumulator to this I/O address.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUT	(**n**),A	X	3	`D3, nn`

OUT0 (n), r

Output to I/O

Operation

({UU, 00h, **n**}) ← **r**

Description

The **r** operand is any of A, B, C, D, E, H, or L. The **n** operand is placed on the lower byte of the address bus, ADDR[7:0], while the High byte of the address bus, ADDR[15:8], is forced to 0. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The CPU next outputs the contents of the **r** register to the I/O address {UU, 00h, **n**}.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUT0	(**n**),**r**	X	4	ED, jj, nn

jj identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 77.

Table 77. Register and jj Opcodes for OUT0 (n), r Instruction (hex)

Register	jj
A	39
B	01
C	09

Table 77. Register and `jj` Opcodes for OUT0 (n), r Instruction (hex)

Register	`jj`
D	`11`
E	`19`
H	`21`
L	`29`

OUTD

Output to I/O and Decrement

Operation

({UU, BC[15:0]}) ← (HL)
B ← B–1
HL ← HL–1

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B and HL registers decrement.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUTD	—	X	5	`ED, AB`
OUTD.S	—	1	6	`52, ED, AB`
OUTD.L	—	0	6	`49, ED, AB`

OUTD2
Output to I/O and Decrement

Operation

({UU, BC[15:0]}) ← (HL)
B ← B–1
C ← C–1
HL ← HL–1

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B, C, and HL registers decrement.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUTD2	—	X	5	`ED, AC`
OUTD2.S	—	1	6	`52, ED, AC`
OUTD2.L	—	0	6	`49, ED, AC`

OUTI

Output to I/O and Increment

Operation

({UU, BC[15:0]}) ← (HL)
B ← B–1
HL ← HL+1

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B register decrements, and the HL register increments.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUTI	—	X	5	`ED, A3`
OUTI.S	—	1	6	`52, ED, A3`
OUTI.L	—	0	6	`49, ED, A3`

OUTI2
Output to I/O and Increment

Operation

({UU, BC[15:0]}) ← (HL)
B ← B–1
C ← C+1
HL ← HL+1

Description

The CPU loads the contents of the memory location specified by the multibyte HL register into CPU memory. The CPU next outputs this byte to I/O address {UU, BC[15:0]}. The upper byte of the address bus, ADDR[23:16] is undefined for I/O addresses. The B register decrements. The C and HL registers increment.

Condition Bits Affected

S	Not affected.
Z	Set if B–1=0; reset otherwise.
H	Not affected.
P/V	Not affected.
N	Set if msb of data is logical 1; reset otherwise.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
OUTI2	—	X	5	`ED, A4`
OUTI2.S	—	1	6	`52, ED, A4`
OUTI2.L	—	0	6	`49, ED, A4`

PEA IX+d

Push Effective Address

Operation

if ADL mode{
(SPL–1) ← IXd[23:16]
(SPL–2) ← IXd[15:8]
(SPL–3) ← IXd[7:0]
SPL ← SPL–3
}
else Z80 mode {
(SPS–1) ← IXd[15:8]
(SPS–2) ← IXd[7:0]
SPS ← SPS–2
}
where IXd indicates the sum of the contents of the register IX and the two's-complement displacement **d**.

Description

In ADL mode, the 24-bit sum of the contents of IX and the two's-complement displacement **d** is pushed onto the stack at SPL. The stack pointer, SPL, decrements by 3.

In Z80 mode, the 16-bit sum of the contents of IX and the two's-complement displacement **d** is pushed onto the stack at SPS. The stack pointer, SPS, decrements by 2.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
PEA	IX+**d**	0/1	5/6	`ED, 65, dd`
PEA.S	IX+**d**	1	6	`52, ED, 65, dd`
PEA.L	IX+**d**	0	7	`49, ED, 65, dd`

PEA IY+d

Push Effective Address

Operation

if ADL mode{
(SPL–1) ← IYd[23:16]
(SPL–2) ← IYd[15:8]
(SPL–3) ← IYd[7:0]
SPL ← SPL–3
}
else Z80 mode {
(SPS–1) ← IYd[15:8]
(SPS–2) ← IYd[7:0]
SPS ← SPS–2
}
where IYd indicates the sum of the contents of the register IY and the two's-complement displacement **d**.

Description

In ADL mode, the 24-bit sum of the contents of IY and the two's-complement displacement **d** is pushed onto the stack at SPL. The stack pointer, SPL, decrements by 3. The most significant byte (MSB) is pushed onto the stack first.

In Z80 mode, the 16-bit sum of the contents of IY and the two's-complement displacement **d** is pushed onto the stack at SPS. The stack pointer, SPS, decrements by 2. The most significant byte (MSB) is pushed onto the stack first.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
PEA	IY+**d**	0/1	5/6	`ED, 66, dd`
PEA.S	IY+**d**	1	6	`52, ED, 66, dd`
PEA.L	IY+**d**	0	7	`49, ED, 66, dd`

POP AF
Pop Stack

Operation

```
if ADL mode {
    F ← (SPL)
    A ← (SPL+1)
    Discard ← (SPL+2)
    SPL ← SPL+3
}
else Z80 mode {
    F ← (SPS)
    A ← (SPS+1)
    SPS ← SPS+2
}
```

Description

In ADL mode, or when the **.L** suffix is employed, 3 bytes are popped off the stack beginning at the memory location specified by SPL. The first byte popped off the stack from SPL is written to the Flags Register, F. The second byte popped off the stack from (SPL+1) is written to the accumulator, A. The third byte popped off the stack from (SPL+2) is discarded. The SPL increments by 3.

In Z80 mode, or when the **.S** suffix is employed, 2 bytes are popped off the stack beginning at the memory location specified by SPS. The first byte popped off the stack from SPS is written to the Flags Register, F. The second byte popped off the stack from (SPS+1) is written to the accumulator, A. The SPS increments by 2.

Condition Bits Affected

The condition bits are written with the Flags register (F) value popped from the stack.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
POP	AF	0/1	3/4	`F1`
POP.S	AF	1	4	`52, F1`
POP.L	AF	0	5	`49, F1`

POP IX/Y

Pop Stack

Operation

```
if ADL mode {
    IX/Y[7:0] ← (SPL)
    IX/Y[15:8] ← (SPL+1)
    IX/Y[23:16] ← (SPL+2)
    SPL ← SPL+3
}
else Z80 mode {
    IX/Y[7:0] ← (SPS)
    IX/Y[15:8] ← (SPS+1)
    SPS ← SPS+2
}
```

Description

In ADL mode, or when the **.L** suffix is employed, 3 bytes are popped off the stack beginning at the memory location specified by SPL. The first byte popped off the stack from SPL is written to the Low byte of the specified Index Register, IXL or IYL. The second byte popped off the stack from (SPL+1) is written to the High byte of the specified Index Register, IXH or IYH. The third byte popped off the stack from (SPL+2) is written to the upper byte of the specified Index Register, IXU or IYU. The SPL increments by 3.

In Z80 mode, or when the **.S** suffix is employed, the first 2 bytes are popped off the stack beginning at the memory location specified by SPS. The first byte popped off the stack from (SPS+1) is written to the Low byte of the specified Index Register, IXL or IYL. The second byte popped off the stack from (SPS+2) is written to the High byte of the specified Index Register, IXH or IYH. The SPS increments by 2.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
POP	IX	0/1	4/5	`DD, E1`
POP.S	IX	1	5	`52, DD, E1`
POP.L	IX	0	6	`49, DD, E1`
POP	IY	0/1	4/5	`FD, E1`
POP.S	IY	1	5	`52, FD, E1`
POP.L	IY	0	6	`49, FD, E1`

POP rr

Pop Stack

Operation

if ADL mode {
  rr[7:0]← (SPL)
  rr[15:8]← (SPL+1)
  rr[23:16]← (SPL+2)
  SPL ← SPL+3
}
else Z80 mode {
  rr[7:0]← (SPS)
  rr[15:8]← (SPS+1)
  SPS ← SPS+2
}

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL.

In ADL mode, or when the **.L** suffix is employed, 3 bytes are popped off the stack beginning at the memory location specified by SPL. The first byte popped off the stack from SPL is written to the Low byte of the specified register, **rr**[7:0]. The second byte popped off the stack from (SPL+1) is written to the High byte of the specified register, **rr**[15:8]. The third byte popped off the stack from (SPL+2) is written to the upper byte of the specified register, **rr**[23:16]. The SPL increments by 3.

In Z80 mode, or when the **.S** suffix is employed, the first 2 bytes are popped off the stack beginning at the memory location specified by SPS. The first byte popped off the stack from (SPS+1) is written to the Low byte of the specified register, **rr**[7:0]. The second byte popped off the stack from (SPS+2) is written to the High byte of the specified register, **rr**[15:8]. The SPS increments by 2.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
POP	**rr**	0/1	3/4	kk
POP.S	**rr**	1	4	52, kk
POP.L	**rr**	0	5	49, kk

kk identifies either the BC, DE, or HL multibyte register and is assembled into one of the opcodes indicated in Table 78.

Table 78. Register and kk Opcodes for POP rr Instruction (hex)

Register	kk
BC	C1
DE	D1
HL	E1

PUSH AF

Push Stack

Operation

```
if ADL mode {
    (SPL–1) ← 00h
    (SPL–2) ← A
    (SPL–3) ← F
    SPL ← SPL–3
}
else Z80 mode {
    (SPS–1) ← A
    (SPS–2) ← F
    SPS ← SPS–2
}
```

Description

In ADL mode, or when the **.L** suffix is employed, 3 bytes are pushed onto the memory locations indicated by SPL, in the following sequence:

1. A value of `00h` is written to the memory location with address SPL–1.
2. The CPU writes the contents of the accumulator, A, to the memory location with address SPL–2.
3. The CPU next writes the contents of the Flags Register, F, to the memory location with address SPL–3.

SPL decrements by three.

In Z80 mode, or when the **.S** suffix is employed, 2 bytes are pushed onto the memory locations indicated by SPS, in the following sequence:

1. The CPU writes the contents of the accumulator, A, to the memory location with address SPS–1.

2. The CPU next writes the contents of the Flags Register, F, to the memory location with address SPS–2.

SPS decrements by two.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
PUSH	AF	0/1	3/4	`F5`
PUSH.S	AF	1	4	`52, F5`
PUSH.L	AF	0	5	`49, F5`

PUSH IX/Y

Push Stack

Operation

if ADL mode {
    (SPL−1) ← **IX/Y**[23:16]
    (SPL−2) ← **IX/Y**[15:8]
    (SPL−3) ← **IX/Y**[7:0]
    SPL ← SPL−3
}
else Z80 mode {
    (SPS−1) ← **IX/Y**[15:8]
    (SPS−2) ← **IX/Y**[7:0]
    SPS ← SPS−2
}

Description

In ADL mode, or when the **.L** suffix is employed, 3 bytes are pushed onto the memory locations indicated by SPL, in the following sequence:

1. The CPU writes the contents of the upper byte of the specified Index Register, IXU or IYU, to the memory location with address SPL−1.
2. The CPU next writes the contents of the High byte of the specified Index Register, IXH or IYH, to the memory location with address SPL−2.
3. The CPU next writes the contents of the Low byte of the specified Index Register, IXL or IYL, to the memory location with address SPL−3.

SPL decrements by three.

In Z80 mode, or when the **.S** suffix is employed, 2 bytes are pushed onto the memory locations indicated by SPS, in the following sequence:

1. The CPU writes the contents of the High byte of the specified Index Register, IXH or IYH, to the memory location with address SPS–1.
2. The CPU next writes the contents of the Low byte of the specified Index Register, IXL or IYL, to the memory location with address SPS–2.

SPS decrements by two.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
PUSH	IX	0/1	4/5	`DD, E5`
PUSH.S	IX	1	5	`52, DD, E5`
PUSH.L	IX	0	6	`49, DD, E5`
PUSH	IY	0/1	4/5	`FD, E5`
PUSH.S	IY	1	5	`52, FD, E5`
PUSH.L	IY	0	6	`49, FD, E5`

PUSH rr
Push Stack

Operation

```
if ADL mode {
    (SPL–1) ← rr[23:16]
    (SPL–2) ← rr[15:8]
    (SPL–3) ← rr[7:0]
    SPL ← SPL–3
}
else Z80 mode {
    (SPS–1) ← rr[15:8]
    (SPS–2) ← rr[7:0]
    SPS ← SPS–2
}
```

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. In ADL mode, or when the **.L** suffix is employed, 3 bytes are pushed onto the memory locations indicated by SPL, in the following sequence:

1. The CPU writes the contents of the upper byte of the specified register, **rr**[23:16], to the memory location with address SPL–1.
2. The CPU next writes the contents of the High byte of the specified register, **rr**[15:8], to the memory location with address SPL–2.
3. The CPU next writes the contents of the Low byte of the specified register, **rr**[7:0], to the memory location with address SPL–3.

SPL decrements by three.

In Z80 mode, or when the **.S** suffix is employed, 2 bytes are pushed onto the memory locations indicated by SPS, in the following sequence:

1. The CPU writes the contents of the High byte of the specified register, **rr**[15:8], to the memory location with address SPS–1.

2. The CPU next writes the contents of the Low byte of the specified register, **rr**[7:0], to the memory location with address SPS–2.

SPS decrements by two.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
PUSH	**rr**	0/1	3/4	`kk`
PUSH.S	**rr**	1	4	`52, kk`
PUSH.L	**rr**	0	5	`49, kk`

`kk` identifies either the BC, DE, or HL multibyte register and is assembled into one of the opcodes indicated in Table 79.

Table 79. Register and `kk` Opcodes for PUSH rr Instruction (hex)

Register	`kk`
BC	`C5`
DE	`D5`
HL	`E5`

RES b, (HL)
Reset Bit

Operation

(HL)[**b**] ← 0

Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). Bit **b** of this value is reset to 0.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RES	**b**,(HL)	X	3	`CB, kk`
RES.S	**b**,(HL)	1	4	`52, CB, kk`
RES.L	**b**,(HL)	0	4	`49, CB, kk`

`kk`=binary code `10 bbb 110`; where `bbb` identifies the bit tested and is assembled into the object code, as indicated in Table 80.

Table 80. `bbb` Opcodes for RES b, (HL) Instruction (hex)

Bit Tested	`bbb`
0	`000`
1	`001`
2	`010`
3	`011`

Table 80. `bbb` Opcodes for RES b, (HL) Instruction (hex) (Continued)

Bit Tested	**`bbb`**
4	`100`
5	`101`
6	`110`
7	`111`

RES b, (IX/Y+d)

Reset Bit

Operation

(**IX**/**Y**+**d**)[**b**] ← 0

Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. Bit **b** of this value is reset to 0.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RES	**b**,(IX+**d**)	X	5	`DD, CB, dd, kk`
RES.S	**b**,(IX+**d**)	1	6	`52, DD, CB, dd, kk`
RES.L	**b**,(IX+**d**)	0	6	`49, DD, CB, dd, kk`
RES	**b**,(IY+**d**)	X	5	`FD, CB, dd, kk`
RES.S	**b**,(IY+**d**)	1	6	`52, FD, CB, dd, kk`
RES.L	**b**,(IY+**d**)	0	6	`49, FD, CB, dd, kk`

`kk`=binary code `10 bbb 110`; where `bbb` identifies the bit tested and is assembled into the object code as indicated in Table 81.

Table 81. `bbb` Opcodes for RES b, (IX/Y+d) Instruction (hex)

Bit Tested	**bbb**
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

RES b, r

Reset Bit

Operation

r[**b**] ← 0

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. Bit **b** of the specified register **r** is reset to 0.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RES	**b,r**	X	2	`CB, jj`

`jj`=binary code `10 bbb rrr`, and `kk`=binary code `10 bbb 110`; where `rrr` identifies the A, B, C, D, E, H, or L register and `bbb` identifies the bit tested and assembled into the object code, as indicated in Table 82.

Table 82. Register, `bbb`, and `rrr` Opcodes for RES b, r Instruction (hex)

Bit Tested	`bbb`	Register	`rrr`
0	`000`	A	`111`
1	`001`	B	`000`
2	`010`	C	`001`
3	`011`	D	`010`

Table 82. Register, `bbb`, and `rrr` Opcodes for RES b, r Instruction (hex)

Bit Tested	**bbb**	Register	**rrr**
4	`100`	E	`011`
5	`101`	H	`100`
6	`110`	L	`101`
7	`111`		

RET

Return from Subroutine

Operation

PC ← **(SP)**

Description

The **RET** instruction returns program control back to the point in the user's application code that had reached the current subroutine via a **CALL** instruction. The return address pops from the stack and is written to the Program Counter. The MADL control bit must be set to 1 to enable mixed-ADL mode code and interrupts. If the MADL is reset to 0, the suffixed instructions do not operate correctly. More detailed operation is provided in Table 83.

Table 83. RET Instruction Detail

ADL	Suffix	Operation
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]}.
1	None	The starting Program Counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0].

Table 83. RET Instruction Detail (Continued)

ADL	Suffix	Operation
0	.S	An invalid suffix. **RET.L** must be used in all mixed-memory mode applications.
1	.S	An invalid suffix. **RET.L** must be used in all mixed-memory mode applications.

0	. L	The starting Program Counter is {MBASE, PC[15:0]}. Pop a byte from SPL into ADL to set memory mode (`03h`=ADL, `02h`=Z80).
		if ADL mode {
		Pop the upper byte of the return address from SPL into PC[23:16].
		Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0].
		The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from {MBASE,SPS} into PC[15:0].
		The ending Program Counter is {MBASE, PC[15:0]}.
		}

1	.L	The starting Program Counter is PC[23:0]. Pop a byte from SPL into ADL to set memory mode (`03h`=ADL, `02h`=Z80).
		if ADL mode {
		Pop 3-byte return address from SPL into PC[23:0].
		The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from SPL into PC[15:0].
		The ending Program Counter is {MBASE, PC[15:0]}.
		}

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RET	—	0/1	5/6	C9
RET.L	—	0	6	49, C9
RET.L	—	1	7	5B, C9

RET cc

Conditional Return from Subroutine

Operation

if **cc** {
    PC ← **(SP)**
}

Description

If the condition is true (1), the **RET** instruction returns program control back to the point in the user's application code that had reached the current subroutine via a **CALL** instruction. The return address pops from the stack and is written to the Program Counter. The MADL control bit must be set to 1 to enable mixed-ADL mode code and interrupts. If the MADL is reset to 0, the suffixed instructions do not operate correctly. More detailed operation is provided in Table 84.

Table 84. RET cc Instruction Detail

ADL	Suffix	Operation (if cc is true)
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]}.
1	None	The starting Program Counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0].

Table 84. RET cc Instruction Detail (Continued)

0	.S	An invalid suffix. **RET.L cc** must be used in all mixed-memory mode applications.
1	.S	An invalid suffix. **RET.L cc** must be used in all mixed-memory mode applications.

0	. L	The starting Program Counter is {MBASE, PC[15:0]}. Pop a byte from SPL into ADL to set the new memory mode (`03h`=ADL, `02h`=Z80).
		if ADL mode {
		Pop the upper byte of the return address from SPL into PC[23:16].
		Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0].
		The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from {MBASE,SPS} into PC[15:0].
		The ending Program Counter is {MBASE, PC[15:0]}.
		}

1	.L	The starting Program Counter is PC[23:0]. Pop a byte from SPL into ADL to set the new memory mode (`03h`=ADL, `02h`=Z80).
		if ADL mode {
		Pop 3-byte return address from SPL into PC[23:0].
		The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from SPL into PC[15:0].
		The ending Program Counter is {MBASE, PC[15:0]}.
		}

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RET	**cc**	0/1	2 if **cc** false, 6/7 if **cc** true	`kk`
RET.L	**cc**	0	3 if **cc** false, 8 if **cc** true and return to Z80 Mode, 9 if **cc** true and return to ADL Mode	`49, kk`
RET.L	**cc**	1	3 if **cc** false, 8 if **cc** true and return to Z80 Mode, 9 if **cc** true and return to ADL Mode	`5B, kk`

The opcode (`kk`) depends on the condition code being tested. According to the relevant condition code, the opcode is assembled as indicated in Table 85.

Table 85. RET CC Opcode Detail

Condition	Relevant Flag	Opcode (hex)
NZ (nonzero)	Z	`C0`
Z (0)	Z	`C8`
NC (no carry)	C	`D0`
C (carry)	C	`D8`
PO (parity odd)	P/V	`E0`
PE (parity even)	P/V	`E8`
P (sign positive)	S	`F0`
M (sign negative/minus)	S	`F8`

RETI

Return from Maskable Interrupt

Operation

PC ← **(SP)**

Description

The **RETI** instruction returns program control back to the point in the user's application code where an interrupt caused the program control to jump to the current maskable interrupt service routine. The return address pops from the stack and is written to the Program Counter. Before the device executes the **RETI** instruction, the enable interrupt instruction (**EI**) should execute to allow recognition of interrupts after completion of the current interrupt service routine. The MADL control bit must be set to 1 to enable mixed-ADL mode interrupts. If the MADL is reset to 0, the suffixed instructions do not operate correctly. More detailed operation is provided in Table 86,

Table 86. RET Instruction Detail

ADL	Suffix	Operation
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]}.
1	None	The starting Program Counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0].

Table 86. RET Instruction Detail

ADL	Suffix	Operation
0	.S	An invalid suffix. **RETI.L** must be used in all mixed-memory mode applications.
1	.S	An invalid suffix. **RETI.L** must be used in all mixed-memory mode applications.

0	. L	The MADL control bit must be set to 1 to enable mixed-ADL mode interrupts. The starting Program Counter is {MBASE, PC[15:0]}. Pop a byte from SPL into ADL to set the new memory mode (03h = ADL, 02h = Z80).
		If ADL mode {
		Pop the upper byte of the return address from SPL into PC[23:16]. Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0]. The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from {MBASE,SPS} into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}.
		}

1	.L	The MADL control bit must be set to 1 to enable mixed-ADL mode interrupts. The starting Program Counter is PC[23:0]. Pop a byte from S2L into ADL to set the new memory mode (03h = ADL, 02h = Z80).
		If ADL mode {
		Pop 3-byte return address from SPL into PC[23:0]. The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from SPL into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}.
		}

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RETI	—	0/1	6/7	`ED, 4D`
RETI.L	—	0	8 if return to Z80 Mode, 9 if return to ADL Mode	`49, ED, 4D`
RETI.L	—	1	8 if return to Z80 Mode, 9 if return to ADL Mode	`5B, ED, 4D`

RETN

Return from Nonmaskable Interrupt

Operation

PC ← **(SP)**
IEF1 ← IEF2

Description

The **RETN** instruction returns program control back to the point in the user's application code where an interrupt caused the program control to jump to the current nonmaskable interrupt service routine. The return address pops from the stack and is written to the Program Counter. The state of IEF2 is copied back into IEF1. As a result of this copy operation, maskable interrupts become immediately enabled following the **RETN**, but only if they were enabled before the nonmaskable interrupt occurred.

The MADL control bit must be set to 1 to enable mixed-ADL mode interrupts. If the MADL is reset to 0, the suffixed instructions do not operate correctly. More detailed operation is provided in Table 87.

Table 87. RETN Instruction Detail

ADL	Suffix	Operation
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Pop a 2-byte return address from {MBASE, SPS} into PC[15:0]. The ADL mode bit remains cleared to 0. The ending Program Counter is {MBASE, PC[15:0]}.
1	None	The starting Program Counter is PC[23:0]. Pop a 3-byte return address from SPL into PC[23:0]. The ADL mode bit remains set to 1. The ending Program Counter is PC[23:0].
0	.S	An invalid suffix. **RETN.L** must be used in all mixed-memory mode applications.
1	.S	An invalid suffix. **RETN.L** must be used in all mixed-memory mode applications.

Table 87. RETN Instruction Detail (Continued)

ADL	Suffix	Operation

0	. L	The MADL control bit must be set to 1 to enable mixed-ADL mode interrupts. The starting Program Counter is {MBASE, PC[15:0]}. Pop a byte from SPL into ADL to set the new memory mode (`03h`=ADL, `02h`=Z80).
		if ADL mode {
		Pop the upper byte of the return address from SPL into PC[23:16].
		Pop 2 LS bytes of the return address from {MBASE, SPS} into PC[15:0]. The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from {MBASE,SPS} into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}.
		}

1	.L	The MADL control bit must be set to 1 to enable mixed-ADL mode interrupts The starting Program Counter is PC[23:0]. Pop a byte from SPL into ADL to set the new memory mode (`03h`=ADL, `02h`=Z80).
		if ADL mode {
		Pop 3-byte return address from SPL into PC[23:0]. The ending Program Counter is PC[23:0]
		}
		else Z80 mode {
		Pop a 2-byte return address from SPL into PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}.
		}

Condition Bits Affected

None.

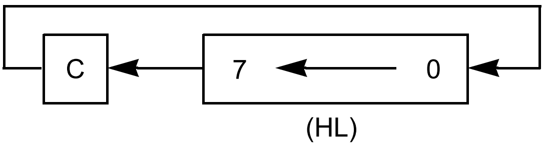
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RETN	—	0/1	6/7	`ED, 45`
RETN.L	—	0	8 if return to Z80 Mode, 9 if return to ADL Mode	`49, ED, 45`
RETN.L	—	1	8 if return to Z80 Mode, 9 if return to ADL Mode	`5B, ED, 45`

RL (HL)

Rotate Left

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies the previous contents of the Carry Flag into bit 0 of the memory location, (HL).

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

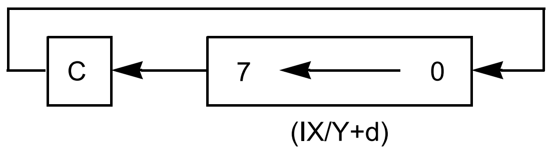
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RL	(HL)	X	5	`CB, 16`
RL.S	(HL)	1	6	`52, CB, 16`
RL.L	(HL)	0	6	`49, CB, 16`

RL (IX/Y+d)

Rotate Left

Operation



Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y**+**d**), by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies the previous contents of the Carry Flag into bit 0 of the memory location, (**IX/Y**+**d**).

Condition Bits Affected

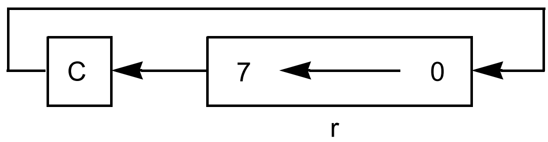
S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RL	(IX+**d**)	X	7	`DD, CB, dd, 16`
RL.S	(IX+**d**)	1	8	`52, DD, CB, dd, 16`
RL.L	(IX+**d**)	0	8	`49, DD, CB, dd, 16`
RL	(IY+**d**)	X	7	`FD, CB, dd, 16`
RL.S	(IY+**d**)	1	8	`52, FD, CB, dd, 16`
RL.L	(IY+**d**)	0	8	`49, FD, CB, dd, 16`

RL r
Rotate Left

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies the previous contents of the Carry Flag into bit 0 of the **r** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RL	**r**	X	2	`CB, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 88.

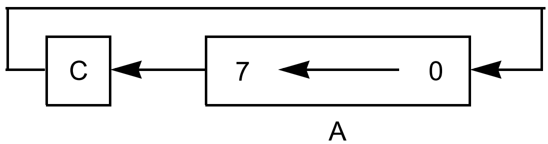
Table 88. Register and `jj` Opcodes for RL r Instruction (hex)

Register	`jj`
A	17
B	10
C	11
D	12
E	13
H	14
L	15

RLA

Rotate Left Accumulator

Operation



Description

The CPU manipulates the contents of the accumulator, A, by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies the previous contents of the Carry Flag into bit 0 of the **m** operand.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Not affected.
N	Reset.
C	Data from bit 7 of the accumulator.

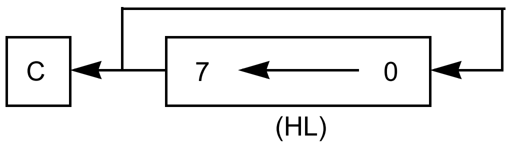
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RLA	—	X	1	17

RLC (HL)

Rotate Left with Carry

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and into bit 0 of the memory location, (HL).

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

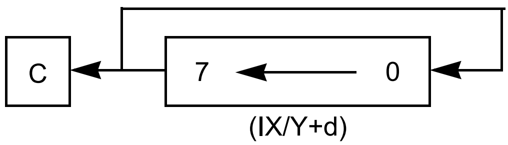
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RLC	(HL)	X	5	`CB, 06`
RLC.S	(HL)	1	6	`52, CB, 06`
RLC.L	(HL)	0	6	`49, CB, 06`

RLC (IX/Y+d)

Rotate Left with Carry

Operation



Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y**+**d**), by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and into bit 0 of the memory location, (**IX/Y**+**d**).

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

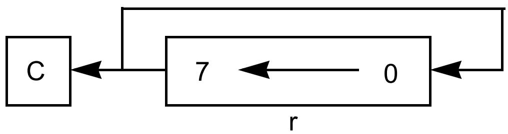
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RLC	(IX+**d**)	X	7	`DD, CB, dd, 06`
RLC.S	(IX+**d**)	1	8	`52, DD, CB, dd, 06`
RLC.L	(IX+**d**)	0	8	`49, DD, CB, dd, 06`
RLC	(IY+**d**)	X	7	`FD, CB, dd, 06`
RLC.S	(IY+**d**)	1	8	`52, FD, CB, dd, 06`
RLC.L	(IY+**d**)	0	8	`49, FD, CB, dd, 06`

RLC r

Rotate Left with Carry

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and into bit 0 of the **r** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RLC	**r**	X	2	`CB, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 89.

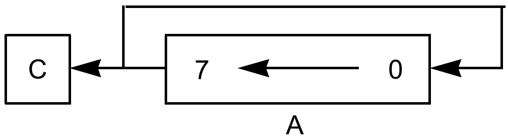
Table 89. Register and jj Opcodes for RLC r Instruction (hex)

Register	jj
A	07
B	00
C	01
D	02
E	03
H	04
L	05

RLCA

Rotate Left with Carry–Accumulator

Operation



Description

The CPU manipulates the contents of the accumulator, A, by rotating them left one bit position. The CPU next copies bit 7 into the Carry Flag and into bit 0.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Not affected.
N	Reset.
C	Data from bit 7 of the accumulator

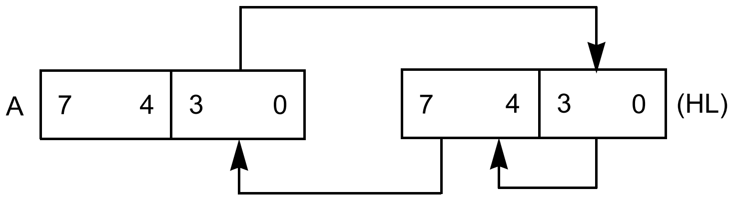
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RLCA	—	X	1	07

RLD

Rotate Left Decimal

Operation



A[3:0] ← HL[7:4]
HL[7:4] ← HL[3:0]
HL[3:0] ← A[3:0]

Description

The CPU copies the contents of the low-order four bits of the memory location (HL) into the high-order four bits of the (HL). The CPU next copies the previous contents of the high-order four bits of the (HL) into the low-order four bits of the accumulator, A. The CPU next copies the previous contents of the low-order four bits of the accumulator into the low-order four bits of the (HL).

Condition Bits Affected

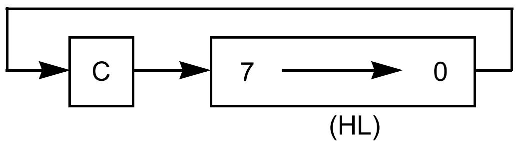
S	Set if the accumulator is negative; reset otherwise.
Z	Set if the accumulator is 0; reset otherwise.
H	Reset.
P/V	Set if parity of the accumulator is even; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RLD	—	X	5	ED, 6F

RR (HL)
Rotate Right

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and copies the previous contents of the Carry Flag into bit 7 of the memory location, (HL).

Condition Bits Affected

\

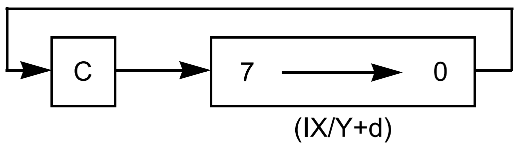
S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RR	(HL)	X	5	`CB, 1E`
RR.S	(HL)	1	6	`52, CB, 1E`
RR.L	(HL)	0	6	`49, CB, 1E`

RR (IX/Y+d)
Rotate Right

Operation



Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y**+**d**), by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and copies the previous contents of the Carry Flag into bit 7 of the memory location, (**IX/Y**+**d**).

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

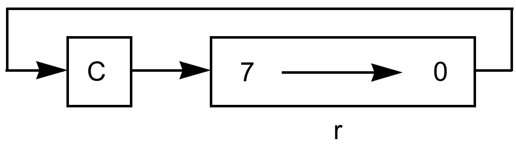
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RR	(IX+**d**)	X	7	`DD, CB, dd, 1E`
RR.S	(IX+**d**)	1	8	`52, DD, CB, dd, 1E`
RR.L	(IX+**d**)	0	8	`49, DD, CB, dd, 1E`
RR	(IY+**d**)	X	7	`FD, CB, dd, 1E`
RR.S	(IY+**d**)	1	8	`52, FD, CB, dd, 1E`
RR.L	(IY+**d**)	0	8	`49, FD, CB, dd, 1E`

RR r

Rotate Right

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and copies the previous contents of the Carry Flag into bit 7 of the **r** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RR	**r**	X	2	`CB, kk`

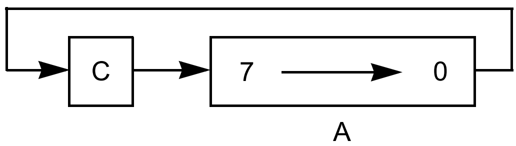
`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 90.

Table 90. Register and `jj` Opcodes for RR r Instruction (hex)

Register	`jj`
A	1F
B	18
C	19
D	1A
E	1B
H	1C
L	1D

RRA
Rotate Right–Accumulator

Operation



Description

The CPU manipulates the contents of the accumulator, A, by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and copies the previous contents of the Carry Flag into bit 7.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Not affected.
N	Reset.
C	Data from bit 0 of the accumulator.

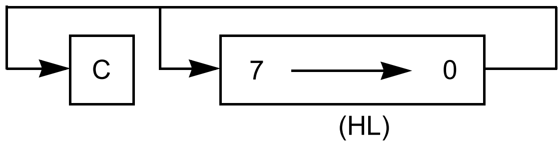
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RRA	—	X	1	`1F`

RRC (HL)

Rotate Right with Carry

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and into bit 7 of the memory location, (HL).

Condition Bits Affected\

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

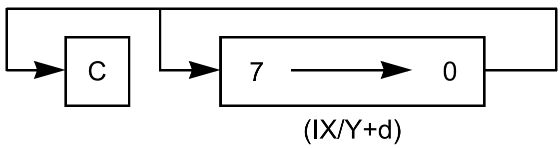
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RRC	(HL)	X	5	`CB, 0E`
RRC.S	(HL)	1	6	`52, CB, 0E`
RRC.L	(HL)	0	6	`49, CB, 0E`

RRC (IX/Y+d)

Rotate Right with Carry

Operation



Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y**+**d**), by rotating them right one bit position. The CPU next copies bit 0 into the Carry Flag and into bit 7 of the memory location (**IX/Y**+**d**).

Condition Bits Affected

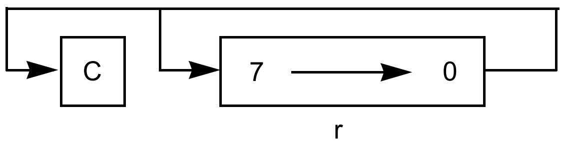
S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RRC	(IX+**d**)	X	7	`DD, CB, dd, 0E`
RRC.S	(IX+**d**)	1	8	`52, DD, CB, dd, 0E`
RRC.L	(IX+**d**)	0	8	`49, DD, CB, dd, 0E`
RRC	(IY+**d**)	X	7	`FD, CB, dd, 0E`
RRC.S	(IY+**d**)	1	8	`52, FD, CB, dd, 0E`
RRC.L	(IY+**d**)	0	8	`49, FD, CB, dd, 0E`

RRC r
Rotate Right with Carry

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and into bit 7 of the **r** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RRC	**r**	X	2	`CB, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 91.

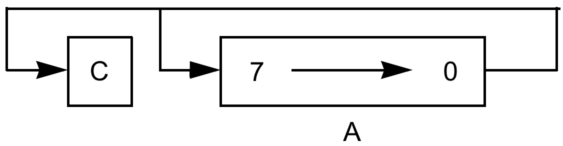
Table 91. Register and `jj` Opcodes for RRC r Instruction (hex)

Register	jj
A	0F
B	08
C	09
D	0A
E	0B
H	0C
L	0D

RRCA

Rotate Right with Carry–Accumulator

Operation



Description

The CPU manipulates the contents of the accumulator, A, by rotating them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and into bit 7.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Not affected.
N	Reset.
C	Data from bit 0 of the accumulator.

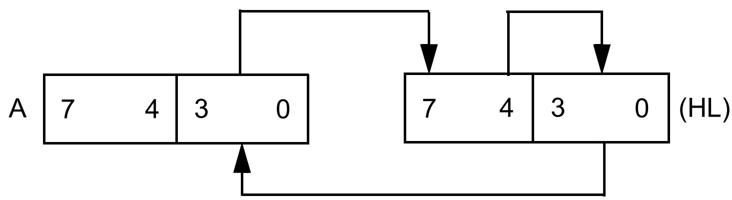
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RRCA	—	X	1	`0F`

RRD

Rotate Right Decimal

Operation



A[3:0] ← HL[3:0]
HL[7:4] ← A[3:0]
HL[3:0] ← HL[7:4]

Description

The CPU copies the contents of the low-order four bits of the memory location (HL) into the low-order four bits of the accumulator, A. The CPU next copies the previous contents of the low-order four bits of the accumulator into the high-order four bits of (HL). The CPU next copies the previous contents of the high-order four bits of (HL) into the low-order four bits of (HL).

Condition Bits Affected

S	Set if the accumulator is negative; reset otherwise.
Z	Set if the accumulator is 0; reset otherwise.
H	Reset.
P/V	Set if parity of the accumulator is even; reset otherwise.
N	Reset.
C	Not affected.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RRD	—	X	5	`ED, 67`

RSMIX

Reset MIXED MEMORY Mode Flag

Operation

MADL ← 0

Description

The MIXED MEMORY Mode Flag (MADL) is reset to 0.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RSMIX	—	X	2	`ED, 7E`

RST n

Restart

Operation

(SP) ← PC
PC ← {0000h,**n**}

Description

The **RST** instruction functions similar to a **CALL** instruction. However, the 8-bit **n** operand is limited to 8 specific values: 00h, 08h, 10h, 18h, 20h, 28h, 30h, and 38h. After stacking the Program Counter (and ADL mode bit, if necessary), the **RST** instruction is written the 8-bit restart vector **n** to the Program Counter.

Table 92. RST N Instruction Detail

ADL	Suffix	Operation
0	None	The starting Program Counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the {MBASE,SPS} stack. The ADL mode bit remains cleared to 0. Write {00h, nn} to PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]} = {MBASE, 00h, nn}.
1	None	The starting Program Counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. The ADL mode bit remains set to 1. Write {0000h, nn} to PC[23:0]. The ending Program Counter is PC[23:0] = {0000h, nn}.

Table 92. RST N Instruction Detail

0	.S	The starting Program Counter is {MBASE, PC[15:0]} Push the 2-byte return address, PC[15:0], onto the {MBASE, SPS} stack. Push a `02h` byte onto the SPL stack, indicating an interrupt from Z80 mode (ADL=0). The ADL mode bit remains cleared to 0. Write {`00h`, nn} to PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}={MBASE, `00h`, nn}.
1	.S	The starting Program Counter is PC[23:0]. Push the 2 LS bytes of the return address, PC[15:0], onto the {MBASE, SPS} stack. Push the MS byte of the return address, PC[23:16], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode, because ADL=1. Reset ADL mode bit to 0. Write {`00h`, nn} to PC[15:0]. The ending Program Counter is {MBASE, PC[15:0]}={MBASE, `00h`, nn}.
0	.L	The starting Program Counter is {MBASE, PC[15:0]}. Push the 2-byte return address, PC[15:0], onto the SPL stack. Push a `02h` byte onto the SPL stack, indicating an interrupt from Z80 mode, because ADL=0. Set the ADL mode bit to 1. Write {`0000h`, nn} to PC[23:0]. The ending Program Counter is PC[23:0]={`0000h`, nn}.
1	.L	The starting Program Counter is PC[23:0]. Push the 3-byte return address, PC[23:0], onto the SPL stack. Push a `03h` byte onto the SPL stack, indicating an interrupt from ADL mode, because ADL=1. The ADL mode bit remains set to 1. Write {`0000h`, nn} to PC[23:0]. The ending Program Counter is PC[23:0]={`0000h`, nn}.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
RST n	n	0/1	5/6	kk
RST.S n	n	1	8	52, kk
RST.L n	n	0	7	49, kk

The opcode (kk) is a function of the 8-bit Restart Address, **n**, and is assembled into one of the opcodes indicated in Table 93.

Table 93. Restart Address and kk Opcodes for RST n Instruction (hex)

Restart Address	kk
00h	C7
08h	C
10h	D7
18h	DF
20h	E7
28h	EF
30h	F7
38h	FF

SBC A, (HL)

Subtract with Carry

Operation

A ← A–(HL)–C

Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). This 8-bit value and the Carry Flag (C) are subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	A,(HL)	X	2	`9E`
SBC.S	A,(HL)	1	3	`52, 9E`
SBC.L	A,(HL)	0	3	`49, 9E`

SBC A, ir

Subtract with Carry

Operation

A ← A−**ir**−C

Description

The **rr** operand is any of the 8-bit registers IXH, IXL, IYH, IYL. The **ir** operand and the Carry Flag (C) are subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	A,IXH	X	2	`DD, 9C`
SBC	A,IXL	X	2	`DD, 9D`
SBC	A,IYH	X	2	`FD, 9C`
SBC	A,IYL	X	2	`FD, 9D`

SBC A, (IX/Y+d)

Subtract with Carry

Operation

A ← A–(**IX/Y**+**d**)–C

Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value and the Carry Flag (C) are subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	A,(IX+**d**)	X	4	`DD, 9E, dd`
SBC.S	A,(IX+**d**)	1	5	`52, DD, 9E, dd`
SBC.L	A,(IX+**d**)	0	5	`49, DD, 9E, dd`
SBC	A,(IY+**d**)	X	4	`FD, 9E, dd`
SBC.S	A,(IY+**d**)	1	5	`52, FD, 9E, dd`
SBC.L	A,(IY+**d**)	0	5	`49, FD, 9E, dd`

SBC A, n

Subtract with Carry

Operation

A ← A−**n**−C

Description

The 8-bit immediate value **n** and the Carry Flag (C) are subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	A,**n**	X	2	`DE, nn`

SBC A, r

Subtract with Carry

Operation

A ← A–**r**–C

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand and the Carry Flag (C) are subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 94.

Table 94. Register and `jj` Opcodes for SBC A, r Instruction (hex)

Register	`jj`
A	9F
B	98
C	99
D	9A
E	9B
H	9C
L	9D

SBC HL, rr

Subtract with Carry

Operation

HL ← HL−**rr**−C

Description

The **rr** operand is any of the multibyte CPU registers BC, DE, or HL. The **rr** operand and the Carry Flag (C) are subtracted from the contents of the HL register. The result is written to HL.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 12; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	HL,**rr**	X	2	`ED, kk`
SBC.S	HL,**rr**	1	3	`52, ED, kk`
SBC.L	HL,**rr**	0	3	`49, ED, kk`

`kk` identifies either the BC, DE, HL, or **SP** multibyte register and is assembled into one of the Opcodes indicated in Table 95.

Table 95. Register and `kk` Opcodes for SBC HL, rr Instruction (hex)

Register	`kk`
BC	`42`
DE	`52`
HL	`62`

SBC HL, SP

Subtract with Carry

Operation

HL ← HL−**SP**−C

Description

The Stack Pointer (**SP**) and the Carry Flag (C) are subtracted from the contents of the HL register. The result is written to HL. In ADL mode, or if the **.L** suffix is employed, the 24-bit Stack Pointer Long (SPL) is used. In Z80 mode of if the **.S** suffix is employed, the 16-bit Stack Pointer Short (SPS) is used.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 12; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SBC	HL,**SP**	X	2	`ED, 72`
SBC.S	HL,**SP**	1	3	`52, ED, 72`
SBC.L	HL,**SP**	0	3	`49, ED, 72`

SCF
Set Carry Flag

Operation

C ← 1

Description

The Carry Flag, C, is set to 1.

Condition Bits Affected

S	Not affected.
Z	Not affected.
H	Reset.
P/V	Not affected.
N	Reset.
C	Set.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SCF	—	X	1	37

SET b, (HL)

Set Bit

Operation

(HL)[**b**] ← 1

Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). Bit **b** of this 8-bit value is set to 1.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SET	**b**,(HL)	X	3	`CB, kk`
SET.S	**b**,(HL)	1	4	`52, CB, kk`
SET.L	**b**,(HL)	0	4	`49, CB, kk`

`kk`=binary code `11 bbb 110`; where `bbb` identifies the bit tested and assembled into the object code, as indicated in Table 96.

Table 96. `bbb` Opcodes for SET b, (HL) Instruction (hex)

Bit Tested	`bbb`	Bit Tested	`bbb`
0	`000`	4	`100`
1	`001`	5	`101`
2	`010`	6	`110`
3	`011`	7	`111`

SET b, (IX/Y+d)

Set Bit

Operation

(**IX/Y**+**d**)[**b**] ← 1

Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. Bit **b** this 8-bit value is set to 1.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SET	**b**,(IX+**d**)	X	5	`DD, CB, dd, kk`
SET.S	**b**,(IX+**d**)	1	6	`52, DD, CB, dd, kk`
SET.L	**b**,(IX+**d**)	0	6	`49, DD, CB, dd, kk`
SET	**b**,(IY+**d**)	X	5	`FD, CB, dd, kk`
SET.S	**b**,(IY+**d**)	1	6	`52, FD, CB, dd, kk`
SET.L	**b**,(IY+**d**)	0	6	`49, FD, CB, dd, kk`

`kk`=binary code `11 bbb 110`; where `bbb` identifies the bit tested and assembled into the object code, as indicated in Table 97.

Table 97. `bbb` Opcodes for SET b, (IX/Y+d) Instruction (hex)

Bit Tested	`bbb`
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

SET b, r

Set Bit

Operation

r[**b**] ← 1

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. Bit **b** of the specified register is set to 1.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SET	**b,r**	X	2	`CB, jj`

`jj`=binary code `11 bbb rrr` and `kk`=binary code `11 bbb 110`; where `rrr` identifies the A, B, C, D, E, H, or L register and `bbb` identifies the bit tested and assembled into the object code, as indicated in Table 98.

Table 98. `bbb`, Register, and `rrr` Opcodes for SET b, r Instruction (hex)

Bit Tested	`bbb`	Register	`rrr`
0	`000`	A	`111`
1	`001`	B	`000`
2	`010`	C	`001`
3	`011`	D	`010`

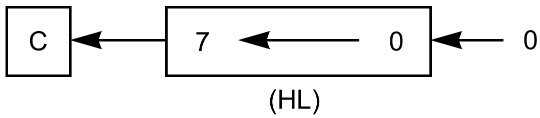
Table 98. `bbb`, Register, and `rrr` Opcodes for SET b, r Instruction (hex)

Bit Tested	bbb	Register	rrr
4	100	E	011
5	101	H	100
6	110	L	101
7	111		

SLA (HL)

Shift Left Arithmetic

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by shifting them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies a 0 into bit 0 of the memory location, (HL).

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

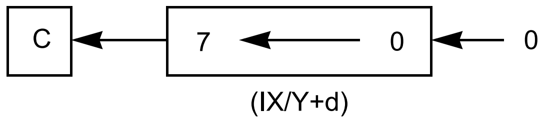
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SLA	(HL)	X	5	`CB, 26`
SLA.S	(HL)	1	6	`52, CB, 26`
SLA.L	(HL)	0	6	`49, CB, 26`

SLA (IX/Y+d)

Shift Left Arithmetic

Operation



Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y**+**d**), by shifting them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies a 0 into bit 0 of the memory location, (**IX/Y**+**d**).

Condition Bits Affected

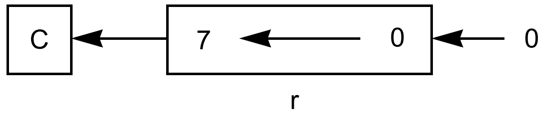
S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SLA	(IX+**d**)	X	7	`DD, CB, dd, 26`
SLA.S	(IX+**d**)	1	8	`52, DD, CB, dd, 26`
SLA.L	(IX+**d**)	0	8	`49, DD, CB, dd, 26`
SLA	(IY+**d**)	X	7	`FD, CB, dd, 26`
SLA.S	(IY+**d**)	1	8	`52, FD, CB, dd, 26`
SLA.L	(IY+**d**)	0	8	`49, FD, CB, dd, 26`

SLA r
Shift Left Arithmetic

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by shifting them left one bit position. The CPU next copies bit 7 into the Carry Flag and copies a 0 into bit 0 of the **r** operand.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 7 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SLA	**r**	X	2	`CB, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 99.

Table 99. Register and `jj` Opcodes for SLA r Instruction (hex)

Register	`jj`
A	27
B	20
C	21
D	22
E	23
H	24
L	25

SLP

Sleep

Operation

This instruction places the CPU into SLEEP mode.

Description

SLEEP mode may not be supported on some eZ80® devices. Refer to the individual product specification for a detailed description.

Condition Bits Affected

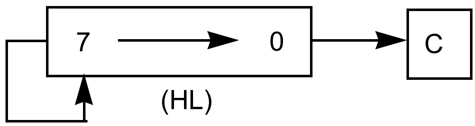
None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SLP	—	X	2	`ED, 76`

SRA (HL)
Shift Right Arithmetic

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by shifting them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and leaves the previous contents of bit 7 unchanged.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

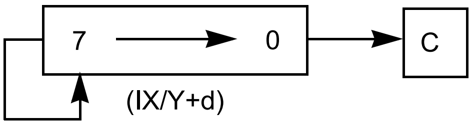
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SRA	(HL)	X	5	`CB, 2E`
SRA.S	(HL)	1	6	`52, CB, 2E`
SRA.L	(HL)	0	6	`49, CB, 2E`

SRA (IX/Y+d)

Shift Right Arithmetic

Operation



Description

The (**IX/Y+d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y+d**), by shifting them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and leaves the previous contents of bit 7 unchanged.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

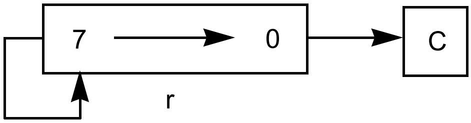
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SRA	(IX+**d**)	X	7	`DD, CB, dd, 2E`
SRA.S	(IX+**d**)	1	8	`52, DD, CB, dd, 2E`
SRA.L	(IX+**d**)	0	8	`49, DD, CB, dd, 2E`
SRA	(IY+**d**)	X	7	`FD, CB, dd, 2E`
SRA.S	(IY+**d**)	1	8	`52, FD, CB, dd, 2E`
SRA.L	(IY+**d**)	0	8	`49, FD, CB, dd, 2E`

SRA r

Shift Right Arithmetic

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by shifting them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and leaves the previous contents of bit 7 unchanged.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SRA	**r**	X	2	`CB, jj`

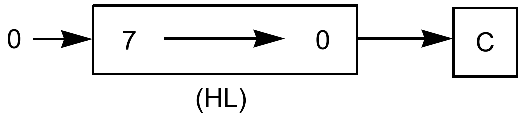
`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 100.

Table 100. Register and `jj` Opcodes for SRA r Instruction (hex)

Register	`jj`
A	`2F`
B	`28`
C	`29`
D	`2A`
E	`2B`
H	`2C`
L	`2D`

SRL (HL)
Shift Right Logical

Operation



Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). The CPU manipulates the contents of this memory location, (HL), by shifting them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and resets bit 7.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

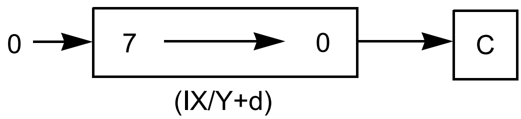
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SRL	(HL)	X	5	`CB, 3E`
SRL.S	(HL)	1	6	`52, CB, 3E`
SRL.L	(HL)	0	6	`49, CB, 3E`

SRL (IX/Y+d)

Shift Right Logical

Operation



Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. The CPU manipulates the contents of this memory location, (**IX/Y**+**d**), by shifting them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and resets bit 7.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

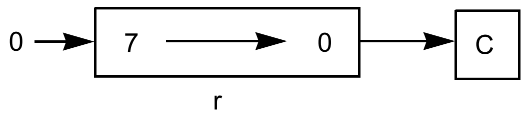
Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SRL	(IX+**d**)	X	7	`DD, CB, dd, 3E`
SRL.S	(IX+**d**)	1	8	`52, DD, CB, dd, 3E`
SRL.L	(IX+**d**)	0	8	`49, DD, CB, dd, 3E`
SRL	(IY+**d**)	X	7	`FD, CB, dd, 3E`
SRL.S	(IY+**d**)	1	8	`52, FD, CB, dd, 3E`
SRL.L	(IY+**d**)	0	8	`49, FD, CB, dd, 3E`

SRL r

Shift Right Logical

Operation



Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The CPU manipulates the contents of the **r** operand by shifting them right one bit position. The CPU next copies the contents of bit 0 into the Carry Flag and resets bit 7.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Data from bit 0 of the source.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SRL	**r**	X	2	`CB, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 101.

Table 101. Register and jj Opcodes for SRL r Instruction (hex)

Register	jj
A	3F
B	38
C	39
D	3A
E	3B
H	3C
L	3D

STMIX

Set MIXED MEMORY Mode Flag

Operation

MADL ← 1

Description

The MIXED MEMORY Mode Flag (MADL) is set to 1.

Condition Bits Affected

None.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
STMIX	—	X	2	`ED, 7D`

SUB A, (HL)

Subtract without Carry

Operation

A ← A–(HL)

Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). This 8-bit value is subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SUB	A,(HL)	X	2	`96`
SUB.S	A,(HL)	1	3	`52, 96`
SUB.L	A,(HL)	0	3	`49, 96`

SUB A, ir

Subtract without Carry

Operation

A ← A–**ir**

Description

The **ir** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The **ir** operand is subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SUB	A,IXH	X	2	`DD, 94`
SUB	A,IXL	X	2	`DD, 95`
SUB	A,IYH	X	2	`FD, 94`
SUB	A,IYL	X	2	`FD, 95`

SUB A, (IX/Y+d)

Subtract without Carry

Operation

A ← A–**(IX/Y+d)**

Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value is subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SUB	A,(IX+**d**)	X	4	`DD, 96, dd`
SUB.S	A,(IX+**d**)	1	5	`52, DD, 96, dd`
SUB.L	A,(IX+**d**)	0	5	`49, DD, 96, dd`
SUB	A,(IY+**d**)	X	4	`FD, 96, dd`
SUB.S	A,(IY+**d**)	1	5	`52, FD, 96, dd`
SUB.L	A,(IY+**d**)	0	5	`49, FD, 96, dd`

SUB A, n

Subtract without Carry

Operation

A ← A–**n**

Description

The 8-bit immediate value **n** is subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SUB	A,**n**	X	2	`D6, nn`

SUB A, r

Subtract without Carry

Operation

A ← A–**r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand is subtracted from the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set if borrow from bit 4; reset otherwise.
P/V	Set if overflow; reset otherwise.
N	Set.
C	Set if borrow; reset otherwise.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
SUB	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 102.

Table 102. Register and `jj` Opcodes for SUB A, r Instruction (hex)

Register	`jj`
A	97
B	90
C	91
D	92
E	93
H	94
L	95

TST A, (HL)

Test

Operation

A **AND** (HL)

Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). This 8-bit value is bitwise ANDed with the contents of the accumulator, A. The appropriate flags are set to 1, depending on the result of the **AND** logical operation. The contents of the accumulator and the memory location are not altered.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
TST	A,(HL)	X	3	`ED, 34`
TST.S	A,(HL)	1	4	`52, ED, 34`
TST.L	A,(HL)	0	4	`49, ED, 73`

TST A, n

Test

Operation

A **AND n**

Description

The 8-bit immediate value **n** is bitwise ANDed with the contents of the accumulator, A. The appropriate flags are set to 1, depending on the result of the **AND** logical operation. The contents of the accumulator are not altered.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
TST	A,**n**	X	3	`ED, 64, nn`

TST A, r

Test

Operation

A **AND r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand is bitwise ANDed with the contents of the accumulator, A. The appropriate flags are set to 1, depending on the result of the **AND** logical operation. The contents of the accumulator and the **r** operand are not altered.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
TST	A,**r**	X	2	`ED, jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 103.

Table 103. Register and `jj` Opcodes for TST A, r Instruction (hex)

Register	`jj`
A	`3C`
B	`04`
C	`0C`
D	`14`
E	`1C`
H	`24`
L	`2C`

TSTIO n

Test I/O Byte

Operation

{`0000h`, C} **AND n**

Description

The CPU places the contents of the C register onto the lower byte of the address bus, ADDR[7:0], while it forces the two upper bytes of the address bus, ADDR[23:0], to 0s. The data at this I/O address {`0000h`, C}, is bitwise ANDed with the 8-bit immediate value **n**. The appropriate flags are set to 1, depending on the result of the **AND** logical operation.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Set.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
TSTIO	**n**	X	4	`ED, 74, nn`

XOR A, (HL)
Logical Exclusive OR

Operation

A ← A **XOR** (HL)

Description

The (HL) operand is an 8-bit value at the memory location specified by the contents of the multibyte register (HL). This 8-bit value is bitwise exclusive-ORed with the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
XOR	A,(HL)	X	2	`AE`
XOR.S	A,(HL)	1	3	`52, AE`
XOR.L	A,(HL)	0	3	`49, AE`

XOR A, ir

Logical Exclusive OR

Operation

A ← A **XOR ir**

Description

The **ir** operand is any of the 8-bit registers IXH, IXL, IYH, or IYL. The **ir** operand is bitwise exclusive-ORed with the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
XOR	A,IXH	X	2	`DD, AC`
XOR	A,IXL	X	2	`DD, AD`
XOR	A,IYH	X	2	`FD, AC`
XOR	A,IYL	X	2	`FD, AD`

XOR A, (IX/Y+d)

Logical Exclusive OR

Operation

A ← A **XOR** (**IX/Y**+**d**)

Description

The (**IX/Y**+**d**) operand is an 8-bit value at the memory location specified by the contents of the Index Register, IX or IY, added to the two's-complement displacement **d**. This 8-bit value is bitwise exclusive-ORed with the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
XOR	A,(IX+**d**)	X	4	`DD, AE, dd`
XOR.S	A,(IX+**d**)	1	5	`52, DD, AE, dd`
XOR.L	A,(IX+**d**)	0	5	`49, DD, AE, dd`
XOR	A,(IY+**d**)	X	4	`FD, AE, dd`
XOR.S	A,(IY+**d**)	1	5	`52, FD, AE, dd`
XOR.L	A,(IY+**d**)	0	5	`49, FD, AE, dd`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 104.

Table 104. Register and `jj` Opcodes for XOR A, (IX/Y+d) Instruction (hex)

Register	jj
A	AF
B	A8
C	A9
D	AA
E	AB
H	AC
L	AD

XOR A, n
Logical Exclusive OR

Operation

A ← A **XOR n**

Description

The 8-bit immediate value **n** is bitwise exclusive-ORed with the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset.

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
XOR	A,**n**	X	2	`EE, nn`

XOR A, r

Logical Exclusive OR

Operation

A ← A **XOR r**

Description

The **r** operand is any of the 8-bit CPU registers A, B, C, D, E, H, or L. The **r** operand is bitwise exclusive-ORed with the contents of the accumulator, A. The result is written to the accumulator.

Condition Bits Affected

S	Set if result is negative; reset otherwise.
Z	Set if result is 0; reset otherwise.
H	Reset.
P/V	Set if parity is even; reset otherwise.
N	Reset.
C	Reset

Attributes

Mnemonic	Operand	ADL Mode	Cycle	Opcode (hex)
XOR	A,**r**	X	1	`jj`

`jj` identifies the A, B, C, D, E, H, or L register and is assembled into one of the opcodes indicated in Table 105.

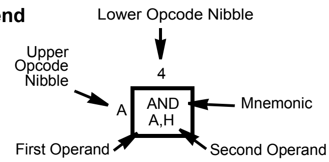
Table 105. Register and `jj` Opcodes for XOR A, r Instruction (hex)

Register	`jj`
A	`AF`
B	`A8`
C	`A9`
D	`AA`
E	`AB`
H	`AC`
L	`AD`

Opcode Maps

Table 106. Opcode Map—First Opcode

Legend



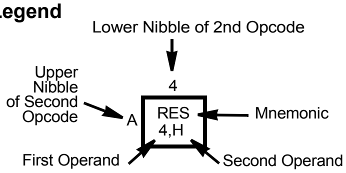
Lower Nibble (Hex) across the top; Upper Nibble (Hex) down the side.

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LD BC, Mmn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA
1	DJNZ d	LD DE, Mmn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR d	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RRA
2	JR NZ,d	LD HL, Mmn	LD (Mmn), HL	INC HL	INC H	DEC H	LD H,n	DAA	JR Z,d	ADD HL,HL	LD HL, (Mmn)	DEC HL	INC L	DEC L	LD L,n	CPL
3	JR NC,d	LD SP, Mmn	LD (Mmn), A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR C,d	ADD HL,SP	LD A, (Mmn)	DEC SP	INC A	DEC A	LD A,n	CCF
4	.SIS suffix	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	.LIS suffix	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
5	LD D,B	LD D,C	.SIL suffix	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	.LIL suffix	LD E,H	LD E,L	LD E,(HL)	LD E,A
6	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
7	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
8	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
9	SUB A,B	SUB A,C	SUB A,D	SUB A,E	SUB A,H	SUB A,L	SUB A,(HL)	SUB A,A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
A	AND A,B	AND A,C	AND A,D	AND A,E	AND A,H	AND A,L	AND A,(HL)	AND A,A	XOR A,B	XOR A,C	XOR A,D	XOR A,E	XOR A,H	XOR A,L	XOR A,(HL)	XOR A,A
B	OR A,B	OR A,C	OR A,D	OR A,E	OR A,H	OR A,L	OR A,(HL)	OR A,A	CP A,B	CP A,C	CP A,D	CP A,E	CP A,H	CP A,L	CP A,(HL)	CP A,A
C	RET NZ	POP BC	JP NZ, Mmn	JP Mmn	CALL NZ, Mmn	PUSH BC	ADD A,n	RST 00h	RET Z	RET	JP Z, Mmn	Table 107	CALL Z, Mmn	CALL Mmn	ADC A,n	RST 08h
D	RET NC	POP DE	JP NC, Mmn	OUT (n),A	CALL NC, Mmn	PUSH DE	SUB A,n	RST 10h	RET C	EXX	JP C, Mmn	IN A,(n)	CALL C, Mmn	Table 108	SBC A,n	RST 18h
E	RET PO	POP HL	JP PO, Mmn	EX (SP),HL	CALL PO, Mmn	PUSH HL	AND A,n	RST 20h	RET PE	JP (HL)	JP PE, Mmn	EX DE,HL	CALL PE, Mmn	Table 109	XOR A,n	RST 28h
F	RET P	POP AF	JP P, Mmn	DI	CALL P, Mmn	PUSH AF	OR A,n	RST 30h	RET M	LD SP,HL	JP M, Mmn	EI	CALL M, Mmn	Table 110	CP A,n	RST 38h

Note: n=8-bit data; Mmn=16- or 24-bit addr or data; d=8-bit two's-complement displacement.

Table 107. Opcode Map—Second Opcode after `0CBh`

Legend



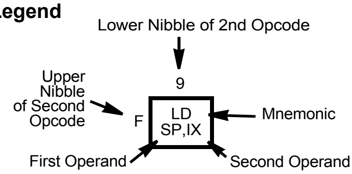
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC (HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
1	RL B	RL C	RL D	RL E	RL H	RL L	RL (HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
2	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA (HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
3									SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
4	BIT 0,B	BIT 0,C	BIT 0,D	BIT 0,E	BIT 0,H	BIT 0,L	BIT 0,(HL)	BIT 0,A	BIT 1,B	BIT 1,C	BIT 1,D	BIT 1,E	BIT 1,H	BIT 1,L	BIT 1,(HL)	BIT 1,A
5	BIT 2,B	BIT 2,C	BIT 2,D	BIT 2,E	BIT 2,H	BIT 2,L	BIT 2,(HL)	BIT 2,A	BIT 3,B	BIT 3,C	BIT 3,D	BIT 3,E	BIT 3,H	BIT 3,L	BIT 3,(HL)	BIT 3,A
6	BIT 4,B	BIT 4,C	BIT 4,D	BIT 4,E	BIT 4,H	BIT 4,L	BIT 4,(HL)	BIT 4,A	BIT 5,B	BIT 5,C	BIT 5,D	BIT 5,E	BIT 5,H	BIT 5,L	BIT 5,(HL)	BIT 5,A
7	BIT 6,B	BIT 6,C	BIT 6,D	BIT 6,E	BIT 6,H	BIT 6,L	BIT 6,(HL)	BIT 6,A	BIT 7,B	BIT 7,C	BIT 7,D	BIT 7,E	BIT 7,H	BIT 7,L	BIT 7,(HL)	BIT 7,A
8	RES 0,B	RES 0,C	RES 0,D	RES 0,E	RES 0,H	RES 0,L	RES 0,(HL)	RES 0,A	RES 1,B	RES 1,C	RES 1,D	RES 1,E	RES 1,H	RES 1,L	RES 1,(HL)	RES 1,A
9	RES 2,B	RES 2,C	RES 2,D	RES 2,E	RES 2,H	RES 2,L	RES 2,(HL)	RES 2,A	RES 3,B	RES 3,C	RES 3,D	RES 3,E	RES 3,H	RES 3,L	RES 3,(HL)	RES 3,A
A	RES 4,B	RES 4,C	RES 4,D	RES 4,E	RES 4,H	RES 4,L	RES 4,(HL)	RES 4,A	RES 5,B	RES 5,C	RES 5,D	RES 5,E	RES 5,H	RES 5,L	RES 5,(HL)	RES 5,A
B	RES 6,B	RES 6,C	RES 6,D	RES 6,E	RES 6,H	RES 6,L	RES 6,(HL)	RES 6,A	RES 7,B	RES 7,C	RES 7,D	RES 7,E	RES 7,H	RES 7,L	RES 7,(HL)	RES 7,A
C	SET 0,B	SET 0,C	SET 0,D	SET 0,E	SET 0,H	SET 0,L	SET 0,(HL)	SET 0,A	SET 1,B	SET 1,C	SET 1,D	SET 1,E	SET 1,H	SET 1,L	SET 1,(HL)	SET 1,A
D	SET 2,B	SET 2,C	SET 2,D	SET 2,E	SET 2,H	SET 2,L	SET 2,(HL)	SET 2,A	SET 3,B	SET 3,C	SET 3,D	SET 3,E	SET 3,H	SET 3,L	SET 3,(HL)	SET 3,A
E	SET 4,B	SET 4,C	SET 4,D	SET 4,E	SET 4,H	SET 4,L	SET 4,(HL)	SET 4,A	SET 5,B	SET 5,C	SET 5,D	SET 5,E	SET 5,H	SET 5,L	SET 5,(HL)	SET 5,A
F	SET 6,B	SET 6,C	SET 6,D	SET 6,E	SET 6,H	SET 6,L	SET 6,(HL)	SET 6,A	SET 7,B	SET 7,C	SET 7,D	SET 7,E	SET 7,H	SET 7,L	SET 7,(HL)	SET 7,A

Note: n=8-bit data; Mmn=16- or 24-bit addr or data; d=8-bit two's-complement displacement.

Table 108. Opcode Map—Second Opcode After `0DDh`

Legend



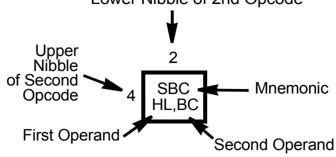
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								LD BC, (IX+d)		ADD IX,BC						LD (IX+d), BC
1								LD DE, (IX+d)		ADD IX,DE						LD (IX+d), DE
2		LD IX, Mmn	LD (Mmn), IX	INC IX	INC IXH	DEC IXH	LD IXH,n	LD HL, (IX+d)		ADD IX,IX	LD IX, (Mmn)	DEC IX	INC IXL	DEC IXL	LD IXL,n	LD (IX+d), HL
3		LD IY, (IX+d)			INC (IX+d)	DEC (IX+d)	LD (IX +d),n	LD IX, (IX+d)		ADD IX,SP					LD (IX+d), IY	LD (IX+d), IX
4					LD B,IXH	LD B,IXL	LD B, (IX+d)						LD C,IXH	LD C,IXL	LD C, (IX+d)	
5					LD D,IXH	LD D,IXL	LD D, (IX+d)						LD E,IXH	LD E,IXL	LD E, (IX+d)	
6	LD IXH,B	LD IXH,C	LD IXH,D	LD IXH,E	LD IXH,IXH	LD IXH,IXL	LD H, (IX+d)	LD IXH,A	LD IXL,B	LD IXL,C	LD IXL,D	LD IXL,E	LD IXL,IXH	LD IXL,IXL	LD L, (IX+d)	LD IXL,A
7	LD (IX+d),B	LD (IX+d),C	LD (IX+d),D	LD (IX+d),E	LD (IX+d),H	LD (IX+d),L		LD (IX+d),A					LD A,IXH	LD A,IXL	LD A, (IX+d)	
8					ADD A,IXH	ADD A,IXL	ADD A, (IX+d)						ADC A,IXH	ADC A,IXL	ADC A, (IX+d)	
9					SUB A,IXH	SUB A,IXL	SUB A, (IX+d)						SBC A,IXH	SBC A,IXL	SBC A, (IX+d)	
A					AND A,IXH	AND A,IXL	AND A, (IX+d)						XOR A,IXH	XOR A,IXL	XOR A, (IX+d)	
B					OR A,IXH	OR A,IXL	OR A, (IX+d)						CP A,IXH	CP A,IXL	CP A, (IX+d)	
C												Table 111				
D																
E		POP IX		EX (SP),IX		PUSH IX				JP (IX)						
F										LD SP,IX						

Note: n=8-bit datA; Mmn=16- or 24-bit addr or data; d=8-bit two’s-complement displacement.

Table 109. Opcode Map—Second Opcode After `0EDh`

Legend



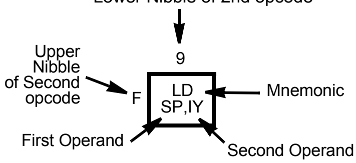
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	IN0 B,(n)	OUT0 (n),B	LEA BC ,IX+d	LEA BC ,IY+d	TST A,B			LD BC, (HL)	IN0 C,(n)	OUT0 (n),C			TST A,C			LD (HL), BC
1	IN0 D,(n)	OUT0 (n),D	LEA DE ,IX+d	LEA DE ,IY+d	TST A,D			LD DE, (HL)	IN0 E,(n)	OUT0 (n),E			TST A,E			LD(HL), DE
2	IN0 H,(n)	OUT0 (n),H	LEA HL ,IX+d	LEA HL ,IY+d	TST A,H			LD HL, (HL)	IN0 L,(n)	OUT0 (n),L			TST A,L			LD (HL), HL
3		LD IY, (HL)	LEA IX ,IX+d	LEA IY ,IY+d	TST A,(HL)			LD IX, (HL)	IN0 A,(n)	OUT0 (n),A			TST A,A		LD (HL),IY	LD (HL), IX
4	IN B,(BC)	OUT (BC),B	SBC HL,BC	LD (Mmn), BC	NEG	RETN	IM 0	LD I,A	IN C,(BC)	OUT (BC),C	ADC HL,BC	LD BC, (Mmn)	MLT BC	RETI		LD R,A
5	IN D,(BC)	OUT (BC),D	SBC HL,DE	LD (Mmn), DE	LEA IX ,IY+d	LEA IY ,IX+d	IM 1	LD A,I	IN E,(BC)	OUT (BC),E	ADC HL,DE	LD DE, (Mmn)	MLT DE		IM 2	LD A,R
6	IN H,(BC)	OUT (BC),H	SBC HL,HL	LD (Mmn), HL	TST A,n	PEA IX+d	PEA IY+d	RRD	IN L,(BC)	OUT (BC),L	ADC HL,HL	LD HL, (Mmn)	MLT HL	LD MB,A	LD A,MB	RLD
7			SBC HL,SP	LD (Mmn), SP	TSTIO n		SLP		IN A,(BC)	OUT (BC),A	ADC HL,SP	LD SP, (Mmn)	MLT SP	STMIX	RSMIX	
8			INIM	OTIM	INI2						INDM	OTDM	IND2			
9			INIMR	OTIMR	INI2R						INDMR	OTDMR	IND2R			
A	LDI	CPI	INI	OUTI	OUTI2				LDD	CPD	IND	OUTD	OUTD2			
B	LDIR	CPIR	INIR	OTIR	OTI2R				LDDR	CPDR	INDR	OTDR	OTD2R			
C			INIRX	OTIRX				LD I,HL			INDRX	OTDRX				
D								LD HL,I								
E																
F																

Note: n=8-bit data; Mmn=16- or 24-bit addr or data; d=8-bit two's-complement displacement.

Table 110. Opcode Map—Second Opcode After `0FDh`

Legend



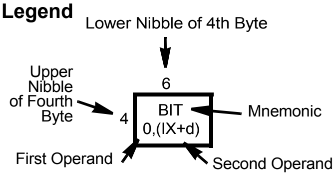
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								LD BC,(IY+d)		ADD IY,BC						LD (IY+d),BC
1								LD DE,(IY+d)		ADD IY,DE						LD (IY+d),DE
2		LD IY,Mmn	LD (Mmn),IY	INC IY	INC IYH	DEC IYH	LD IYH,n	LD HL,(IY+d)		ADD IY,IY	LD IY,(Mmn)	DEC IY	INC IYL	DEC IYL	LD IYL,n	LD (IY+d),HL
3		LD IX,(IY+d)			INC (IY+d)	DEC (IY+d)	LD (IY+d),n	LD IY,(IY+d)		ADD IY,SP					LD (IY+d),IX	LD (IY+d),IY
4					LD B,IYH	LD B,IYL	LD B,(IY+d)						LD C,IYH	LD C,IYL	LD C,(IY+d)	
5					LD D,IYH	LD D,IYL	LD D,(IY+d)						LD E,IYH	LD E,IYL	LD E,(IY+d)	
6	LD IYH,B	LD IYH,C	LD IYH,D	LD IYH,E	LD IYH,IYH	LD IYH,IYL	LD H,(IY+d)	LD IYH,A	LD IYL,B	LD IYL,C	LD IYL,D	LD IYL,E	LD IYL,IYH	LD IYL,IYL	LD L,(IY+d)	LD IYL,A
7	LD (IY+d),B	LD (IY+d),C	LD (IY+d),D	LD (IY+d),E	LD (IY+d),H	LD (IY+d),L		LD (IY+d),A					LD A,IYH	LD A,IYL	LD A,(IY+d)	
8					ADD A,IYH	ADD A,IYL	ADD A,(IY+d)						ADC A,IYH	ADC A,IYL	ADC A,(IY+d)	
9					SUB A,IYH	SUB A,IYL	SUB A,(IY+d)						SBC A,IYH	SBC A,IYL	SBC A,(IY+d)	
A					AND A,IYH	AND A,IYL	AND A,(IY+d)						XOR A,IYH	XOR A,IYL	XOR A,(IY+d)	
B					OR A,IYH	OR A,IYL	OR A,(IY+d)						CP A,IYH	CP A,IYL	CP A,(IY+d)	
C												Table 112				
D																
E		POP IY		EX (SP),IY		PUSH IY				JP (IY)						
F										LD SP,IY						

Note: n=8-bit data; Mmn=16- or 24-bit addr or data; d=8-bit two’s-complement displacement.

Table 111. Opcode Map—Fourth Byte After `0DDh`, `0CBh`, and `dd`

Legend



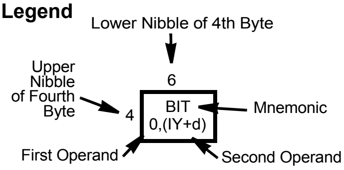
Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0							RLC (IX+d)								RRC (IX+d)	
1							RL (IX+d)								RR (IX+d)	
2							SLA (IX+d)								SRA (IX+d)	
3															SRL (IX+d)	
4							BIT 0, (IX+d)								BIT 1, (IX+d)	
5							BIT 2, (IX+d)								BIT 3, (IX+d)	
6							BIT 4, (IX+d)								BIT 5, (IX+d)	
7							BIT 6, (IX+d)								BIT 7, (IX+d)	
8							RES 0, (IX+d)								RES 1, (IX+d)	
9							RES 2, (IX+d)								RES 3, (IX+d)	
A							RES 4, (IX+d)								RES 5, (IX+d)	
B							RES 6, (IX+d)								RES 7, (IX+d)	
C							SET 0, (IX+d)								SET 1, (IX+d)	
D							SET 2, (IX+d)								SET 3, (IX+d)	
E							SET 4, (IX+d)								SET 5, (IX+d)	
F							SET 6, (IX+d)								SET 7, (IX+d)	

Note: d=8-bit two's-complement displacement.

Table 112. Opcode Map—Fourth Byte After `0FDh`, `0CBh`, and `dd`

Legend



Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0							RLC (IY+d)								RRC (IY+d)	
1							RL (IY+d)								RR (IY+d)	
2							SLA (IY+d)								SRA (IY+d)	
3															SRL (IY+d)	
4							BIT 0, (IY+d)								BIT 1, (IY+d)	
5							BIT 2, (IY+d)								BIT 3, (IY+d)	
6							BIT 4, (IY+d)								BIT 5, (IY+d)	
7							BIT 6, (IY+d)								BIT 7, (IY+d)	
8							RES 0, (IY+d)								RES 1, (IY+d)	
9							RES 2, (IY+d)								RES 3, (IY+d)	
A							RES 4, (IY+d)								RES 5, (IY+d)	
B							RES 6, (IY+d)								RES 7, (IY+d)	
C							SET 0, (IY+d)								SET 1, (IY+d)	
D							SET 2, (IY+d)								SET 3, (IY+d)	
E							SET 4, (IY+d)								SET 5, (IY+d)	
F							SET 6, (IY+d)								SET 7, (IY+d)	

Note: d=8-bit two's-complement displacement.

Glossary

absolute delay. The time interval or phase difference between transmission and reception of a signal.

acknowledge character (ACK). A transmission control character transmitted by the receiving station as an affirmative response to the sending station.

active device. A device that requires a source of energy for its operation and yields an output that is a function of present and past input signals. Examples of active devices include controlled power supplies, transistors, LEDs, amplifiers, and transmitters.

ADC. Analog-to-Digital Converter—a circuit that converts an analog signal to a digital bit stream. See A/D. Add with Carry; an arithmetic instruction.

ADD. Add without Carry; an arithmetic instruction.

Add/Subtract Flag. The Add/Subtract Flag is used by the decimal adjust accumulator instructions (**DAA**) to distinguish between **ADD** and **SUBTRACT** instructions.

added bit. A bit delivered to the intended destination user in addition to intended user information bits and delivered overhead bits. Also called *extra bit*.

added block. Any block, or other delimited bit group, delivered to the intended destination user in addition to intended user information bits and delivered overhead bits. Also called *extra block*.

ADDR. Address.

address lines. The direct signals that go from the CPU to other devices connected to the bus.

address space. The physical or logical area of the target system's memory map. The memory map could be physically partitioned into ROM to store code, and RAM for data. The memory can also be divided logically to form separate areas for code and data storage.

ADL. ADDRESS AND DATA LONG mode takes advantage of the eZ80® CPU's 16 MB linear addressing space, 24-bit CPU registers, and enhanced instruction set. Also called ADL MEMORY mode.

AFA, AFB. Automatic Frequency control output A and B.

AGND. Analog Ground.

Alternate register set. One of two banks of working registers in the eZ80® CPU. The alternate register set contains an 8-bit accumulator register (A') and six 8-bit working registers (B', C', D', E', H', and L'). These six 8-bit alternate working registers can also be combined to function as the multibyte register pairs BC', DE', and HL'. The 8-bit Flag register F' completes the alternate register set. See Main register set.

ALU. Arithmetic Logical Unit. The ALU is contained within the data block of the CPU. The ALU performs the arithmetic and logic functions on the addresses and the data passed over from the control block or from the CPU registers.

American National Standards Institute (ANSI). The U.S. standards organization that establishes procedures for the development and coordination of voluntary American National Standards.

ARAM. Audio-Quality RAM.

architecture. Of a computer, the physical configuration, logical structure, formats, protocols, and operational sequences for processing data, controlling the configuration, and controlling the operations. Computer architecture may also include word lengths, instruction codes, and the interrelationships among the main parts of a computer or group of computers.

Arithmetic Logical Unit (ALU). the element that can perform the basic data manipulations in the central processor. Usually, the ALU can add, subtract, complement, negate, rotate, AND, and OR.

array. An arrangement of elements in one or more dimensions. In a programming language, an aggregate that consists of data objects with identical attributes, each of which may be uniquely referenced by subscription.

AS. Address Strobe.

ASCII. Acronym for American Standard Code for Information Interchange. The standard code used for information interchange among data processing systems, data communications systems, and associated equipment in the United States.

ASYNC. Asynchronous Communication Protocol.

ASYNCA. Asynchronous Channel A.

ASYNCB. Asynchronous Channel B.

asynchronous. An event or device that is not synchronous with CPU timing (or other process).

Asynchronous Transfer Mode (ATM). A high-speed multiplexing and switching method utilizing fixed-length cells of 53 octets to support multiple types of traffic. ATM, specified in international standards, is asynchronous in the sense that cells carrying user data need not be periodic. A method used for transmitting voice, video, and data over high-speed local area networks.

asynchronous transmission. Data transmission in which the instant that each character, or block of characters, starts is arbitrary; once started, the time of occurrence of each signal representing a bit within the character, or block, has the same relationship to significant instants of a fixed time frame.

AT commands. A de facto standard for modem commands from an attached CPU, used in most 1,200 and 2,400 b/s modems.

AT command set. The character set used in modem control strings. Characters include the alphabet from A through T plus a special carriage return, <CR>.

ATE. Automatic Test Equipment.

ATM. See Asynchronous Transfer Mode.

AV_{DD}. Analog power.

baud. A unit of measure of transmission capacity. The speed at which a modem can transmit data. The number of events or signal changes that occur in one second. Because one event can encode more than one bit in high-speed digital communications, baud rate and bits per second are not always synonymous, especially at speeds above 2400 bps.

baud rate. A unit of measure of the number of state changes (from 0 to 1 or 1 to 0) per second on an asynchronous communications channel.

big-endian. A computer architecture in which, within a given multibyte numeric representation, the most significant byte contains the lowest address (the word is stored “big-end-first”).

binary (b). A number system based on A binary digit is a bit.

bit. *binary digit*—a digit of a binary system. It contains only two possible values: 0 or 1.

Bit Error Ratio (BER). The number of erroneous bits divided by the total number of bits transmitted, received, or processed over some stipulated period.

Bit Error Ratio Tester (BERT). A testing device that compares a received data pattern with a known transmitted pattern to determine the level of transmission quality.

bit configuration. The sequence of bits used to encode a character.

bit inversion. The changing of the state of a bit to the opposite state. The changing of the state that represents a given bit, i.e., a 0 or a 1, to the opposite state.

bit rate (BR). In a bit stream, the number of bits occurring per unit time, usually expressed in bits per second.

bit string. A sequence of bits. In a bit stream, individual bit strings may be separated by data delimiters.

bps. bits per second—the number of binary digits transmitted every second during a data-transfer procedure.

BR. See Bit Rate.

BRG. Baud Rate Generator.

BUSACK. Bus Acknowledge—a message granting permission for a device to transfer data on the bus.

BUSREQ. Bus Request—a message from a device requesting permission to transfer data on the bus.

byte (B). A sequence of adjacent bits (usually 8) considered as a unit. A collection of four sequential bits of memory. Two sequential bytes (8 bits) comprise one word.

C. Carry.

CALL. Call Subroutine; a program control instruction.

CALL cc. Conditional Call Subroutine; a program control instruction.

Carry Flag. The Carry Flag bit is set or reset depending on the operation performed, such as an **ADD**, which can generate a carry, or a **SUBTRACT**, which can generate a borrow.

CCF. Clear Carry Flag. Complement Carry Flag; a processor control instruction.

CE. Chip Enable.

C. See Carry Flag.

check bit. A bit, such as a parity bit, derived from and appended to a bit string for later use in error detection and possibly error correction.

checksum. A field of one or more bytes appended to a block of *n* words that contain a truncated binary sum formed from the contents of that block. The sum is used to verify the integrity of data in a ROM or on a tape.

CIEF. Clear IE Flag.

clock rate. The rate at which a clock issues timing pulses.

CLR. Clear.

comparator. In analog computing, a functional unit that compares two analog variables and indicates the result of that comparison. A device that compares two items of data and indicates the result of that comparison. A device for determining the dissimilarity of two items such as two pulse patterns or words.

CP. Compare with Accumulator; an arithmetic instruction.

CPD (CPDR). Compare and Decrement (with Repeat); a compare instruction.

CPI (CPIR). Compare and Increment (with Repeat); a compare instruction.

CPL. Complement Accumulator; a logical instruction.

CTR. Counter.

CTRL. Control.

D. Decimal-Adjust Flag.

D/A. Digital-to-Analog—the conversion of a digital signal to an analog signal. See DAC.

DAA. Decimal Adjust Accumulator; an arithmetic instruction.

DAC. Digital-to-Analog Converter. A circuit that converts a digital bit stream (binary numbers) into voltage signals at specific levels. See D/A.

DART. Dual-Channel Asynchronous Receiver/Transmitter—an SIO that supports asynchronous data communications only.

data bus. An I/O bus used by the eZ80® CPU for passing data to and from internal and external memory.

DCE. Data Circuit-terminating Equipment—connects data terminal equipment (DTE) to a data circuit. A modem is an example of a DCE.

DEC. Decrement; an arithmetic instruction.

DECW. Decrement Word.

DI. Disable Interrupt; a processor control instruction.

digital phase-locked loop. A phase-locked loop in which the reference signal, the controlled signal, or the controlling signal, or any combination of these, is in digital form.

DI. Disable interrupt.

DJNZ. Decrement and Jump if Nonzero; a program control instruction.

DMA. Direct Memory Access—a device that is dedicated to the task of controlling high-speed block transfers of data independently of the CPU.

downconverter. A device that translates frequencies from higher to lower frequencies.

DPLL. Digital Phase Locked Loop.

DR. Data Read.

DRAM. Dynamic Random Access Memory—a computer memory that requires a refresh signal to be sent to it periodically. Most computers use DRAM chips for memory. Contrast to static RAM (SRAM).

DREQ. Data Request. DMA Request.

DSR. Data Set Ready.

DSR signal. Data Set Ready signal.

DST. Destination.

DTE. Data Terminal Equipment. Equipment that sends and receives data.

DTR. Data Transfer Rate.

EC. Enable Clock.

EPROM. Erasable Programmable Read-Only Memory.

EI. Enable Interrupt; a processor control instruction (also see IE).

EPM. EPROM Program Mode.

EQ. A Boolean operator meaning Equal to.

EX. Exchange Registers; an exchange instruction.

EXX. Exchange CPU Multibyte Register Banks; an exchange instruction.

exception. An error, unusual condition, or external signal that can set a status bit. It may or may not cause an interrupt, depending on whether or not the corresponding interrupt is enabled.

EXTAL. External clock/crystal.

eZ80®. ZiLOG's next-generation Internet processor core. A single-cycle instruction fetch machine that is four times faster than ZiLOG's original Z80, offering linear addressing that can address up to 16MB of memory.

Fetch. The act of retrieving information (instructions or data) from memory.

glitch. A pulse or burst of noise; sometimes reserved for the more annoying types of noise pulses that cause crashes and failures.

GPIO. General Purpose Input/Output.

GPR. General Purpose Register.

H. See Half-Carry Flag.

h. See Hexadecimal.

Half-Carry Flag. The Half-Carry Flag is set or reset, depending on the carry and borrow status between bits 3 and 4 of an 8-bit arithmetic operation. This flag is used by the decimal adjust accumulator instruction (**DAA**) to correct the result of a packed BCD addition or subtraction.

HALT. HALT mode; a processor control instruction.

high-pass filter. A filter that passes frequencies above a given frequency and attenuates all others.

I^2C. The Inter-Integrated Circuit serial bus developed by Phillips International for interconnecting devices within electronics, telecommunications, and industrial consumer electronic products.

ICE. In-Circuit Emulator. A ZiLOG product that supports the application design process.

IE. Interrupt Enable.

IEF1 and IEF2. See Interrupt Enable Flag.

IEI. Interrupt Enable IN.

IEO. Interrupt Enable OUT.

IIHX. Intel Hexadecimal format.

IllOp. Illegal Operation.

IM. Immediate Data Addressing Mode. Interrupt Mode; a processor control instruction. On the eZ80® CPU, there are 3 interrupt mode instructions: IM0, IM1, and IM2.

IMASK. Interrupt mask register.

IMR. Interrupt Mask Register.

IN. Input from I/O; an input/output instruction.

INC. Increment; an arithmetic instruction.

INCW. Increment Word.

IND. Input from I/O and Decrement; an input/output instruction.

Index registers (IX and IY). The multibyte registers IX and IY allow standard addressing and relative displacement addressing in memory.

INDM. Input from I/O and Decrement; an input/output instruction.

INDMR. Input from I/O and Decrement with Repeat; an input/output instruction.

INDR. Input from I/O and Decrement with Repeat; an input/output instruction.

IND2. Input from I/O and Decrement; an input/output instruction.

IND2R. Input from I/O and Decrement with Repeat; an input/output instruction.

INI. Input from I/O and Increment; an input/output instruction.

INIM. Input from I/O and Increment; an input/output instruction.

INIMR. Input from I/O and Increment with Repeat; an input/output instruction.

INIR. Input from I/O and Increment with Repeat; an input/output instruction.

INI2. Input from I/O and Increment; an input/output instruction.

INI2R. Input from I/O and Increment with Repeat; an input/output instruction.

INT. Interrupt.

INTACK. Interrupt Acknowledge.

Internet Control Message Protocol. An Internet protocol that reports datagram delivery errors. ICMP is a key part of the TCP/IP protocol suite. The packet internet gopher (ping) application is based on ICMP.

Internet protocol (IP). A DOD standard protocol designed for use in interconnected systems of packet-switched computer communication networks.

interrupt. A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. The three types of interrupts include: internal hardware, external hardware, and software.

interrupt acknowledge cycle. The time required for the eZ80® CPU to respond to an interrupt service request.

Interrupt Enable Flag. In the eZ80® CPU, there are two interrupt enable flags, IEF1 and IEF2, that are set or reset using the Enable Interrupt (**EI**) and Disable Interrupt (**DI**) instructions.

Interrupt Page Address register (I). The 8-bit I register stores the upper 8 bits of the interrupt vector table address for Mode 2 vectored interrupts.

interrupt request (IRQ). Hardware lines that carry a signal from a device to the processor.

Interrupt Service Routine. An interrupt service routine can affect the exchange of data, status information, or control information between the CPU and an interrupting peripheral.

interrupt vector address. The address used by the eZ80® CPU as the starting point for the associated interrupt service routine.

IN0. Input from I/O on Page 0; an input/output instruction.

IOCS. Auxiliary Chip Select Output Signal.

IORQ. I/O Request.

IP. Internet Protocol.

IPR. Interrupt Priority Register.

IRQ. Interrupt Request.

ISR. See Interrupt Service Routine.

IVEDCT bus. An internal 8-bit bus used by on-chip peripherals for passing an interrupt vector address byte to the eZ80® CPU.

JP. Jump; a program control instruction.

JP cc. Conditional Jump; a program control instruction.

JR. Jump Relative; a program control instruction.

JR cc. Conditional Jump Relative; a program control instruction.

Latch. A hardware service that senses information and holds it until reset.

LD. Load; an arithmetic instruction.

LDD (LDDR). Load and Decrement (with Repeat); a block transfer instruction.

LDI (LDIR). Load and Increment (with Repeat); a block transfer instruction.

LEA. Load Effective Address; a load instruction.

little-endian. A computer architecture in which, within a given 16- or 32-bit word, bytes at lower addresses bear lower significance (the word is stored “little-end-first”). The PDP-11 and VAX families of computers and a lot of communications and networking hardware are little-endian.

low-pass filter. A filter network that passes all frequencies below a specified frequency with little or no loss, but strongly attenuates higher frequencies.

lsb. least significant bit.

LSB. Least Significant Byte.

MAC. MAC An acronym for Media Access Control, the method a computer uses to transmit or receive data across a LAN.

MADL. See Mixed-ADL mode.

Main register set. One of two banks of working registers in the eZ80® CPU. The main register set contains the 8-bit accumulator register (A) and six 8-bit working registers (B, C, D, E, H, and L). See Alternate register set.

Maskable Interrupt. Maskable interrupts can be enabled and disabled. If enabled, the eZ80® CPU will respond to a maskable interrupt service request from an external device or on-chip peripheral. If disabled, the eZ80® CPU will not respond to an maskable interrupt service request from an external device or on-chip peripheral. A maskable interrupt can be disabled by the programmer. See nonmaskable interrupt.

MBASE. Z80 Memory Mode Base Address Register. The 8-bit MBASE register determines the page of memory currently employed when operating in Z80 mode. The MBASE register is *only* used during Z80 mode. However, the MBASE register can *only* be altered from within ADL mode.

MIE. Master Interrupt Enable.

Mixed-ADL mode (MADL). The MADL control bit is used to indicate whether or not a program contains code that runs in both ADL and Z80 MEMORY modes. Also Mixed-Memory Mode Flag.

MLT. Multiply; an arithmetic instruction.

msb. most significant bit.

MSB. Most Significant Byte.

multiplexing (MUXing). The combining of two or more information channels onto a common transmission medium. In electrical communications, the two basic forms of multiplexing are time-division multiplexing (TDM) and frequency-division multiplexing (FDM).

MUX. Multiplexer.

N. See Add/Subtract Flag.

NACK. See Negative Acknowledge Character.

NC. No Carry Flag.

DJNZ. Decrement and Jump if Not Zero.

NEG. Negate Accumulator; an arithmetic instruction.

Negative-Acknowledge Character (NACK). A transmission control character sent by a station as a negative response to the station with which the connection has been set up. In binary synchronous communication protocol, the NAK is used to indicate that an error was detected in the previously received block and that the receiver is ready to accept retransmission of that block.

NMI. Nonmaskable interrupt.

Nonmaskable Interrupt. Nonmaskable interrupts are always enabled. The eZ80® CPU will always respond to a nonmaskable interrupt service request from an external device or on-chip peripheral. A nonmaskable interrupt cannot be disabled by the programmer. See maskable interrupt.

NOP. An acronym for No Operation, an instruction whose sole function is to increment the program counter, but that does not affect any changes to registers or memory.

NORMAL mode. Mode of operation without error correction active.

NRZ. NonReturn to Zero.

NRZI. NonReturn to Zero Inverted.

OE. Output Enable.

Opcode. Operation Code, a quantity that is altered by a microcontroller's instruction.

Opcode suffix. Opcode suffixes are additions to an instruction set that assist with memory mode switching operations. These suffixes are appended to many instructions to indicate that a memory mode change or an exception to a standard memory mode operation is being requested. There are 4 individual suffixes available for use on the eZ80®: **.SIS**, **.SIL**, **.LIS**, and **.LIL**.

Open-Drain.

Operand. The data unit that is operated on; usually identified by an address in an instruction. For example, in "add 100 to 400", 100 and 400 are operands.

OR. Logical OR; a logical instruction.

OTDM (OTDMR). Output to I/O and Decrement (with Repeat); an input/output instruction.

OTIM (OTIMR). Output to I/O and Increment (with Repeat); an input/output instruction.

OUT. Output to I/O; an input/output instruction.

OUTD (OTDR). Output to I/O and Decrement (with Repeat); an input/output instruction.

OUTD2 (OTD2R). Output to I/O and Decrement (with Repeat); an input/output instruction.

OUTI (OTIR). Output to I/O and Increment (with Repeat); an input/output instruction.

OUTI2 (OTI2R). Output to I/O and Increment (with Repeat); an input/output instruction.

OUT0. Output to I/0 on Page 0; an input/output instruction.

PARC. Parallel Controls Register.

Parity Bit. An extra binary bit attached to each byte of synchronous data allowing detection of transmission errors.

Parity/overflow flag. The parity/overflow flag is set or reset depending on the operation performed. For arithmetic operations, this flag indicates an overflow condition when the result in the accumulator is greater than the maximum possible number (+127) or is less than the minimum possible number (–128).

PARM. Parallel Mode Register.

PC. Program Counter (see Program Counter register).

PEA. Push Effective Address; a load instruction.

Persistent mode. One of two types of mode changes available to the eZ80®. Persistent mode switches allow the eZ80® to operate for long periods in ADL mode, then switch to Z80 mode to run a section of Z80 code, and then return to ADL mode. See single-instruction mode.

phase-locked loop (PLL). A special analog circuit that controls an oscillator so that it maintains a constant phase angle relative to a reference signal.

PHI. System Clock.

PLC. Production Languages Corporation.

POP. Retrieve a Value from the Stack; A load instruction.

POR. Power-On Reset.

PRE. Prescaler.

Prefetch. The act of retrieving information (instructions or data) from memory in advance of their intended use. Pipelined CPUs use a prefetch to gather instructions and data that will be required by the CPU for future operations.

Program Counter register. The multibyte Program Counter register stores the address of the current instruction being fetched from memory.

PUSH. To store a value in the stack; a load instruction.

P/V. See Parity/Overflow Flag.

PWM. Pulse Width Modulator. In digital audio and video systems, the representation of an analog signal by its direct digitized values.

QAM. Quadrature Amplitude Modulation. Symbols are represented by a combination of signal amplitude and phase. QAM is used in modems that are compliant with V.22bis and higher. Sometimes pronounced *kwam*.

RDYE. Data Ready.

Refresh Counter Register (R). The Refresh Counter Register (R) contains a count of executed instruction fetch cycles. The 7 least significant bits (lsb) of the R register are automatically incremented after each instruction fetch. The most significant bit (msb) can only be changed by writing to the R register. The R register can be read from and written to using the dedicated instructions **LD A,R** and **LD R,A**, respectively.

REQ. Request.

RES. Reset Bit; a bit manipulation instruction.

RET. Return from subroutine; a program control instruction.

RET cc. Conditional Return; a program control instruction.

RETI. Return from Interrupt; a program control instruction.

RETN. Return from nonmaskable interrupt; a program control instruction.

RL. ROMless Control. Rotate Left; a rotate instruction.

RLA. Rotate Left–Accumulator; a rotate instruction.

RLC. Rotate Left Circular; a rotate instruction.

RLCA. Rotate Left Circular–Accumulator; a rotate instruction.

RLD. Rotate Left Decimal; a rotate instruction.

ROM. See Read-Only Memory.

ROMCS. ROM Chip Select.

ROMless. No Read-Only Memory. External memory is required.

RP. Register Pointer.

RR. Read Register. Rotate Right; a rotate instruction.

RRA. Rotate Right–Accumulator; a shift instruction.

RRC. Rotate Right Circular; a shift instruction.

RRCA. Rotate Right Circular–Accumulator; a shift instruction.

RRD. Rotate Right Decimal; a shift instruction.

RSMIX. Reset MIXED MEMORY Mode Flag; a processor control instruction.

RST. Restart; a program control instruction. as opposed to RES or RESET.

S. See Sign Flag.

SBC. Subtract with Carry; an arithmetic instruction.

SCF. Set Carry Flag; a processor control instruction.

SCLK. System Clock.

SET. Set Bit; a bit manipulation instruction.

SIEF. Set IE Flag.

Sign Flag. The Sign flag stores the state of the most significant bit of the accumulator (bit 7).

Single-instruction mode. One of two types of mode changes available to the eZ80®. Single-instruction mode changes allow certain instructions to operate using either ADL or Z80 addressing mode without making a persistent change to either mode. See persistent mode.

SLA. Shift Left; a shift instruction.

SLL. Shift Left Logical.

SLP. Sleep; a processor control instruction.

SMR. Stop-Mode Recovery.

SP. Stack Pointer.

SPH. Stack Pointer High.

SPL. Stack Pointer Low. The 24-bit Stack Pointer Long register.

SPS. The 16-bit Stack Pointer Short register.

SR. Shift Right.

SRA. Shift Right Arithmetic; a shift instruction.

SRL. Shift Right Logical; a shift instruction.

STMIX. Set Mixed-ADL mode flag; a processor control instruction.

SUB. Subtract without Carry; an arithmetic instruction.

T_A. Ambient Temperature.

TpC. External Clock Cycle.

tristate. A form of transistor-to-transistor logic in which output stages, or input and output stages, can assume three states. Two are normal low-impedance 1 and 0 states; the third is a high-impedance state that allows many tristate devices to time-share bus lines. This industry term is not trademarked, and is available for ZiLOG use. Do not use *3-state* or *three-state*.

TST. Test Accumulator; a logical instruction.

TSTIO. Test I/O; an input/output instruction.

TX. Abbreviation for transmitter, transmit.

UART. Universal Asynchronous Receiver/Transmitter—a component or functional block that manages asynchronous communications. A UART converts data from the parallel format in which it is stored to a serial format for transmission.

upconverter. A device that translates frequencies from lower to higher frequencies.

USART. Universal Synchronous/Asynchronous Receiver/Transmitter. Can manage synchronous and asynchronous transmissions.

VBO. Voltage Brown-Out.

V_{CC}. Supply voltage.

V_{PP}. Programmed Voltage.

V_{REF}. Analog reference voltage.

WAIT state. A clock cycle during which no instructions are executed because the processor is waiting for data from memory.

WDT. Watch-Dog Timer. A timer that, when enabled under normal operating conditions, must be reset within the time period set within the application (WDTMR (1,0)). If the timer is not reset, a Power-on Reset occurs. Some older manuals refer to this timer as the WDTMR.

WDTOUT. Watch-Dog Timer output.

X_{IN}. Crystal Input.

XOR. Logical Exclusive OR; a logical instruction.

X_{OUT}. Crystal Output.

XTAL. Internal clock/crystal.

XTAL1. See X_{IN}.

XTAL2. See X_{OUT}.

ZDS. ZiLOG Developer Studio. ZiLOG's program development environment for Windows 95/98/NT.

Zero Flag. For 8-bit arithmetic and logical operations, the Z Flag is set to 1 if the resulting byte in the accumulator is 0. If the byte is not 0, the Z Flag is reset to 0.

Z80 Mode. The Z80-compatible addressing mode of ZiLOG's eZ80® CPU. Also called Z80 MEMORY mode.

Index

Numerics

A

E

F

H

I

M

R

S

Customer Support

If you experience any problems while operating this product, please check the ZiLOG Knowledge Base:

http://kb.zilog.com/kb/oKBmain.asp

If you cannot find an answer or have further questions, please see the ZiLOG Technical Support web page:

http://support.zilog.com