Z80 Family

CPU User Manual

User Manual

UM008005-0205

Z80 CPU
User's Manual

Revision History

Each instance in Table 1 reflects a change to this document from its previous revision. To see more detail, click the appropriate link in the table.

Table 1. Revision History of this Document

Date	Revision Level	Section	Description	Page #
December 2004	04	Z80 Instruction Set	Corrected discrepancies in the bit patterns for IM 0, IM 1 and IM 2 instructions.	176,177, 178
February 2005	05	Z80 Instruction Set, CPU Instruction Description	Corrected illustration for the Rotate and Shift Group RLCA instruction. Also corrected the hex code for the RLCA instruction on page 63.	190, 63

Table of Contents

List of Instructions

List of Figures

List of Tables

Manual Objectives

This user manual describes the architecture and instruction set of the Z80 CPU.

About This Manual

ZiLOG recommends that the user read and understand everything in this manual before setting up and using the product. However, we recognize that users have different styles of learning: some will want to set up and use their new evaluation kit while they read about it; others will open these pages only to check on a particular specification. Therefore, we have designed this manual to be used either as a *how to* procedural manual or a reference guide to important data.

Intended Audience

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

Manual Organization

The Z80 CPU User's Manual is divided into four chapters.

Overview

Presents an overview of the User's Manual Architecture, Pin descriptions, timing and Interrupt Response.

Hardware and Software Implementation

Presents examples of the User's Manual hardware and software.

Z80 CPU Instruction Description

Presents the User's Manual instruction types, addressing modes and instruction Op Codes.

Z80 Instruction Set

Presents an overview of the User's Manual assenbly language, status indicator flags and the Z80 instructions.

Related Documents

Part Number	Title	DC number
Part Number	Title	DC number
Part Number	Title	DC number

Manual Conventions

The following assumptions and conventions are adopted to provide clarity and ease of use:

Use of the Words *Set* and *Clear*

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

Notation for Bits and Similar Registers

A field of bits within a register is designated as: Register (*n–n*). For example: PWM_CR (31–20). A field of bits within a bus is designated as: Bus_{n-n}. For example: $PCntl_{7-4}$. A range of similar (whole) registers is designated as: Register*n*–Register*n*. For example: OPBCS5–OPBCS0.

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.

Courier Font

Commands, code lines and fragments, register (and other) mnemonics, values, equations, and various executable items are distinguished from general text by the use of the Courier font. This convention is not used within tables. For example: The STP bit in the CNTR register must be 1. Where the use of the font is not possible, as in the Index, the name of the entity is presented in upper case.

Hexadecimal Values Designated by H

Hexadecimal values are designated by a uppercase *H* and appear in the Courier typeface. For example: STAT is set to F8H.

Use of All Uppercase Letters

The use of all uppercase letters designates the names of states and commands. For example: The receiver can force the SCL line to Low to force the transmitter into a WAIT state. The bus is considered BUSY after the Start condition. A START command triggers the processing of the initialization sequence.

Use of Initial Uppercase Letters

Initial uppercase letters designate settings, modes, and conditions in general text. For example: The Slave receiver leaves the data line High. In Transmit mode, the byte is sent most significant bit first. The Master can generate a Stop condition to abort the transfer.

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

Trademarks

Z80, Z180, Z380 and Z80382 are trademarks of ZiLOG, Inc.

Overview

ARCHITECTURE

The ZiLOG Z80 CPU family of components are fourth-generation enhanced microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The speed offerings from 6–20 MHz suit a wide range of applications which migrate software. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers.

The Z80 CPU also contains a Stack Pointer, Program Counter, two index registers, a REFRESH register, and an INTERRUPT register. The CPU is easy to incorporate into a system since it requires only a single +5V power source. All output signals are fully decoded and timed to control standard memory or peripheral circuits; the Z80 CPU is supported by an extensive family of peripheral controllers.

Figure 1 illustrates the internal architecture and major elements of the Z80 CPU.
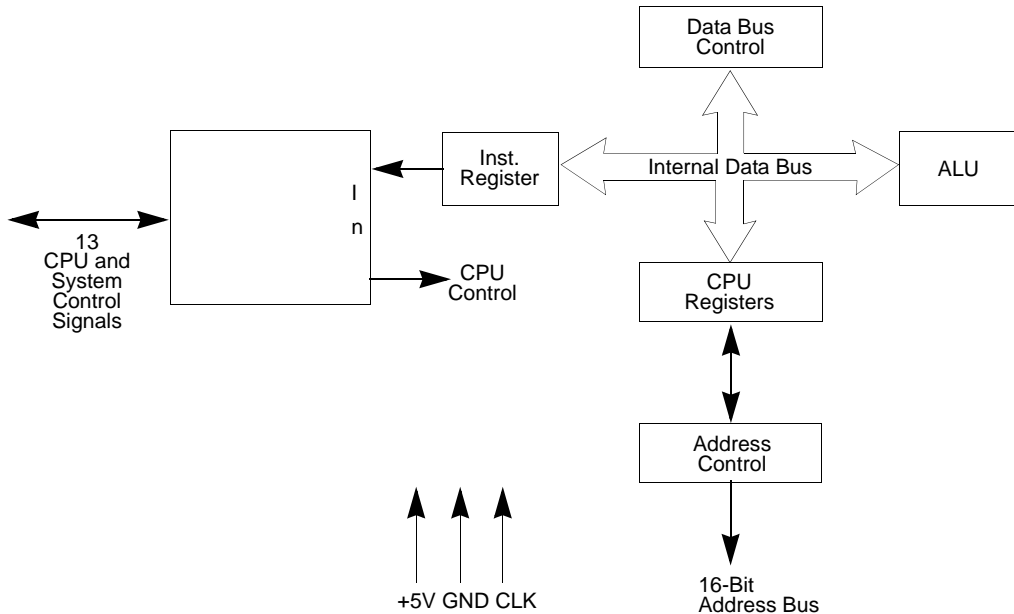
Figure 1.   Z80 CPU Block Diagram

CPU Registers

The Z80 CPU contains 208 bits of R/W memory that are available to the programmer. Figure 2 illustrates how this memory is configured to eighteen 8-bit registers and four 16-bit registers. All Z80 registers are implemented using static RAM. The registers include two sets of six general-purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers and six special-purpose registers.

Main Register Set   Alternate Register Set

Accumulator	Flags	Accumulator	Flags
A	F	A '	F '
B	C	B '	B '
D	E	D '	E '
H	L	H '	L '

General Purpose Registers

Interrupt Vector I	Memory Refresh R
Index Register	IX
Index Register	IY
Stack Pointer	SP
Program Counter	PC

Special Purpose Registers
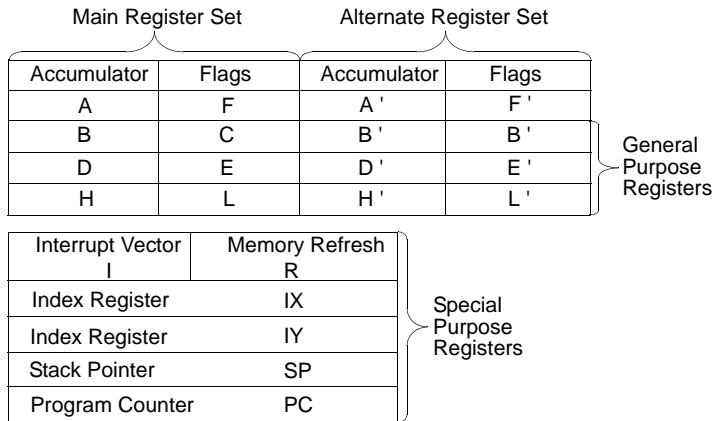
Figure 2.   Z80 CPU Register Configuration

Special-Purpose Registers

Program Counter (PC)

The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs, the new value is automatically placed in the PC, overriding the incrementer.

Stack Pointer (SP)

The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack to specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts,

unlimited subroutine nesting and simplification of many types of data manipulation.

Two Index Registers (IX and IY)

The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

Interrupt Page Address Register (I)

The Z80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I register is used for this purpose and stores the high order eight bits of the indirect address while the interrupting device provides the lower eight bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with minimal access time to the routine.

Memory Refresh Register (R)

The Z80 CPU contains a memory refresh counter, enabling dynamic memories to be used with the same ease as static memories. Seven bits of this 8-bit register are automatically incremented after each instruction fetch. The eighth bit remains as programmed, resulting from an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is transparent to the programmer and does not slow the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper eight bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the FLAG register indicates specific conditions for 8-bit or 1 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair with a single exchange instruction so that it is possible to work with either pair.

General Purpose Registers

Two matched sets of general-purpose registers, each set containing six 8-bit registers, may be used individually as 8-bit registers or as 16-bit register pairs. One set is called BC, DE, and HL while the complementary set is called BC', DE', and HL'. At any one time, the programmer can select either set of registers to work through a single exchange command for the entire set. In systems that require fast interrupt response, one set of general-purpose registers and an ACCUMULATOR/FLAG register may be reserved for handling this fast routine. One exchange command is executed to switch routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general-purpose registers are used for a wide range of applications. They also simplify programing, specifically in ROM-based systems where little external read/write memory is available.

Arithmetic Logic Unit (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally, the ALU communicates with the registers and the external data bus by using the internal data bus. Functions performed by the ALU include:

- Add

- Subtract

- Logical AND

- Logical OR

- Logical Exclusive OR

- Compare

- Left or Right Shifts or Rotates (Arithmetic and Logical)

- Increment

- Decrement

- Set Bit

- Reset Bit

- Test bit

Instruction Register and CPU Control

As each instruction is fetched from memory, it is placed in the INSTRUCTION register and decoded. The control sections performs this function and then generates and supplies the control signals necessary to read or write data from or to the registers, control the ALU, and provide required external control signals.

PIN DESCRIPTION

Overview

The Z80 CPU I/O pins are illustrated in Figure 3 and the function of each is described in the following paragraphs.
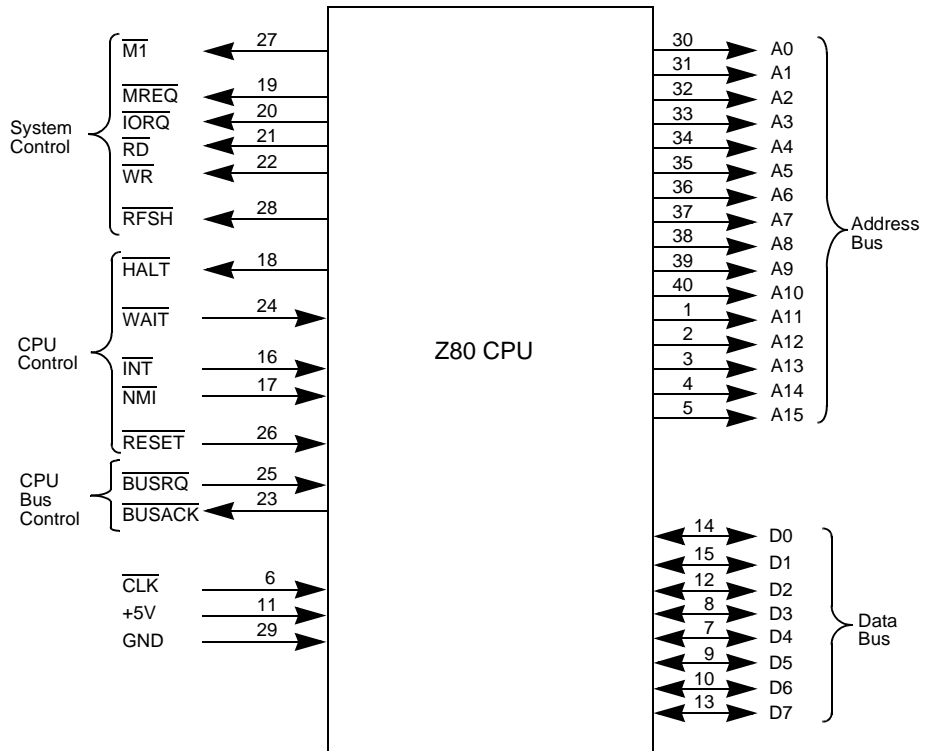
Figure 3.   Z80 I/O Pin Configuration

Pin Functions

A15–A0

Address Bus (output, active High, tristate). A15-A0 form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64 Kbytes) and for I/O device exchanges.

BUSACK

Bus Acknowledge (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$ $\overline{\text{RD}}$, and $\overline{\text{WR}}$ have entered their high-impedance states. The external circuitry can now control these lines.

BUSREQ

Bus Request (input, active Low). Bus Request has a higher priority than $\overline{\text{NMI}}$ and is always recognized at the end of the current machine cycle. $\overline{\text{BUSREQ}}$ forces the CPU address bus, data bus, and control signals $\overline{\text{MREQ}}$ $\overline{\text{IORQ}}$, RD, and WR to go to a high-impedance state so that other devices can control these lines. $\overline{\text{BUSREQ}}$ is normally wired-OR and requires an external pull-up for these applications. Extended $\overline{\text{BUSREQ}}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMS.

D7–D0

Data Bus (input/output, active High, tristate). D7–D0 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

HALT

HALT State (output, active Low). $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT instruction and is waiting for either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. During HALT, the CPU executes NOPs to maintain memory refresh.

INT

Interrupt Request (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{\text{INT}}$ is normally wired-OR and requires an external pull-up for these applications.

IORQ

Input/Output Request (output, active Low, tristate). $\overline{\text{IORQ}}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{\text{IORQ}}$ is also generated concurrently with $\overline{\text{M1}}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{\text{M1}}$

Machine Cycle One (output, active Low). $\overline{\text{M1}}$, together with $\overline{\text{MREQ}}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{\text{M1}}$ together with $\overline{\text{IORQ}}$, indicates an interrupt acknowledge cycle.

MREQ

Memory Request (output, active Low, tristate). $\overline{\text{MREQ}}$ indicates that the address bus holds a valid address for a memory read of memory write operation.

NMI

Non-Maskable Interrupt (input, negative edge-triggered). $\overline{\text{NMI}}$ has a higher priority than $\overline{\text{INT}}$. $\overline{\text{NMI}}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location `0066H`.

RD

Read (output, active Low, tristate). $\overline{\text{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

RESET

Reset (input, active Low). $\overline{\text{RESET}}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and registers I and R, and sets the

interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Notice that $\overline{\text{RESET}}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

RFSH

Refresh (output, active Low). $\overline{\text{RFSH}}$, together with $\overline{\text{MREQ}}$ indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

WAIT

WAIT (input, active Low). $\overline{\text{WAIT}}$ communicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a $\overline{\text{WAIT}}$ state as long as this signal is active. Extended $\overline{\text{WAIT}}$ periods can prevent the CPU from properly refreshing dynamic memory.

WR

Write (output, active Low, tristate). $\overline{\text{WR}}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

CLK

Clock (input). Single-phase MOS-level clock.

TIMING

Overview

The Z80 CPU executes instructions by stepping through a precise set of basic operations. These include:

- Memory Read or Write

- I/O Device Read or Write

- Interrupt Acknowledge

All instructions are series of basic operations. Each of these operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The clock periods are referred to as T (time) cycles and the operations are referred to as M (machine) cycles. Figure 4 illustrates how a typical instruction is series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2, and M3). The first machine cycle of any instruction is a fetch cycle which is four, five, or six T cycles long (unless lengthened by the WAIT signal, which is described in the next section). The fetch cycle (M1) is used to fetch the opcode of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices, and they may have anywhere from three to five T cycles (again, they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles.

During T2 and every subsequent Tw, the CPU samples the WAIT line with the falling edge of Clock. If the WAIT line is active at this time, another WAIT state is entered during the following cycle. Using this technique, the read can be lengthened to match the access time of any type of memory device.
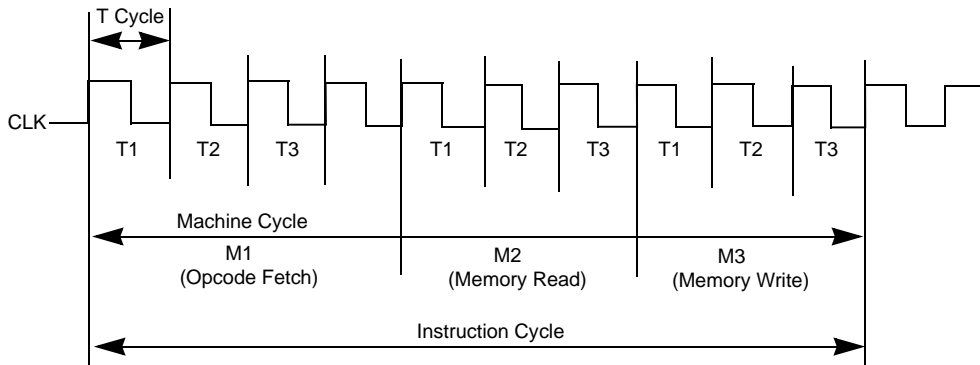
Figure 4.   Basic CPU Timing Example

Instruction Fetch

Figure 5 depicts the timing during an M1 (opcode fetch) cycle. The PC is placed on the address bus at the beginning of the M1 cycle. One half clock cycle later the $\overline{\text{MREQ}}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{\text{MREQ}}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{\text{RD}}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{\text{RD}}$ and $\overline{\text{MREQ}}$ signals. Thus, the data has already been sampled by the CPU before the $\overline{\text{RD}}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time.

During T3 and T4, the lower seven bits of the address bus contain a memory refresh address and the $\overline{\text{RFSH}}$ signal becomes active tindicating that a refresh read of all dynamic memories must be accomplished. An $\overline{\text{RD}}$ signal is not generated during refresh time to prevent data from different memory

segments from being gated onto the data bus. The $\overline{\text{MREQ}}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself because the refresh address is only guaranteed to be stable during $\overline{\text{MREQ}}$ time.
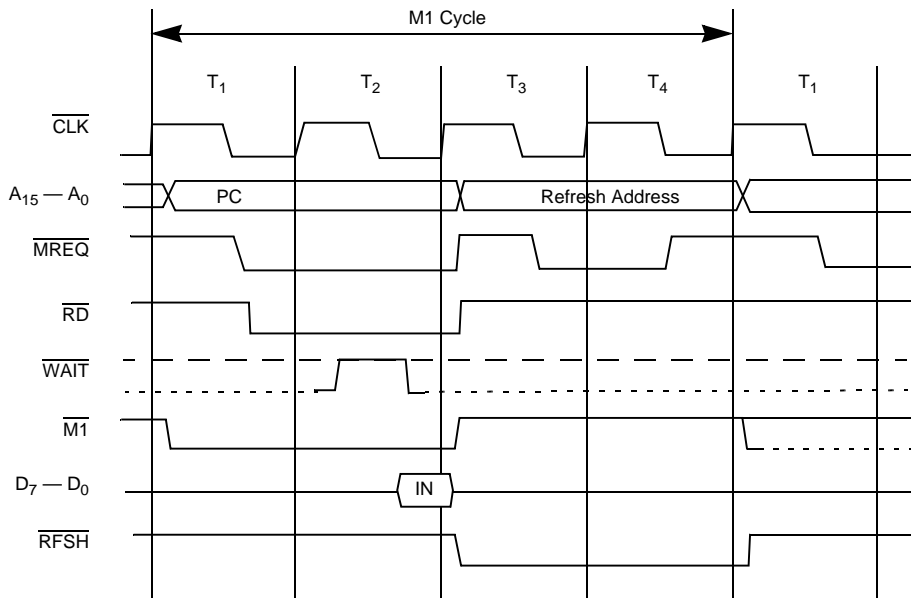
Figure 5.   Instruction Op Code Fetch

Memory Read Or Write

Figure 6 illustrates the timing of memory read or write cycles other than an Op Code fetch cycle. These cycles are generally three clock periods long unless wait states are requested by the memory through the $\overline{\text{WAIT}}$ signal. The $\overline{\text{MREQ}}$ signal and the $\overline{\text{RD}}$ signal are used the same as in the fetch cycle. In a memory write cycle, the $\overline{\text{MREQ}}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{\text{WR}}$ line is active when data on the data bus is stable so that

it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore, the $\overline{WR}$ signal goes inactive one-half T state before the address and data bus contents are changed so that the overlap requirements for almost any type of semiconductor memory type is met.
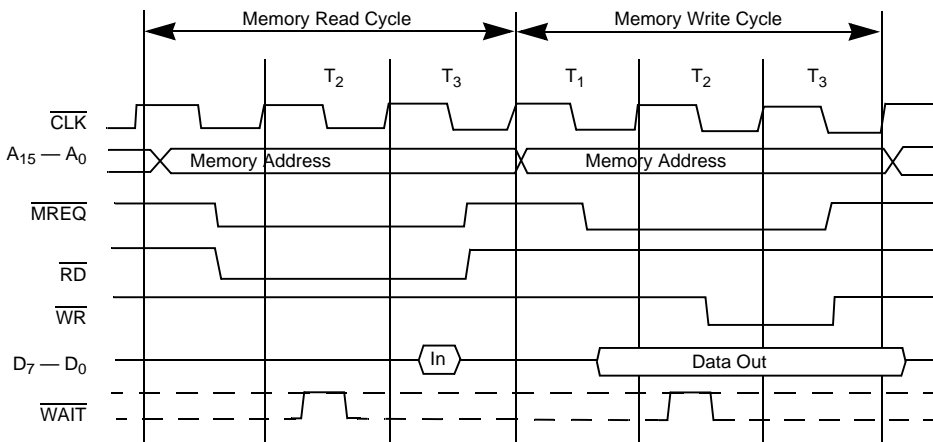
Figure 6.   Memory Read or Write Cycle

Input or Output Cycles

Figure 7 illustrates an I/O read or I/O write operation. During I/O operations a single wait state is automatically inserted. The reason is that during I/O operations, the time from when the $\overline{IORQ}$ signal goes active until the CPU must sample the $\overline{WAIT}$ line is very short. Without this extra state, sufficient time does not exist for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required. Also, without this wait state, it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time, the $\overline{WAIT}$ request signal is sampled.

During a read I/O operation, the $\overline{RD}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{WR}$ line is used as a clock to the I/O port.
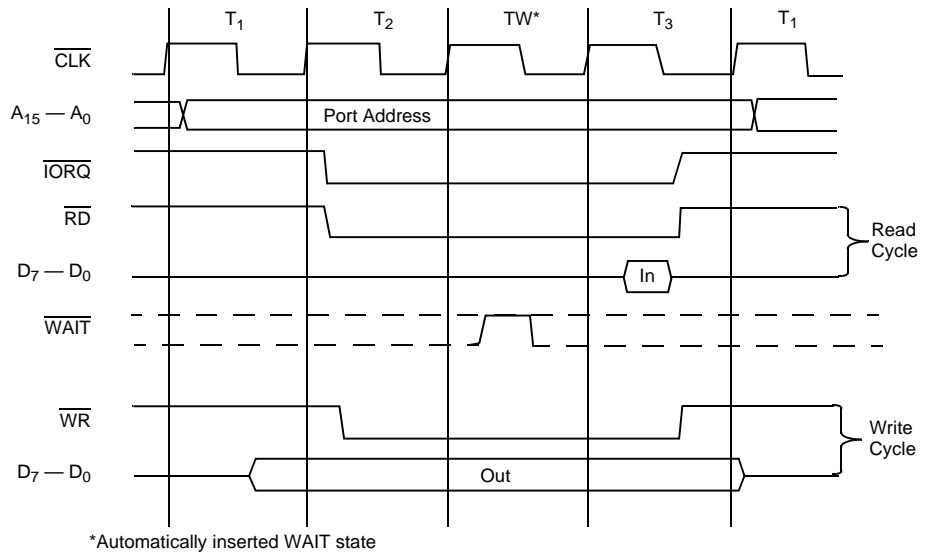
Figure 7.   Input or Output Cycles

Bus Request/Acknowledge Cycle

Figure 8 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{\text{BUSREQ}}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{\text{BUSREQ}}$ signal is active, the CPU sets its address, data, and tristate control signals to the high-impedance state with the rising edge of the next clock pulse. At that time, any external device can control the buses to transfer data between memory and I/O devices. (This operation is generally known as Direct Memory Access [DMA] using cycle stealing.) The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is required. If very long DMA cycles are used, and dynamic memories are used, the external controller also performs the refresh function. This situation only occurs if very large blocks of data

are transferred under DMA control. During a bus request cycle, the CPU cannot be interrupted by either an $\overline{\text{NMI}}$ or an $\overline{\text{INT}}$ signal.
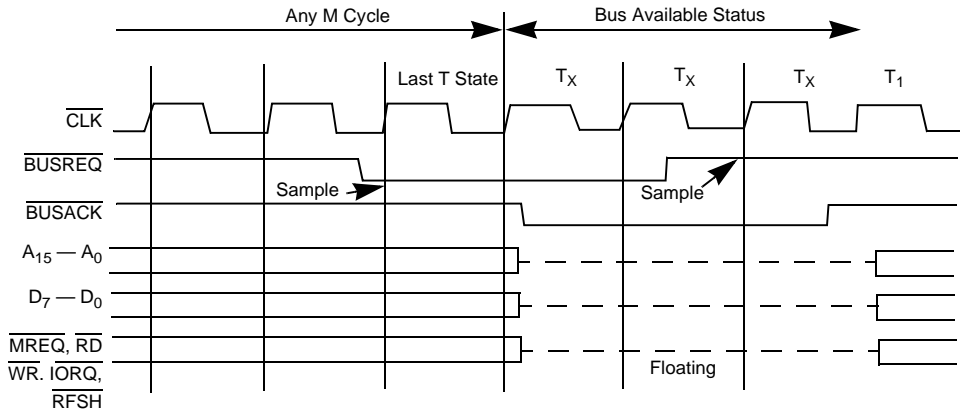
Figure 8.   Bus Request/Acknowledge Cycle

Interrupt Request/Acknowledge Cycle

Figure 9 illustrates the timing associated with an interrupt cycle. The CPU samples the interrupt signal ($\overline{\text{INT}}$) with the rising edge of the last clock at the end of any instruction. The signal is not accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{\text{BUSREQ}}$ signal is active. When the signal is accepted, a special M1 cycle is generated. During this special M1 cycle, the $\overline{\text{IORQ}}$ signal becomes active (instead of the normal $\overline{\text{MREQ}}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which I/O device must insert the response vector. Refer to Chapter 6 for details on how the interrupt response vector is utilized by the CPU.
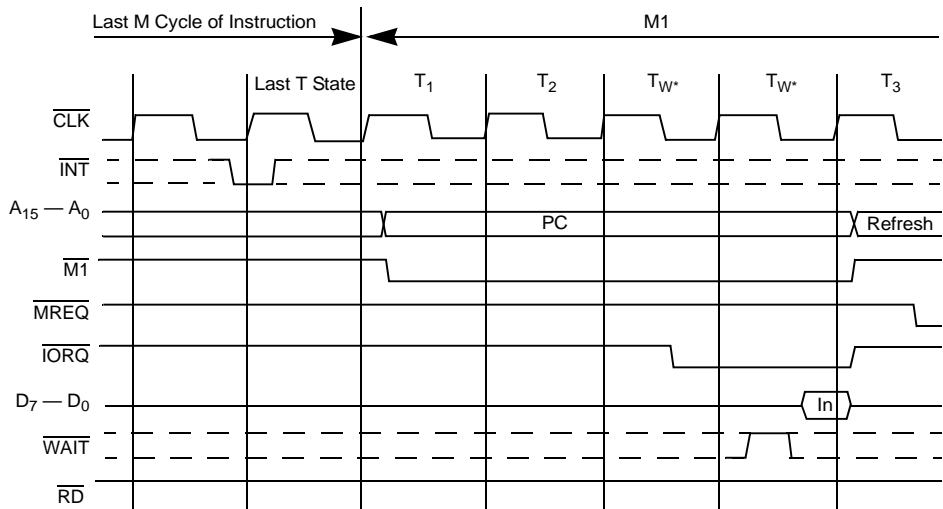
Figure 9.   Interrupt Request/Acknowledge Cycle

Non-Maskable Interrupt Response

Figure 10 illustrates the request/acknowledge cycle for the non-maskable interrupt. This signal is sampled at the same time as the interrupt line, but this line takes priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non-maskable interrupt is similar to a normal memory read operation. The only difference is that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066H. The service routine for the non-maskable interrupt must begin at this location if this interrupt is used.
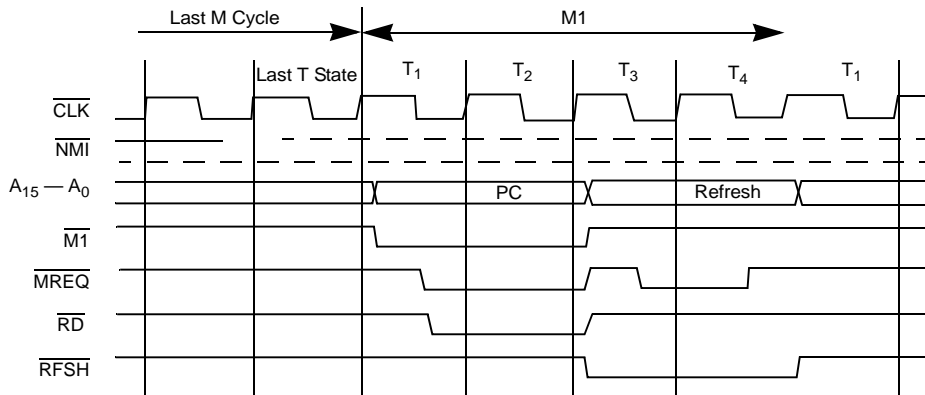
Figure 10.   Non-Maskable Interrupt Request Operation

HALT Exit

Whenever a software HALT instruction is executed, the CPU executes NOPs until an interrupt is received (either a non-maskable or a maskable interrupt while the interrupt flip-flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as depicted in Figure 11. If a non-maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the HALT state is exited on the next rising clock edge. The following cycle is an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non-maskable one is acknowledged since it has highest priority. The purpose of executing NOP instructions while in the HALT state is to keep the memory refresh signals active. Each cycle in the HALT state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The HALT acknowledge signal is active during this time indicating that the processor is in the HALT state.

HALT Instruction is repeated during this Memory Cycle
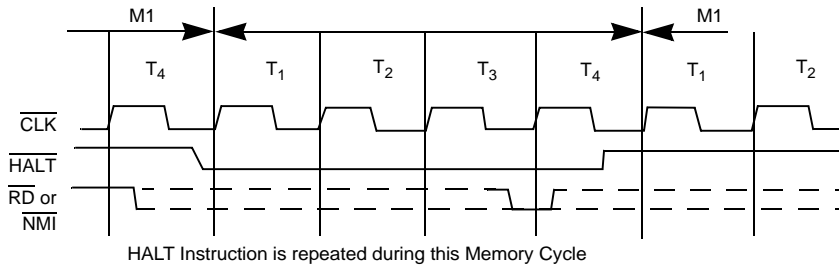
Figure 11. HALT Exit

Power-Down Acknowledge Cycle

When the clock input to the CMOS Z80 CPU is stopped at either a High or Low level, the CMOS Z80 CPU stops its operation and maintains all registers and control signals. However, ICC2 (standby supply current) is guaranteed only when the system clock is stopped at a Low level during T4 of the machine cycle following the execution of the HALT instruction. The timing diagram for the power-down function, when implemented with the HALT instruction, is shown in Figure 12.
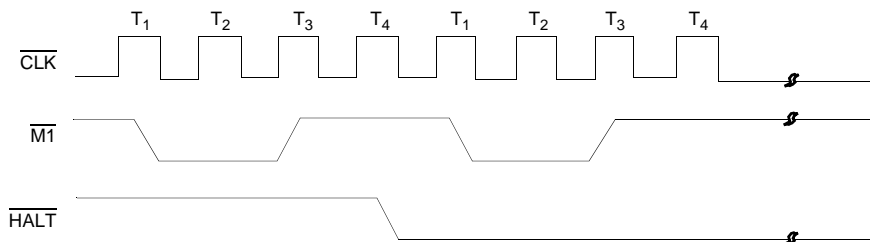
Figure 12. Power-Down Acknowledge

Power-Down Release Cycle

The system clock must be supplied to the CMOS Z80 CPU to release the power-down state. When the system clock is supplied to the CLK input, the CMOS Z80 CPU restarts operations from the point at which the power-down state was implemented. The timing diagrams for the release from power-down mode are featured in Figure 13 , 14 and 15.

When the HALT instruction is executed to enter the power-down state, the CMOS Z80 CPU also enters the HALT state. An interrupt signal (either $\overline{\text{NMI}}$ or ANT) or a $\overline{\text{RESET}}$ signal must be applied to the CPU after the system clock is supplied in order to release the power-down state.
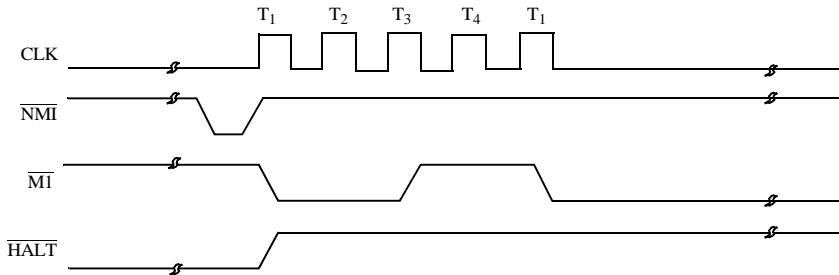
Figure 13. Power-Down Release Cycle No. 1
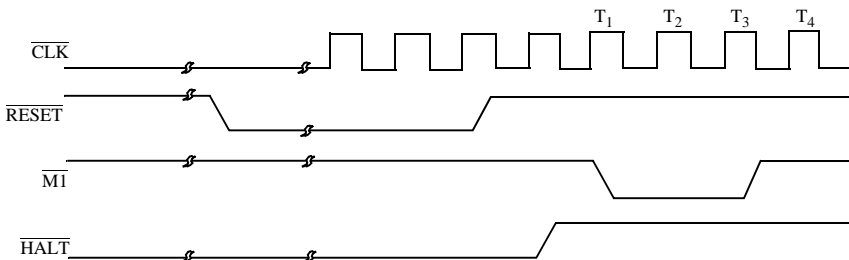
Figure 14. Power-Down Release Cycle No. 2
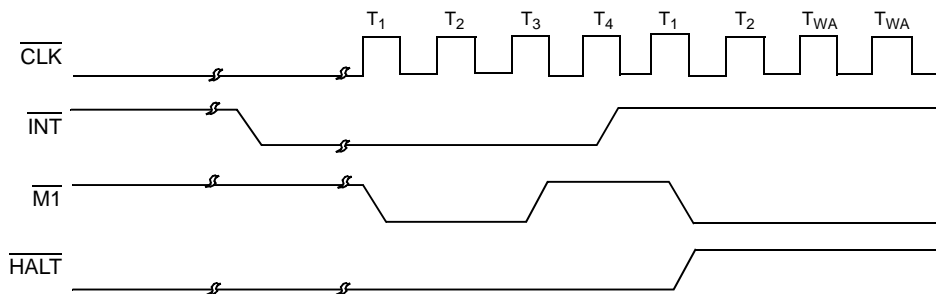
Figure 15. Power-Down Release Cycle No. 3

INTERRUPT RESPONSE

Overview

An interrupt allows peripheral devices to suspend CPU operation and force the CPU to start a peripheral service routine. This service routine usually involves the exchange of data, status, or control information between the CPU and the peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

Interrupt Enable/Disable

The Z80 CPU has two interrupt inputs, a software maskable interrupt ($\overline{\text{INT}}$) and a non-maskable interrupt ($\overline{\text{NMI}}$). The non-maskable interrupt cannot be disabled by the programmer and is accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that can be enabled or disabled selectively by the programmer. This routine allows the programmer to disable the interrupt during periods when his program has timing constraints that do not allow interrupt. In the Z80 CPU, there is an interrupt enable flip-flop (IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt cannot be accepted by the CPU.

The two enable flip-flops are IFF1 and IFF2.

IFF1		IFF2
Disables interrupts from being accepted		Temporary storage location for IFF1

The state of IFF1 is used to inhibit interrupts while IFF2 is used as a temporary storage location for IFF1.

A CPU reset forces both the IFF1 and IFF2 to the reset state, which disables interrupts. Interrupts can be enabled at any time by an EI instruction from the programmer. When an EI instruction is executed, any pending interrupt request is not accepted until after the instruction following EI is executed. This single instruction delay is necessary when the next instruction is a return instruction. Interrupts are not allowed until a return is completed. The EI instruction sets both IFF1 and IFF2 to the enable state. When the CPU accepts a maskable interrupt, both IFF1 and IFF2 are automatically reset, inhibiting further interrupts until the programmer issues a new EI instruction. Note that for all of the previous cases, IFF1 and IFF2 are always equal.

The purpose of IFF2 is to save the status of IFF1 when a non-maskable interrupt occurs. When a non-maskable interrupt is accepted, IFF1 resets to prevent further interrupts until reenabled by the programmer. Thus, after a non-maskable interrupt is accepted, maskable interrupts are disabled but the previous state of IFF1 has been saved so that the complete state of the CPU just prior to the non-maskable interrupt can be restored at any time. When a `Load Register A with Register I` (LD A, I) instruction or a `Load Register A with Register R` (LD A, R) instruction is executed, the state of IFF2 is copied to the parity flag where it can be tested or stored.

A second method of restoring the status of IFF1 is through the execution of a `Return From Non-Maskable Interrupt` (RETN) instruction. This instruction indicates that the non-maskable interrupt service routine is complete and the contents of IFF2 are now copied back into IFF1 so that the status of IFF1 just prior to the acceptance of the non-maskable interrupt is restored automatically.

Table 2 is a summary of the effect of different instructions on the two enable flip-flops.

Table 2. Interrupt Enable/Disable, Flip-Flops

Action	IFF1	IFF2	Comments
CPU Reset	0	0	Maskable Interrupt, $\overline{INT}$ Disabled

Table 2. Interrupt Enable/Disable, Flip-Flops

Action	IFF1	IFF2	Comments
DI Instruction Execution	0	0	Maskable $\overline{\text{INT}}$ Disabled
EI Instruction Execution	1	1	Maskable, $\overline{\text{INT}}$ Enabled
LD A,I Instruction Execution	*	*	IFF2 →☐ Parity Flag
LD A,R instruction Execution	*	*	IFF2 →☐ Parity Flag
Accept $\overline{\text{NMI}}$	0	*	Maskable☐Interrupt
RETN Instruction Execution	IFF2	*	IFF2 →☐indicates completion of non-maskable interrupt service routine.

CPU Response

Non-Maskable

The CPU always accepts a non-maskable interrupt. When this occurs, the CPU ignores the next instruction that it fetches and instead performs a restart to location 0066H. The CPU functions as if it had recycled a restart instruction, but to a location other than one of the eight software restart locations. A restart is merely a call to a specific address in page 0 of memory.

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is similar to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU executes it. Thus, the interrupting device provides the next instruction to be executed. Often this is a restart instruction because the interrupting device only need supply a single byte instruction. Alternatively, any other

instruction such as a 3-byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is two more than the normal number for the instruction. This occurs because the CPU automatically adds two wait states to an Interrupt response cycle to allow sufficient time to implement an external daisy-chain for priority control. Figure 9 and Figure 10 illustrate the detailed timing for an interrupt response. After the application of $\overline{RESET}$, the CPU automatically enters interrupt Mode 0.

Mode 1

When this mode is selected by the programmer, the CPU responds to an interrupt by executing a restart to location 0038H. Thus, the response is identical to that for a non-maskable interrupt except that the call location is 0038H instead of 0066H. The number of cycles required to complete the restart instruction is two more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8-bit byte from the user, an indirect call can be made to any memory location.

In this mode, the programmer maintains a table of 16-bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16-bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper eight bits of this pointer is formed from the contents of the I register. The I register must be loaded with the applicable value by the programmer, such as LD I, A. A CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Only seven bits are required from the interrupting device because the least-significant bit must be a zero. This is required

because the pointer is used to get two adjacent bytes to form a complete 16-bit service routine starting address and the addresses must always start in even locations.
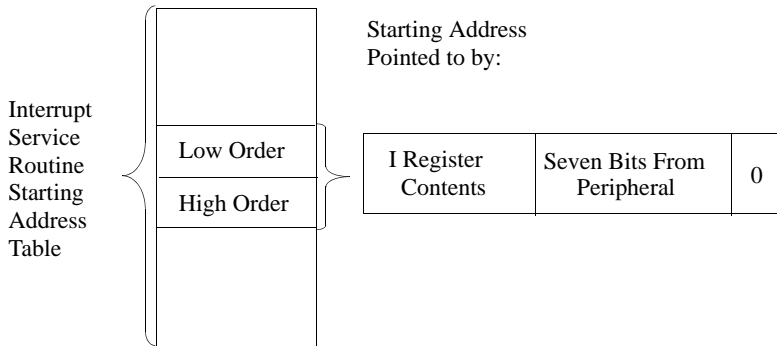
Figure 16. Mode 2 Interrupt Response Mode

The first byte in the table is the least-significant (low order portion of the address). The programmer must complete this table with the correct addresses before any interrupts are accepted.

The programmer can change this table by storing it in Read/Write Memory, which also allows individual peripherals to be serviced by different service routines.

When the interrupting device supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table, and performs a jump to this address. This mode of response requires 19 clock periods to complete (seven to fetch the lower eight bits from the interrupting device, six to save the program counter, and six to obtain the jump address).

The Z80 peripheral devices include a daisy-chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80 CPU Peripherals User Manual for more complete information.

Hardware and Software Implementation Examples

HARDWARE

Minimum System

This chapter is an introduction to implementing systems that use the Z80 CPU. Figure 17 illustrates a simple Z80 system.

Any Z80 system must include the following elements:

- 5V Power Supply

- Oscillator

- Memory Devices

- I/O Circuits

- CPU
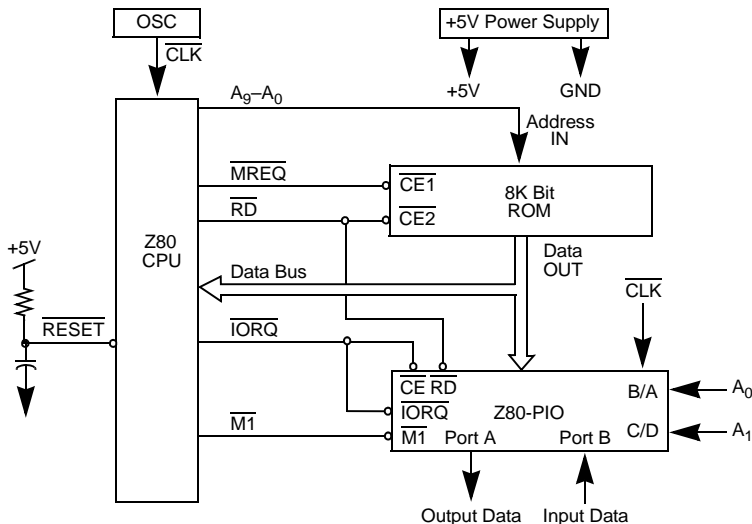
Figure 17. Minimum Z80 Computer System

Because the Z80 CPU requires only a single 5V power supply, most small systems can be implemented using only this single supply.

The external memory can be any mixture of standard RAM, ROM, or PROM. In Figure 17, a single 8K bit ROM (1 Kbytes) comprises the entire memory system. The Z80 internal register configuration contains sufficient Read/Write storage, requiring no external RAM memory.

I/O circuits allow computer systems to interface with the external devices. In Figure 17, the output is an 8-bit control vector and the input is an 8-bit status word. The input data can be gated to the data bus using any standard three-state driver while the output data can be latched with any type of standard TTL latch. A Z80 PIO serves as the I/O circuit. This single circuit attaches to the data bus as indicated and provides the required 16 bits of TTL compatible I/O. (Refer to the Z80 CPU Peripherals User's Manual for details on the operation of this circuit.) This powerful computer is built with only three LSI circuits, a simple oscillator, and a single 5V power supply.

Adding RAM

Most computer systems require some external Read/Write memory for data storage and stack implementation. Figure 18 illustrates how 256 bytes of static memory are added to the previous example in Figure 17. The memory space is assumed to be organized as follows:

Address:

1 Kbyte ROM	0000H
	03FFH
256 Bytes RAM	0400H
	04FFFH

In this diagram the address space is described in hexadecimal notation. Address bit A10 separates the ROM space from the RAM space, allowing this address to be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder is required to form the chip selects.
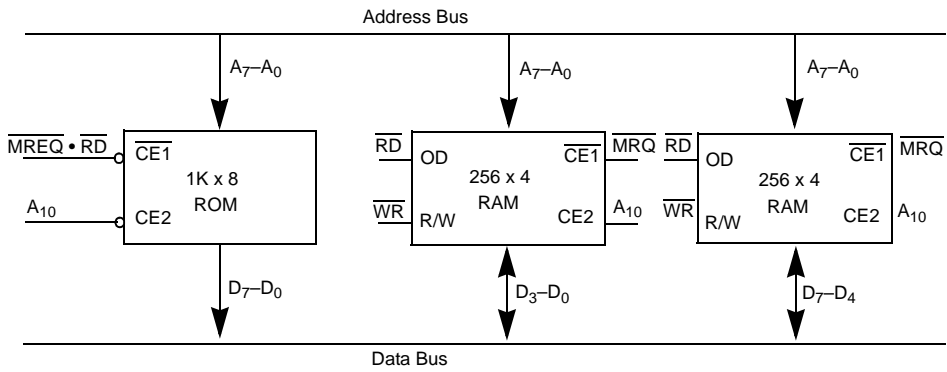
Figure 18. ROM and RAM Implementation

Memory Speed Control

Slow memories can reduce costs for many applications. The $\overline{\text{WAIT}}$ line on the CPU allows the Z80 to operate with any speed memory. Memory access time requirements, which are covered in Chapter A3, are most severe during the $\overline{\text{M1}}$ cycle instruction fetch. All other memory access cycles complete in an additional one half clock cycle. Hence, it is sometimes appropriate to add one wait state to the $\overline{\text{M1}}$ cycle so slower memories can be used. Figure 19 is an example of a simple circuit that accomplishes this objective. This circuit can be changed to add a single wait state to any memory access as indicated in Figure 20.
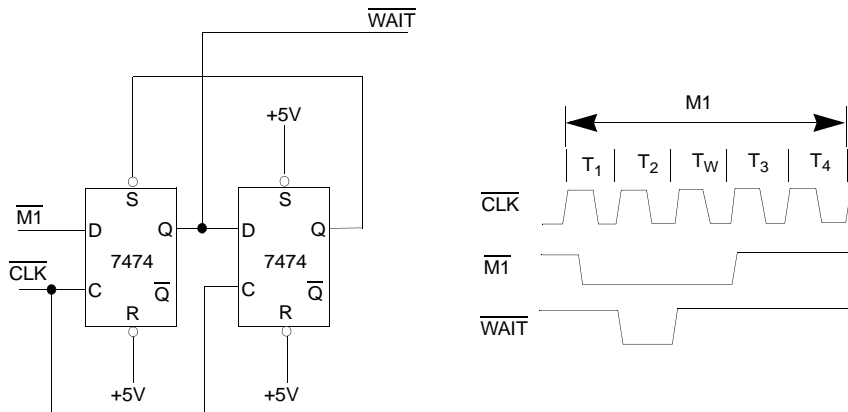
Figure 19. Adding One Wait State to an M1 Cycle
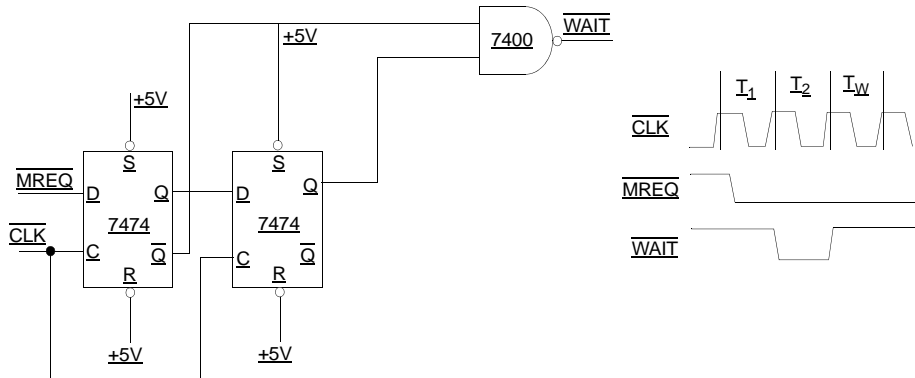
Figure 20.   Adding One Wait State to Any Memory Cycle

Interfacing Dynamic Memories

Each individual dynamic RAM has it's own specifications that require minor modifications to the examples given here. ZiLOG Application Notes are available describing how the Z80 CPU is interfaced with most popular dynamic RAM.

Figure 21 illustrates the logic necessary to interface 8 Kbytes of dynamic RAM using 18-pin 4K dynamic memories. This logic assumes that the RAMs are the only memory in the system so that A12 is used to select between the two pages of memory. During refresh time, all memories in the system must be read. The CPU provides the correct refresh address on lines A0 through A6. When adding more memory to the system, it is necessary to replace only the two gates that operate on A12 with a decoder that operates on all required address bits. Address buffers and data bus buffers are generally required for larger systems.
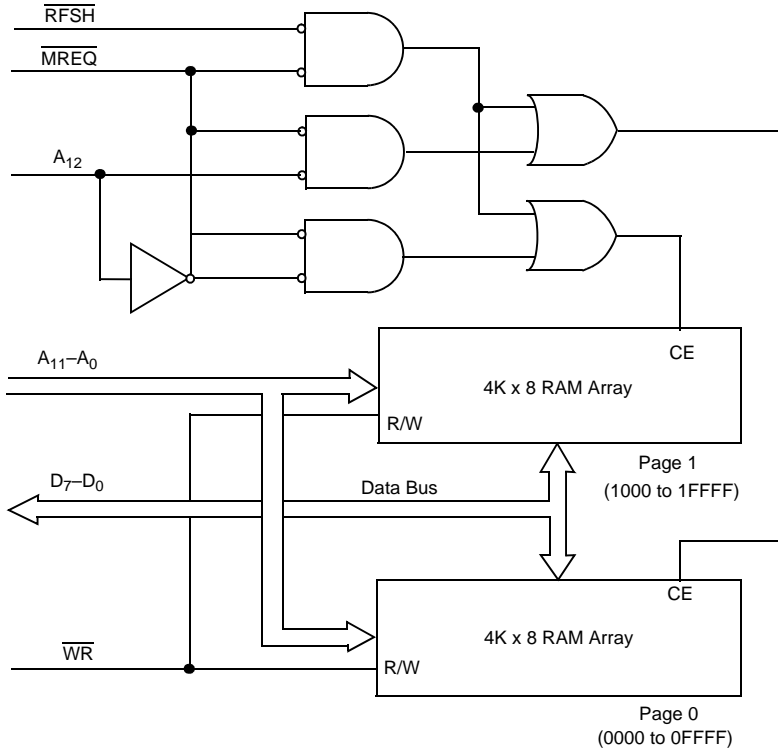
Figure 21. Interfacing Dynamic RAMs

SOFTWARE IMPLEMENTATION EXAMPLES

Overview of Software Features

The Z80 instruction set provides the user with a large number of operations to control the Z80 CPU. The main alternate and index registers can hold arithmetic and logical operations, form memory addresses, or act as fast-access storage for frequently used data.

Information can be moved directly from register to register, from memory to memory, from memory to registers, or from registers to memory. In addition, register contents and register/memory contents can be exchanged without using temporary storage. In particular, the contents of main and alternate registers can be completely exchanged by executing only two instructions, EX and EXX. This register exchange procedure can be used to separate the set of working registers from different logical procedures or to expand the set of available registers in a single procedure.

Storage and retrieval of data between pairs of registers and memory can be controlled on a last-in first-out basis through PUSH and POP instructions that utilize a special STACK POINTER register (SP). This stack register is available both to manipulate data and to automatically store and retrieve addresses for subroutine linkage. When a subroutine is called, for example, the address following the CALL instruction is placed on the top of the push-down stack pointed to by SP. When a subroutine returns to the calling routine, the address on the top of the stack is used to set the program counter for the address of the next instruction. The stack pointer is adjusted automatically to reflect the current top stack position during PUSH, POP, CALL, and RET instructions. This stack mechanism allows pushdown data stacks and subroutine calls to be nested to any practical depth because the stack area can potentially be as large as memory space.

The sequence of instruction execution can be controlled by six different flags (carry, zero, sign, parity/overflow, add/subtract, half-carry), which reflect the results of arithmetic, logical, shift, and compare instructions.

After the execution of an instruction that sets a flag, that flag can be used to control a conditional jump or return instruction. These instructions provide logical control following the manipulation of single bit, 8-bit byte, or 18-bit data quantities.

A full set of logical operations, including AND, OR, XOR (exclusive-OR), CPL (NOR), and NEG (two's complement) are available for Boolean operations between the accumulator and all other 8-bit registers, memory locations, or immediate operands.

In addition, a full set of arithmetic and logical shifts in both directions are available which operate on the contents of all 8-bit primary registers or directly on any memory location. The carry flag can be included or set by these shift instructions to provide both the testing of shift results and to link register/register or register/memory shift operations.

Examples of Specific Z80 Instructions

Example One:

When a 737-byte data string in memory location DATA must be moved to location BUFFER, the operation is programmed as follows:

```
LD   HL, DATA  ;START ADDRESS OF DATA STRING
LD   DE, BUFFER;START ADDRESS OF TARGET BUFFER
LD   BC, 737   ;LENGTH OF DATA STRING
LDIR           ;MOVE STRING - TRANSFER MEMORY POINTED
               ;TO BY HL INTO MEMORY LOCATION POINTED
               ;TO BY DE INCREMENT HL AND DE,
               ;DECREMENT BC PROCESS UNTIL BC = 0.
```

Eleven bytes are required for this operation and each byte of data is moved in 21 clock cycles.

Example Two:

A string in memory (limited to a maximum length of 132 characters) starting at location DATA is to be moved to another memory location starting at location BUFFER until an ASCII $ (used as a string delimitor) is found. This operation is performed as follows:

```
      LD  HL, DATA   ;STARTING ADDRESS OF DATA STRING
      LD  DE, BUFFER ;STARTING ADDRESS OF TARGET BUFFER
      LD  BC, 132    ;MAXIMUM STRING LENGTH
      LD  A, '$'     ;STRING DELIMITER CODE
LOOP: CP  (HL)       ;COMPARE MEMORY CONTENTS WITH
                     ;DELIMITER
      JR  Z, END-$   ;GO TO END IF CHARACTERS EQUAL
      LDI            ;MOVE CHARACTER (HL) to (DE)
                     ;INCREMENT HL AND DE, DECREMENT BC
      JP  PE, LOOP   ;GO TO "LOOP" IF MORE CHARACTERS
END:                 ;OTHERWISE, FALL THROUGH
                     ;NOTE: P/V FLAG IS USED
                     ;TO INDICATE THAT REGISTER BC WAS
                     ;DECREMENTED TO ZERO.
```

Nineteen bytes are required for this operation.

Example Three:

A 16-digit decimal number is shifted as depicted in the Figure 22. This shift is performed to mechanize BCD multiplication or division. The 16-digit decimal number is represented in packed BCD format (two BCD digits/byte) The operation is programmed as follows:

```
      LD   HL, DATA ;ADDRESS OF FIRST BYTE
      LD   B, COUNT ;SHIFT COUNT
      XOR  A        ;CLEAR ACCUMULATOR
ROTAT:RLD            ;ROTATE LEFT LOW ORDER DIGIT IN ACC
                     ;WITH DIGITS IN (HL)
      INC  HL       ;ADVANCE MEMORY POINTER.
      DJNZ ROTAT-$  ;DECREMENT B AND GO TO ROTAT IF
```

```
                         ;B IS NOT ZERO, OTHERWISE FALL
                         ;THROUGH.
```

Eleven bytes are required for this operation.
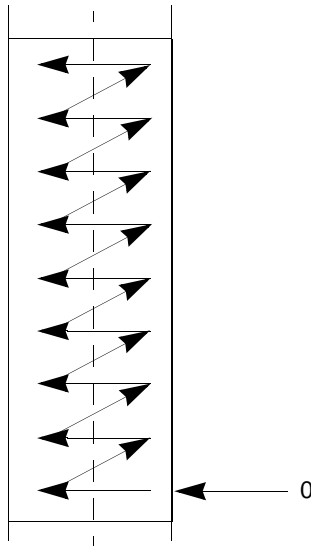
Figure 22. Shifting of BCD Digits/Bytes

Example Four:

One number is to be subtracted from another number, both of which are in packed BCD format and are of equal but varying length. The result is stored in the location of the minuend. The operation is programmed as follows:

```
        LD   HL, ARG1   ;ADDRESS OF MINUEND
        LD   DE, ARG2   ;ADDRESS OF SUBTRAHEND
        LD   B, LENGTH  ;LENGTH OF TWO ARGUMENTS
        AND  A          ;CLEAR CARRY FLAG
SUBDEC: LD   A, (DE)    ;SUBTRAHEND TO ACC
        SBC  A, (HL)    ;SUBTRACT (HL) FROM ACC
```

```
        DAA                 ;ADJUST RESULT TO DECIMAL CODED
VALUE
        LD   (HL), A   ;STORE RESULT
        INC  HL        ;ADVANCE MEMORY POINTERS
        INC  DE
        DJNZ SUBDEC - $;DECREMENT B AND GO TO "SUBDEC"
                          ;IF B
                          ;NOT ZERO, OTHERWISE FALL
                          ;THROUGH
```

Seventeen bytes are required for this operation.

Examples of Programming Tasks

As depicted in Table 3, this example program sorts an array of numbers to ascending order, using a standard exchange sorting algorithm. These numbers range from 0 to 255.

Table 3. Bubble Listing

Loc	Obj Code	Stmt	Source Statement		
		1	;	standard exchange (bubble) sort routine	
		2	;		
		3	;	at entry:	hl contains address of data
		4			c contains number of elements to be sorted
		5			(1 <c<256)
		6	;		
		7	;	at exit	data sorted in ascending order
		8	;		
		9	;	use of registers	
		10	;		
		11	;	register	contents
		12	;		
		13	;	a	temporary storage for calculations
		14	;	b	counter for data array

Table 3. Bubble Listing (Continued)

Loc	Obj Code	Stmt	Source Statement			
		15	;	c	length of data array	
		16	;	d	first element in comparison	
		17	;	e	second element in comparison	
		18	;	h	flag to indicate exchange	
		19	;	l	unused	
		20	;	ix	pointer into data array	
		21	;	iy	unused	
		22	;			
0000	222600	23	sort:	ld	(data), hl	; save data address
0003	cb84	24	loop:	res	flag, h	; initialize exchange flag
0005	41	25		ld	b, c	; initialize length counter
0006	05	26		dec	b	; adjust for testing
0007	dd2a2600	27		ld	ix, (data)	; initialize array pointer
000b	dd7e00	28	next:	ld	a, (ix)	; first element in comparison
000e	57	29		ld	d, a	; temporary storage for element
goof	dd5e01	30		ld	e, (ix+1)	; second element in comparison
0012	93	31		sub	e	; comparison first to second
0013	3008	32		jr	pc, noex-$	; if first > second, no jump
0015	dd7300	33		ld	(ix), e	; exchange array elements
0018	dd7201	34		ld	(ix+i), d	
001b	cbc4	35		set	flag, h	; record exchange occurred
0010	dd23	36	noex:	inc	ix	; point to next data element
001f	10ea	37		djnz	next-$	; count number of comparisons
		38				; repeat if more data pairs
0021	cb44	39		bit	flag, h	; determine if exchange occurred
0023	20de	40		jr	nz, loop-$	; continue if data unsorted
0025	c9	41		ret		; otherwise, exit
		42	;			

Table 3. Bubble Listing (Continued)

Loc	Obj Code	Stmt	Source Statement			
0026		43	flag:	equ	0	; designation of flag bit
0026		44	data:	defs	2	; storage for data address
		45		end		

The following program (see Table 4) multiplies two unsigned 16-bit integers, leaving the result in the HL register pair.

Table 4. Multiply Listing

Loc	Obj Code	Stmt	Source Statement		
0000		1	mult:;		unsigned sixteen bit integer multiply.
		2	;		on entrance: multiplier in de.
		3	;		multiplicand in hl.
		4	;		
		5	;		on exit result in hl.
		6	;		
		7	;		register uses:
		8	;		
		9	;		
		10	;	h	high order partial result
		11	;	l	low order partial result
		12	;	d	high order multiplicand
		13	;	e	low order multiplicand
		14	;	b	counter for number of shifts
		15	;	c	high order bits of multiplier

Table 4. Multiply Listing (Continued)

Loc	Obj Code	Stmt	Source Statement			
		16	;	a	low order bits of multiplier	
		17	;			
0000	0610	18		ld	b, 16;	number of bits-initialize
0002	4a	19		ld	c, d;	move multiplier
0003	7b	20		ld	a, e;	
0004	eb	21		ex	de, hl;	move multiplicand
0005	210000	22		ld	hl, 0;	clear partial result
0008	cb39	23	mloop:	srl	c;	shift multiplier right
000a	if	24		rra		least significant bit is
		25	;			in carry.
000b	3001	26		jr	nc, noadd-$;	if no carry, skip the add.
good	19	27		add	hl, de;	else add multiplicand to
		28	;			partial result.
000e	eb	29	noadd:	ex	de, h l;	shift multiplicand left
goof	29	30		add	hl, hl;	by multiplying it by two.
0010	eb	31		ex	de, hl;	
0011	10f5	32		djnz	mloop-$;	repeat until no more bits.
0013	c9	33		ret;		
		34		end;		

Z80 CPU Instruction Description

Overview

The Z80 CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions fall into these major groups:

- Load and Exchange

- Block Transfer and Search

- Arithmetic and Logical

- Rotate and Shift

- Bit Manipulation (Set, Reset, Test)

- Jump, Call, and Return

- Input/Output

- Basic CPU Control

Instruction Types

The load instructions move data internally among CPU registers or between CPU registers and external memory. All these instructions specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general-purpose registers such as move the data to register B from register C. This group also includes load-immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z80. With a single instruction, a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when processing large strings of data. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. When the character is found or the end of the block is reached, the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so they do not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general-purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation.

An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The rotate and shift group allows any register or any memory location to be rotated right or left, with or without carry either arithmetic or logical. Also, a digit in the accumulator can be rotated right or left with two digits in any memory location.

The bit manipulation instructions allow any bit in the accumulator, any general-purpose register, or any external memory location to be set, reset, or tested with a single instruction. For example, the most-significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general-purpose programming.

The JUMP, CALL, and RETURN instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of call is the RESTART instruction. This instruction actually contains the new address as a part of

the 8-bit Op Code. This is possible because only eight separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX, or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

The input/output group of instructions in the Z80 allow for a wide range of transfers between external memory locations or the general-purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower eight bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows multiple I/O ports to share common software driver routines. This advantage is not possible when the address is part of the Op Code if the routines are stored in ROM. Another feature of these input instructions is the automatic setting of the flag register, making additional operations unnecessary to determine the state of the input data. The parity state is one example.

The Z80 CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general-purpose registers, these instructions provide fast I/O block transfer rates. The power of this I/O instruction set is demonstrated by the Z80 CPU providing all required floppy disk formatting on double-density floppy disk drives on an interrupt-driven basis. For example, the CPU provides the preamble, address, data, and enables the CRC codes.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip-flop or setting the mode of interrupt response.

Addressing Modes

Most of the Z80 instructions operate on data stored in internal CPU registers, external memory, or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section is a brief summary of the types of addressing used in the Z80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate

In this mode of addressing, the byte following the Op Code in memory contains the actual operand.

Op Code	—One or Two Bytes
Op Code	

D7   D0
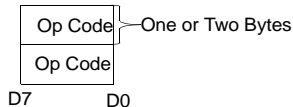
Examples of this type of instruction is loading the accumulator with a constant, where the constant is the byte immediately following the Op Code.

Immediate Extended

This mode is an extension of immediate addressing in that the two bytes following the Op Codes are the operand.

Op Code	One or Two Bytes
Op Code	Low Order
Op Code	High Order
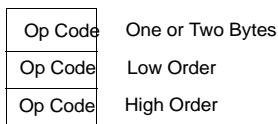
Examples of this type of instruction is loading the HL register pair (16-bit register) with 16 bits (two bytes) of data.

Modified Page Zero Addressing

The Z80 has a special single byte CALL instruction to any of eight locations in page zero of memory. This instruction, which is referred to as a restart, sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

| Op Code | One Byte |

B7   B0 Effective Address is
(B5 B4 B3 000)2
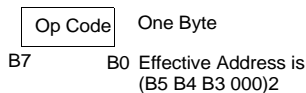
Relative Addressing

Relative addressing uses one byte of data following the Op Code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the Op Code of the following instruction.

| Op Code |
| Op Code |

Jump Relative (One Byte Op Code)
8-Bit Two's Complement
Displacement Added to
Address (A+2)
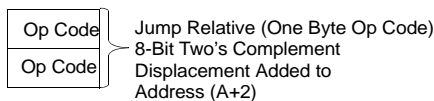
The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can Significantly reduce memory space requirements. The signed displacement can range between +127 and -128 from A+2. This allows for a total displacement of +129 to -126 from the jump relative Op Code address. Another major advantage is that it allows for relocatable code.

Extended Addressing

Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.

Op Code
Low Order Address to Low Order Operand
High Order Address to Low Order Operand

One or Two Bytes
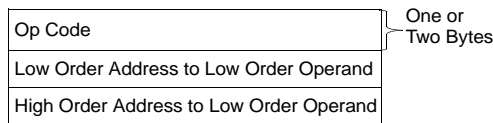
Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

During extended addressing use, specify the source or destination address of an operand. This notation (nn) is used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. The two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (3200) refers to the contents of memory at location 1200.

Indexed Addressing

In this type of addressing, the byte of data following the Op Code contains a displacement that is added to one of the two index registers (the Op Code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.

Op Code
Op Code
Displacement

Two Byte Op Code

Operand added to index register to form a pointer to memory
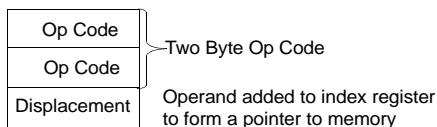
An example of an indexed instruction is to load the contents of the memory location (Index Register + Displacement) into the accumulator.

The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data because the index register can point to the start of any table. Two index registers are provided because very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z80 are referred to as IX and IY. To indicate indexed addressing the notation use:

```
(IX+d) or (IY+d)
```

Here d is the displacement specified after the Op Code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing

Many of the Z80 Op Codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing is to load the data in register 6 into register C.

Implied Addressing

Implied addressing refers to operations where the Op Code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing

This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.

Op Code — One or Two Bytes

An example of this type of instruction is to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing

except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z80 are extensions of this type of addressing where automatic register incrementing, decrementing, and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol (HL) specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing

The Z80 contains a large number of bit set, reset, and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect, and indexed) while three bits in the Op Code specify which of the eight bits is to be manipulated.

Addressing Mode Combinations

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

Instruction Op Codes

This section describes each of the Z80 instructions and provides tables listing the Op Codes for every instruction. In each of these tables, the Op Codes in shaded areas are identical to those offered in the 8080A CPU.

Also depicted is the assembly language mnemonic that is used for each instruction. All instruction Op Codes are listed in hexadecimal notation. Single byte Op Codes require two hex characters while double byte Op Codes require four hex characters. For convenience, the conversion from hex to binary is repeated in Table 5.

Table 5. Hex, Binary, Decimal Conversion Table

Hex		Binary		Decimal
0	=	0000	=	0
1	=	0001	=	1
2	=	0010	=	2
3	=	0011	=	3
4	=	0100	=	4
5	=	0101	=	5
6	=	0110	=	6
7	=	0111	=	7
8	=	1000	=	8
9	=	1001	=	9
A	=	1010	=	10
B	=	1011	=	11
C	=	1100	=	12
D	=	1101	=	13
E	=	1110	=	14
F	=	1111	=	15

The Z80 instruction mnemonics consist of an Op Code and zero, one, or two operands. Instructions where the operand is implied contains no operand. Instructions that contain only one logical operand, where one operand is invariant (such as the Logical OR instruction), are represented by a one operand mnemonic. Instructions that contain two varying operands are represented by two operand mnemonics.

Load and Exchange

Table 6 defines the Op Code for all the 8-bit load instructions implemented in the Z80 CPU. Also described in this table is the type of addressing used for each instruction. The source of the data is found on the top horizontal row and the destination is specified in the left column. For example, load register C from register B uses the Op Code 48H. In all the figures, the Op Code is specified in hexadecimal notation and the 48H (0100 1000 binary) code is fetched by the CPU from the external memory during M1 time, decoded, and then the register transfer is automatically performed by the CPU.

The assembly language mnemonic for this entire group is LD, followed by the destination, followed by the source (LD DEST, SOURCE). Note that several combinations of addressing modes are possible. For example, the source may use register addressing and the destination may be register indirect; such as load the memory location pointed to by register HL with the contents of register D. The Op Code for this operation is 72. The mnemonic for this load instruction is LD (HL), D.

The parentheses around the HL indicates that the contents of HL are used as a pointer to a memory location. In all Z80 load instruction mnemonics, the destination is always listed first, with the source following. The Z80 assembly language is defined for ease of programming. Every instruction is self documenting and programs written in Z80 language are easy to maintain.

In Table 6, some Op Codes that are available in the Z80 use two bytes. This feature is an efficient method of memory utilization because 8-, 18-, 24-, or 32-bit instructions are implemented in the Z80. Often utilized instructions such as arithmetic or logical operations are only eight bits, which results in better memory utilization than is achieved with fixed instruction sizes such as 16 bits.

Table 6. 8-Bit Load Group LD

Destination		Implied		Register							Reg Indirect			11indexed		Ext Addr.	Imme.
		I	R	A	B	C	D	E	F	L	(HL)	(BC)	(DE)	(IX+d)	(lY+d)	Inn)	n
Register	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	FD 7E d	DD 7E d	FD 3A nn	FD 2E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		DD D5 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		DD DE n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		DD 1B n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		DD 1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		DD 2B n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		DD 36 n
Reg Indirect	(HL)			77	70	71	72	73	74	75							DD 78 D
	(BC)			02													
	(DE)			12													
INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXT, ADDR	(nn)			32 n n			.										
IMPLIED	I			ED 47													
	R			ED 4F													

All load instructions using indexed addressing for either the source or destination location actually use three bytes of memory with the third byte being the displacement d. For example, a load register E with the operand pointed to by IX with an offset of +8 is written:

```
LID E, (IX + 8)
```

The instruction sequence for this in memory is:

Address A	DD	Op Code
A+1	5E	
A+2	08	Displacement Operand
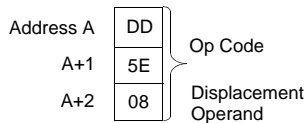
The two extended addressing instructions are also three byte instructions. For example, the instruction to load the accumulator with the operand in memory location 6F32H is written:

```
LID A, (6F 32H)
```

and its instruction sequence is:

Address A	3A	Op Code
A+1	32	Low Order Address
A+2	6F	High Order Address
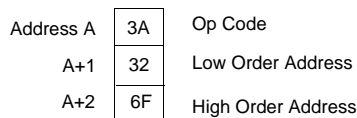
Notice that the low order portion of the address is always the first operand.

The load immediate instructions for the general-purpose 8-bit registers are two-byte instructions. The instruction load register H with the value 36H is written:

```
LD H, 36H
```

and its sequence is:

Address A	26	Op Code
A+1	36	Operand
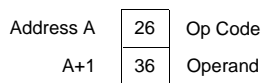
Loading a memory location using indexed addressing for the destination and immediate addressing for the source requires four bytes. For example,

```
LD (IX - 15), 21H
```

appears as:

| Address A | DD |
| A+1 | 36 |

Op Code

| A+2 | F1 | One or Two Bytes
Displacement (-15 in Signed |
| A+3 | 21 | Two's Complement
Operand to Load |
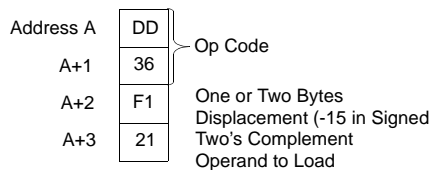
Notice that with any indexed addressing the displacement always follows directly after the Op Code.

Table 7 specifies the 16-bit load operations. The extended addressing feature covers all register pairs. Register indirect operations specifying the stack pointer are the PUSH and POP instructions. The mnemonic for these instructions is PUSH and POP. These differ from other 16-bit loads in that the stack pointer is automatically decremented and incremented as each byte is pushed onto or popped from the stack respectively. For example, the instruction PUSH AF is a single byte instruction with the Op Code of F5H. During execution, this sequence is generated:

```
Decrement SP
LD (SP), A
Decrement SP
LD (SP), F
```

The external stack now appears as:
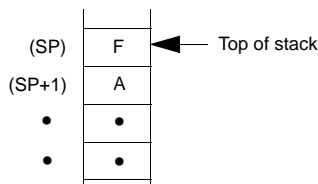
The POP instruction is the exact reverse of a PUSH. All PUSH and POP instructions utilize a 16-bit operand and the high order byte is always pushed first and popped last.

```
PUSH BC   is PUSH 8 then C

PUSH DE   is PUSH D then E

PUSH HL   is PUSH H then L

POP HL    is POP L then H
```

The instruction using extended immediate addressing for the source requires two bytes of data following the Op Code. For example,

```
LD DE, 0659H
```

appears as:

In all extended immediate or extended addressing modes, the low order byte always appears first after the Op Code.

Table 8 lists the 16-bit exchange instructions implemented in the Z80. Op Code 08H allows the programmer to switch between the two pairs of accumulator flag registers while D9H allows the programmer to switch between the duplicate set of six general-purpose registers. These Op Codes are only one byte in length to minimize the time necessary to perform the exchange so that the duplicate banks can be used to make very fast interrupt response times.

Table 7. 16-Bit Load Group LD, PUSH and POP

		Source							Imm. Ext.	Ext. Addr.	Reg. Indir.
		Register									
Register		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
	AF										P1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
EXT ADDR.	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
REG. IND.	(SP)	F6	C6	D6	E6		DD E6	FD E6			

PUSH Instructions →

NOTE: The Push & Pop instruction adjust the SP after every execution.

POP Instructions

Table 8. Exchanges EX and EXX

		Implied Addressing				
		AF'	BC', DE', and HL'	HL	IX	IY
IMPLIED	AF	08				
	BC DE HL		D9			
	DE			EB		
REG. IND.	(SP)			E3	DD E3	FD E3

Block Transfer and Search

Table 9 lists the extremely powerful block transfer instructions. These instructions operate with three registers.

- HL points to the source location
- DE points to the destination location
- BC is a byte counter

After the programmer initializes these three registers, any of these four instructions can be used. The LDI (Load and Increment) instruction moves one byte from the location pointed to by HL to the location pointed to by DE. Register pairs HL and DE are then automatically incremented and are ready to point to the following locations. The byte counter (register pair BC) is also decremented at this time. This instruction is valuable when blocks of data must be moved but other types of processing are required between each move. The LDIR (Load, Increment and Repeat) instruction is an extension of the LDI instruction. The same load and increment operation is repeated until the byte counter reaches the count of zero. Thus, this single instruction can move any block of data from one location to any other.

Because 16-bit registers are used, the size of the block can be up to 64 Kbytes (1K = 1024) long and can be moved from any location in memory to any other location. Furthermore, the blocks can be overlapping because there are no constraints on the data used in the three register pairs.

The LDD and LDDR instructions are very similar to the LDI and LDIR. The only difference is that register pairs HL and DE are decremented after every move so that a block transfer starts from the highest address of the designated block rather than the lowest.

Table 10 specifies the Op Codes for the four block search instructions. The first, CPI (Compare and Increment) compares the data in the accumulator with the contents of the memory location pointed to by register HL. The result of the compare is stored in one of the flag bits and the HL register pair is then incremented and the byte counter (register pair BC) is decremented.

The instruction CPIR is merely an extension of the CPI instruction in which the compare is repeated until either a match is found or the byte counter (register pair BC) becomes zero. Thus, this single instruction can search the entire memory for any 8-bit character.

The CPD (Compare and Decrement) and CPDR (Compare, Decrement, and Repeat) are similar instructions, their only difference is that they decrement HL after every compare so that they search the memory in the opposite direction. The search is started at the highest location in the memory block.

These block transfer and compare instructions are extremely powerful in string manipulation applications.

Table 9. Block Transfer Group

Destination		Source	
Reg. Indir.	(DE)	Reg. Indir.	
		(HL)	
		(ED) A0	LDI - Load (DE) → (HL) Inc HL and DE, Dec BC
		(ED) B0	LDIR, - Load (DE) →(HL) Inc HL and DE, Dec BC, Repeat until BC = 0
		(ED) A8	LDD - Load (DE) → (HL) Inc HL and DE, Dec BC
		(ED) B8	LDDR - Load (DE) → (HL) Dec HL and DE, Dec BC, Repeat until BC = 0

Note:
Reg HL points to source
Reg DE points to destination
Reg BC is byte counter

Table 10. Block Search Group

Search Location	
Reg. Indir.	
(HL)	
(ED) A1	CPI Inc HL, Dec BC
(ED) B1	CPRI. Inc HL, Dec BC Repeat until) BC = 0 or find match
(ED) A9	WD Dec HL and BC
(ED) B9	CPDR Dec HL and BC Repeat until BC = 0 or find match

Note: HL points to location in memory to be compared with accumulator contents
BC Is byte counter

Arithmetic and Logical

Table 11 lists all the 8-bit arithmetic operations that can be performed with the accumulator, also listed are the increment (INC) and decrement

(DEC) instructions. In all these instructions, except INC and DEC, the specified 8-bit operation is performed between the data in the accumulator and the source data. The result of the operation is placed in the accumulator with the exception of compare (CP) that leaves the accumulator unchanged. All these operations effect the flag register as a result of the specified operation. INC and DEC instructions specify a register or a memory location as both source and destination of the result. When the source operand is addressed using the index registers, the displacement must follow directly. With immediate addressing, the actual operand follows directly. For example, the instruction AND 07H is:

Address A	E6	Op Code
A+1	07	Operand

Assuming that the accumulator contained the value F3H, the result of 03H is placed in the accumulator:

Accumulator before operation 1111 0011 = F3H

Operand   0000 0111 = 07H

Result to Accumulator 0000 0011 = 03H

The Add instruction (ADD) performs a binary add between the data in the source location and the data in the accumulator. The Subtract (SUB) performs a binary subtraction. When the Add with Carry is specified, (ADC) or the Subtract with Carry (SBC), then the Carry flag is also added or subtracted respectively. The flags and decimal adjust instruction (DAA) in the Z80 allow arithmetic operations for:

- Multiprecision packed BCD numbers
- Multiprecision signed or unsigned binary numbers
- Multiprecision two's complement signed numbers

Other instructions in this group are logical and (AND), logical or (OR), exclusive or (XOR), and compare (CP).

Five general-purpose arithmetic instructions operate on the accumulator or carry flag. These five are listed in Table 12. The decimal adjust instruction can adjust for subtraction as well as addition, making BCD arithmetic operations simple. Note that to allow for this operation the flag N is used. This flag is set if the last arithmetic operation was a subtract. The negate accumulator (NEG) instruction forms the two's complement of the number in the accumulator. Finally, notice that a reset carry instruction is not included in the Z80 because this operation can be easily achieved through other instructions such as a logical AND of the accumulator with itself.

Table 13 lists all the 16-bit arithmetic operations between 16-bit registers. There are five groups of instructions including add with carry and subtract with carry. ADC and SBC affect all the flags. These two groups simplify address calculation operations or other 16-bit arithmetic operations.

Table 11. 8-Bit Arithmetic and Logic

	Source										
	Register Addressing							Reg Indir.	Indexed		Immed.
	A	B	C	D	E	F	L	(HL)	(IX+d)	(lY+d)	n
ADD	87	80	81	82	83	84	85	88	DD 86 d	FD 86 d	C6 n
ADD W CARRY ADC	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT SUB	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARR SBC	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
AND	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n

Table 11. 8-Bit Arithmetic and Logic (Continued)

	Source										
	Register Addressing							**Reg Indir.**	**Indexed**		**Immed.**
OR	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE CP	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT INC	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT DEC	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Table 12. General-Purpose AF Operation

Decimal Adjust Acc, DAA	27
Complement Acc, CPL	2F
Negate Acc, NEG (2's complement	ED 44
Complement Carry Flag, CCF	3F
Set Carry Flag, SCF	37

Table 13. 16-Bit Arithmetic

Source					
BC	DE	HL	SP	IX	IY

Table 13. 16-Bit Arithmetic (Continued)

Destination			Source					
		HL	09	19	29	39		
	ADD	IX	DD 09	DD 19		DD 39	DD 29	
		IY	FD 09	FD 19		FD 39		FD 29
	ADD with carry and set flags ADC	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	SUB with carry and set flags SBC	HL	ED 42	ED 52	ED 62	ED 72		
	Increment INC		03	13	23	33	DD 23	FD 23
	Decrement DEC		DB	1B	2B	3B	DD 2B	FD 2B

Rotate and Shift

A major feature of the Z80 is to rotate or shift data in the accumulator, any general-purpose register, or any memory location. All the rotate and shift Op Codes are depicted in Figure 14. Also included in the Z80 are arithmetic and logical shift operations. These operations are useful in a wide range of applications including integer multiplication and division. Two BCD digit rotate instructions (RRD and RLD) allow a digit in the accumulator to be rotated with the two digits in a memory location pointed to by register pair HL (See Figure 14). These instructions allow for efficient BCD arithmetic.

Table 14. Rotates and Shifts

Type of Rotate Shift	Source												A
	A	B	C	D	E	F	L	(HL)	(IX+d)	(lY+d)			A
RCL	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 06	CB 0E	DD CB d 06	FD CB d 06		RLCA	07
RRC	CB 0F	CB 08	CB 09	CB 0A	CB 06	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E		RRCA	0F
RL	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16		RLA	17
RR	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E		RRA	1F
SLA	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26			
SRA	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E			
SRL	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E			
									ED 6F				
									ED 67				

CY b_7 b_0 Rotate Left Circular

Rotate Right Circular

Rotate Left

Rotate Right

CY   Shift Left Arithmetic

Shift Right Arithmetic

Shift Right Logical

0

b_3-b_0 ACC b_7-b_4 b_3-b_0 (HL)   Rotate Digit Left

ACC   (HL)   Rotate Digit Right
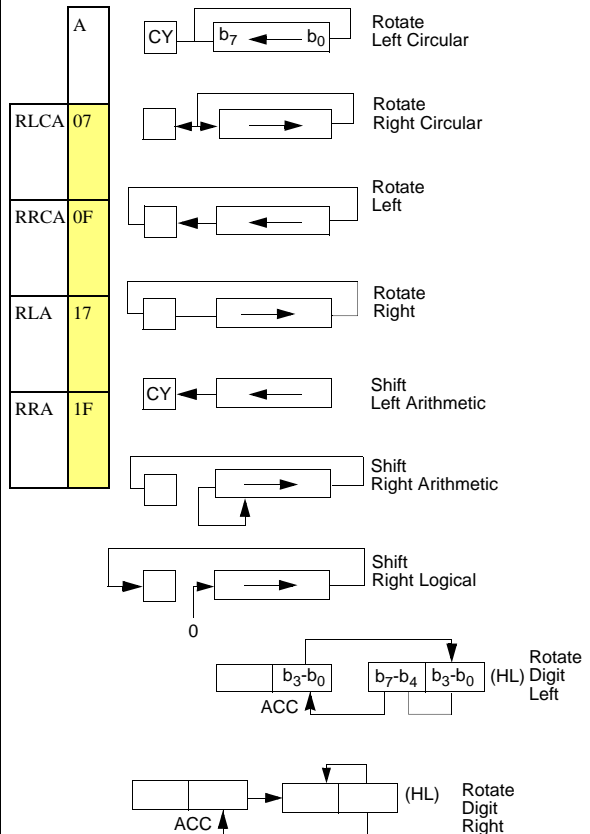
Bit Manipulation

The ability to set, reset, and test individual bits in a register or memory location is needed in almost every program. These bits may be flags in a general-purpose software routine, indications of external control

conditions, or data packed into memory locations, making memory utilization more efficient.

The Z80 can set, reset, or test any bit in the accumulator, any general-purpose register or any memory location with a single instruction. Table 15 lists the 240 instructions that are available for this purpose. Register addressing can specify the accumulator or any general-purpose register on which the operation is to be performed. Register indirect and indexed addressing are available to operate on external memory locations. Bit test operations set the Zero flag (Z) if the tested bit is a zero.

Jump, Call, and Return

Table 16 lists all the jump, call, and return instructions implemented in the Z80 CPU. A jump is a branch in a program where the program counter is loaded with the 16-bit value as specified by one of the three available addressing modes (Immediate Extended, Relative, or Register Indirect). Notice that the jump group has several conditions that can be specified before the jump is made. If these conditions are not met, the program merely continues with the next sequential instruction. The conditions are all dependent on the data in the flag register. The immediate extended addressing is used to jump to any location in the memory. This instruction requires three bytes (two to specify the 16-bit address) with the low order address byte first, followed by the high order address byte.

For example, an unconditional jump to memory location 3E32H is:

Address A	C3	Op Code
A+1	32	Low Order Address
A+2	3E	High Order Address

The relative jump instruction uses only two bytes, the second byte is a signed two's complement displacement from the existing PC. This displacement can be in the range of +129 to -126 and is measured from the address of the instruction Op Code.

Three types of register indirect jumps are also included. These instructions are implemented by loading the register pair HL or one of the index registers 1X or IY directly into the PC. This feature allows for program jumps to be a function of previous calculations.

A call is a special form of a jump where the address of the byte following the call instruction is pushed onto the stack before the jump is made. A return instruction is the reverse of a call because the data on the top of the stack is popped directly into the PC to form a jump address. The call and return instructions allow for simple subroutine and interrupt handling. Two special return instruction are included in the Z80 family of components. The return from interrupt instruction (RETI) and the return from nonmaskable interrupt (RETN) are treated in the CPU as an unconditional return identical to the Op Code C9H. The difference is that (RETI) can be used at the end of an interrupt routine and all Z80 peripheral chips recognize the execution of this instruction for proper control of nested priority interrupt handling. This instruction, coupled with the Z80 peripheral devices implementation, simplifies the normal return from nested interrupt. Without this feature, the following software sequence is necessary to inform the interrupting device that the interrupt routine is completed:

Disable Interrupt	Prevent interrupt before routine is exited.
LD A, n	Notify peripheral that service
OUT n, A	routine is complete.
Enable Interrupt	
Return	

This seven byte sequence can be replaced with the one byte EI instruction and the two byte RETI instruction in the Z80. This is important because interrupt service time often must be minimized.

Table 15. Bit Manipulation Group

Register Addressing								Reg. Indir.	Indexed	
A	8	C	D	E	H	L		(HL)	(IX+d)	(IY+d)
Bit									DD	FD

Table 15. Bit Manipulation Group (Continued)

		Register Addressing							Reg. Indir.	Indexed	
Test Bit	0	C8 47	C8 40	C8 41	C8 42	C8 43	C8 44	C8 45	C8 46	C8 d 46 DD	C8 d 46 FD
	1	C8 4F	C8 48	C8 49	C8 4A	C8 48	C8 4C	C8 4D	C8 4E	C8 d 4E	C8 d 4E
	2	C8 57	C8 50	C8 51	C8 52	C8 53	C8 54	C8 55	C8 56	DD C8 d 56	FD C8 d 56
	3	C8 5F	C8 58	C8 59	C8 5A	C8 5B	C8 5C	C8 5D	C8 5E	DD C8 d 46	FD C8 d 46
	4	C8 67	C8 60	C8 61	C8 62	C8 63	C8 64	C8 65	C8 66	DD C8 d 66	FD C8 d 66
	5	C8 6F	C8 68	C8 69	C8 6A	C8 68	C8 6C	C8 6D	C8 6E	DD C8 d 6E	FD C8 d 6E
	6	C8 77	C8 70	C8 71	C8 72	C8 73	C8 74	C8 75	C8 76	DD C8 d 76	FD C8 d 76
	7	C8 7F	C8 78	C8 79	C8 7A	C8 78	C8 7C	CS 7D	C8 7E	DD C8 d 46	DD C8 d 46

Table 15. Bit Manipulation Group (Continued)

		Register Addressing								Reg. Indir.	Indexed	
Rest Bit RES	0	C8 87	C8 80	C8 81	C8 82	C8 83	C8 84	C8 85		C8 86	DD C8 d 86	FD C8 d 86
	1	C8 8F	C8 88	C8 89	C8 8A	C8 88	C8 8C	C8 8D		C8 8E	DD C8 d 8E	FD C8 d 8E
	2	C8 97	C8 90	CS 91	C8 92	C8 93	C8 94	C8 95		C8 96	DD C8 d 96	FD C8 d 96
	3	C8 9F	C8 98	C8 99	C8 9A	CS 98	C8 90	C8 90		C8 9E	DD C8 d 9E	FD C8 d 9E
	4	C8 A7	C8 AO	C8 AI	C8 A2	C6 A3	C8 A4	C8 A5		C8 A6	DD C8 d A6	FD C8 d A6
	5	C8 AF	C8 A8	C8 A9	C8 AA	08 AB	C8 AC	C8 AD		C8 AE	DD C8 d AE	FD C8 d AE
	6	C8 B7	C8 B0	C8 B1	C8 82	C8 B3	C8 B4	C8 B5		C8 B6	DD C8 d B6	FD C8 d B6
	7	C8 BF	C8 B8	C8 89	C8 8A	C8 B8	C8 8C	C8 BD		C8 9E	DD C8 d BE	DD C8 d BE

Table 15. Bit Manipulation Group (Continued)

		Register Addressing								Reg. Indir.	Indexed	
Set Bit SET	0	C8 C7	C8 C0	C8 C1	C8 C2	C8 C3	C8 C4	C8 C5	C8 C6	DD C8 d C6	FD C8 d C6
	1	C8 CF	C8 C8	C8 C9	C8 CA	C8 C8	C8 CC	C8 CD	C8 CE	DD C8 d CE	FD C8 d CE
	2	C8 D7	C8 DO	C8 D1	C8 D2	C8 D3	C8 D4	C8 DS	C8 D6	DD C8 d D6	FD C8 d D6
	3	C8 DF	C8 D8	C8 09	C8 DA	C8 DS	C8 DC	C8 DD	C8 DE	DD C8 d DE	FD C8 d DE
	4	C8 E7	C8 E0	C8 E1	C8 E2	C8 E3	C8 E4	C8 E5	C8 E6	DD C8 d E6	FD C8 d E6
	5	C8 EF	C8 E8	C8 E9	C8 EA	C8 EB	C8 EC	C8 ED	C8 EE	DD C8 d EE	FD C8 d EE
	6	C8 F7	C8 FO	C8 F1	C8 F2	C8 F3	C8 F4	C8 FS	C8 F6	DD C8 d F6	FD C8 d F6
	7	C8 FF	C8 F8	C8 F9	C8 FA	C8 FB	C8 FC	C8 FD	C8 FE	DD C8 d FE	FD C8 d FE

Table 16. Jump, Call, and Return Group

			Condition									
			Un-Cond.	Carry	Non Carry	Zero	Non Zero	Parity Even	Parity Odd	Sign Neg	Sign Pos	Reg B≠0
JUMP JP	IMMED. EXT.	nn	C3 n n	D8 n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP JR	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP JP	Register INDIR.	(HL)	EB									
		(IX)	DD E9									
		(IY)	FD E9									
CALL	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
Decrement B, Jump If Non Zero DJNZ	RELATIVE	PC+e										10 e-2
Return RE	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
Return From INT RETI			ED 4D									
Return From Non Maskable INT RETN			ED 45									

The instruction DJNZ is used to facilitate program loop control. This two byte, relative jump instruction decrements the B register and the jump occurs if the B register has not been decremented to zero. The relative displacement is expressed as a signed two's complement number. A simple example of its use is:

Address	Instruction	Comments
N, N+1	LD B, 7	: set B register to count of 7
N+2 to N+9	(Perform a sequence of instructions)	: loop to be performed 7 times
N+10,N+11	DJNZ -8	: to jump from N+12 to N+2
N + 12	(Next Instruction)	

Table 17 lists the eight Op Codes for the restart instruction. This
instruction is a single byte call to any of the eight addresses listed. The
simple mnemonic for these eight calls is also listed. This instruction is
useful for frequently-used routines because memory consumption is
minimized.

Table 17. Restart Group

		Op Code	
CALL Address	0000H	C7	RST 0
	0008H	CF	RST 8
	0010H	D7	RST 16
	0018H	DF	RST 24
	0020H	E7	RST 32
	0028H	EF	RST 40
	0030H	F7	RST 48
	0038H	FF	RST 56

Input/Output

The Z80 has an extensive set of input and output instructions as shown in
Table 18 and Table 19. The addressing of the input or output device can
be either absolute or register indirect, using the C register. In the register
indirect addressing mode, data can be transferred between the I/O devices
and any of the internal registers. In addition, eight block transfer
instructions have been implemented. These instructions are similar to the
memory block transfers except that they use register pair HL for a pointer
to the memory source (output commands) or destination (input
commands) while register B is used as a byte counter. Register C holds
the address of the port for which the input or output command is required.
Because register B is eight bits in length, the I/O block transfer command
handles up to 256 bytes.

In the instructions IN A, and OUT n, A, the I/O device address n appears in
the lower half of the address bus (A7-A0) while the accumulator content

is transferred in the upper half of the address bus. In all register indirect input output instructions, including block I/O transfers, the content of register C is transferred to the lower half of the address bus (device address) while the content of register B is transferred to the upper half of the address bus.

CPU Control Group

Table 20 illustrates the six general-purpose CPU control instructions. The NOP is a do-nothing instruction. The HALT instruction suspends CPU operation until a subsequent interrupt is received, while the DI and E1 are used to lock out and enable interrupts. The three interrupt mode commands set the CPU to any of the three available interrupt response modes as follows. If Mode 0 is set, the interrupting device can insert any instruction on the data bus and allow the CPU to execute it. Mode 1 is a simplified mode where the CPU automatically executes a restart (RST) to location 0038H so that no external hardware is required (the old PC content is pushed onto the stack). Mode 2 is the most powerful because it allows for an indirect call to any location in memory. With this mode, the CPU forms a 16-bit memory address where the upper eight bits are the content of register I and the lower eight bits are supplied by the interrupting device. This address points to the first of two sequential bytes in a table where the address of the service routine is located. The CPU automatically obtains the starting address and performs a CALL instruction to this address.

Address of Interrupt
Service Routine

Pointer to Interrupt Table,
Register I is Upper Address,
Peripheral Supplies
Lower Address
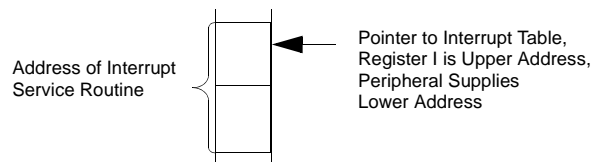
Table 18. Input Group

			Immed.	Register Indir.	
			(n)	(c)	
Input Destination	Input IN	Register Address	A	DB n	ED 7B	
			B		ED 40	
			C		ED 48	
			D		ED 50	
			E		ED 58	
			H		ED 60	
			L		ED 68	
	INI - input & inc HL, Dec B	Register Indir	(HL)		ED A2	Block Input Commands
	INIR - INP, Inc HL, Dec B, repeat IF B≠0				ED B2	
	IND - input & Inc Dec HL, Dec B				ED AA	
	INDR - input, Dec HL, Dec B, repeat IF B≠0				ED BA	

Table 19. 8-Bit Arithmetic and Logic

			Source									
			Register							Register Indir.		
			A	B	C	D	E	H	L	(HL)		
11OUT	Immed.	(n)	D3 n									
	Reg Ind.	(c)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69			
11OUT - output inc HL, dec B										ED A3	Block Output Command
11OUT - output dec B, repeat if B≠0										ED B3	
11OUT - output dec HL and B										ED AB	
11OUTDR - output, dec HL and B, repeat IF B≠0										ED BB	
	Port Destination Address										

Table 20. Miscellaneous CPU Control

NOP	00	
HALT	76	
Disable INT (EI)	F3	
Enable INT (EI)	FB	
Set INT mode 0 IM0	ED 46	8080A mode
Set INT mode 1 IM1	ED 56	Call to location 0038H
Set INT mode 2 IM2	ED 5E	indirect call using register I and B bits from INTER device as a pointer

Z80 Instruction Set

Z80 Assembly Language

The assembly language allows the user to write a program without concern for memory addresses or machine instruction formats. It uses symbolic addresses to identify memory locations and mnemonic codes (Op Codes and operands) to represent the instructions. Labels (symbols) are assigned to a particular instruction step in a source program to identify that step as an entry point for use in subsequent instructions. Operands following each instruction represent storage locations, registers, or constant values. The assembly language also includes assembler directives that supplement the machine instruction. A pseudo-op, for example, is a statement that is not translated to a machine instruction, but rather is interpreted as a directive that controls the assembly process.

A program written in assembly language is called a source program, which consists of symbolic commands called statements. Each statement is written on a single line and may consist of from one to four entries: A label field, an operation field, an operand field, and a comment field. The source program is processed by the assembler to obtain a machine language program (object program) that can be executed directly by the Z80 CPU.

ZiLOG provides several assemblers that differ in the features offered. Both absolute and relocatable assemblers are available with the Development and Micro-computer Systems. The absolute assembler is contained in base level software operating in a 16K memory space, while the relocating assembler is part of the RIO environment operating in a 32K memory space.

Z80 Status Indicator Flags

The flag registers (F and F') supply information to the user about the status of the Z80 at any given time. The bit positions for each flag is listed below:

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

Symbol	Field Name
C	Carry Flag
N	Add/Subtract
P/V	Parity/Overflow Flag
H	Half Carry Flag
Z	Zero Flag
S	Sign Flag
X	Not Used

Each of the two flag registers contains 6 bits of status information that are set or cleared by CPU operations. (Bits 3 and 5 are not used.) Four of these bits (C, P/V, Z, and S) may be tested for use with conditional JUMP, CALL, or RETURN instructions. Two flags may not be tested (H, N) and are used for BCD arithmetic.

Carry Flag

The Carry Flag (C) is set or cleared depending on the operation performed. For ADD instructions that generate a Carry, and SUB instructions that generate a Borrow, the Carry Flag sets. The Carry Flag is reset by an ADD instruction that does not generate a Carry, and by a SUB instruction that does not generate a Borrow. This saved Carry facilitates software routines

for extended precision arithmetic. Also, the DAA instruction sets the Carry Flag if the conditions for making the decimal adjustment are met.

For instructions RLA, RRA, RLS, and RRS, the Carry bit is used as a link between the least significant byte (LSB) and most significant byte (MSB) for any register or memory location. During instructions RLCA, RLC, and SLA, the Carry contains the last value shifted out of Bit 7 of any register or memory location. During instructions RRCA, RRC, SRA, and SRL, the Carry contains the last value shifted out of Bit 0 of any register or memory location.

For the logical instructions AND, OR, and XOR, the Carry is reset.

The Carry Flag can also be set by the Set Carry Flag (SCF) and complemented by the Compliment Carry Flag (CCF) instructions.

Add/Subtract Flag

The Add/Subtract Flag (N) is used by the Decimal Adjust Accumulator instruction (DAA) to distinguish between ADD and SUB instructions. For ADD instructions, N is cleared to 0. For SUB instructions, N is set to 1.

Add/Subtract Flag

The Decimal Adjust Accumulator instruction (DAA) uses this flag to distinguish between ADD and SUBTRACT instructions. For all ADD instructions, N sets to 0. For all SUBTRACT instructions, N sets to 1.

Parity/Overflow Flag (P/V)

This flag is set to a specific state 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). This Overflow condition is determined by examining the sign bits of the operands.

For addition, operands with different signs never cause Overflow. When adding operands with like signs and the result has a different sign, the Overflow Flag is set, for example:

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

The two numbers added together resulted in a number that exceeds +127 and the two positive operands have resulted in a negative number (-95), which is incorrect. The Overflow Flag is therefore set.

For subtraction, Overflow can occur for operands of unlike signs. Operands of like signs never cause Overflow. For example:

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

The minuend sign has changed from a Positive to a negative, giving an incorrect difference. Overflow is set.

Another method for identifying an Overflow is to observe the Carry to and out of the sign bit. If there is a Carry in and no Carry out, or if there is no Carry in and a Carry out, then Overflow has occurred.

This flag is also used with logical operations and rotate instructions to indicate the resulting parity is Even. The number of 1 bits in a byte are counted. If the total is Odd, ODD parity is flagged (P = 0). If the total is Even, EVEN parity is flagged (P = 1).

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, if the byte counter decrements to 0, the flag is cleared to 0, otherwise the flag is set to 1.

During LD A, I and LD A, R instructions, the P/V Flag is set with the value of the interrupt enable flip-flop (IFF2) for storage or testing.

When inputting a byte from an I/O device with an IN r, (C), instruction, the P/V Flag is adjusted to indicate the data parity.

Half Carry Flag

The Half-Carry Flag (H) is set (1) or cleared (0) 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 add or subtract operation. The H Flag is set (1) or cleared (0) according to the following table:

H Flag	Add	Subtract
1	A Carry occurs from Bit 3 to Bit 4	A Borrow from Bit 4 occurs
0	No Carry occurs from Bit 3 to Bit 4	No Borrow from Bit 4 occurs

Zero Flag

The Zero Flag (Z) is set (1) or cleared (0) 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 a 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 (Search) instructions, the Z flag is set to 1 if the value in the Accumulator is equal to the value in the memory location indicated by the value of the Register pair HL.

When testing a bit in a register or memory location, the Z flag contains the complemented state of the indicated bit (see "Bit b, s").

When inputting or outputting a byte between a memory location and an I/O device (INI, IND, OUTI, and OUTD), if the result of decrementing the B Register is 0, the Z flag is 1, otherwise the Z flag is 0. Also for byte inputs from I/O devices using IN r, (C), the Z flag is set to indicate a 0-byte input.

Sign Flag

The Sign Flag (S) stores the state of the most-significant bit of the Accumulator (bit 7). When the Z80 performs arithmetic operations on signed numbers, the binary twos-complement notation is used to represent and process numeric 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 to 6 for a total range of from 0 to 127. A negative number is represented by the twos complement of the equivalent positive number. The total range for negative numbers is from −1 to −128.

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

Z80 Instruction Description

Execution time (E.T.) for each instruction is given in microseconds for an assumed 4 MHz clock. Total machine cycles (M) are indicated with total clock periods (T States). Also indicated are the number of T States for each M cycle. For example:

M Cycles: 2T States: 7(4,3) 4 MHzE.T.: 1.75

indicates that the instruction consists of 2 machine cycles. The first cycle contains 4 clock periods (T States). The second cycle contains 3 clock periods for a total of 7 clock periods or T States. The instruction executes in 1.75 microseconds.

Register format is indicated for each instruction with the most-significant bit to the left and the least-significant bit to the right.

8-Bit Load Group

LD r, r'

Operation: r, ← r'

Op Code: LD

Operands: r, r'

0	1	◄── r ──►	◄── r' ──►
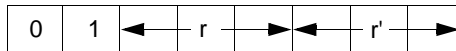
Description: The contents of any register r' are loaded to any other register r. r, r'
identifies any of the registers A, B, C, D, E, H, or L, assembled as follows
in the object code:

Register	r, C
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	MHz E.T.
1	4	1.0

Condition Bits Affected: None

Example: If the H register contains the number 8AH, and the E register contains 10H,
the instruction LD H, E results in both registers containing 10H.

LD r,n

Operation: r ← n

Op Code: LD

Operands: r, n

0	0	◄── r ──►	1	1	0
◄────── n ──────►					
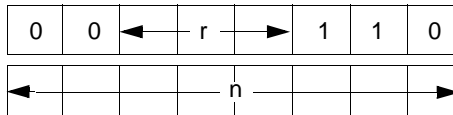
Description: The 8-bit integer n is loaded to any register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: At execution of LD E, A5H the contents of register E are A5H.

LD r, (HL)

Operation: r ← (HL)

Op Code: LD

Operands: r, (HL)

0	1	◄──── r ────►	1	1	0
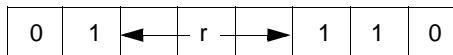
Description: The 8-bit contents of memory location (HL) are loaded to register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If register pair HL contains the number 75A1H, and memory address 75A1H contains byte 58H, the execution of LD C, (HL) results in 58H in register C.

LD r, (IX+d)

Operation: r ← (IX+d)

Op Code: LD

Operands: r, (IX+d)

1	1	0	1	1	1	0	1	DD

0	1	◄── r ──►	1	1	0

◄──────────── d ────────────►
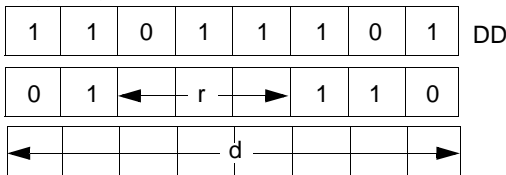
Description: The operand (IX+d), (the contents of the Index Register IX summed with a two's complement displacement integer d) is loaded to register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	**r**
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	**T States**	**4 MHz E.T.**
5	19 (4, 4, 3, 5, 3)	2.50

Condition Bits Affected: None

Example: If the Index Register IX contains the number 25AFH, the instruction LD B, (IX+19H) causes the calculation of the sum 25AFH + 19H, which points to memory location 25C8H. If this address contains byte 39H, the instruction results in register B also containing 39H.

LD r, (IY+d)

Operation: r ← (IY+D)

Op Code: LD

Operands: r, (lY+d)

1	1	1	1	1	1	1	1	FD

0	1	◄—— r ——►	1	1	0

◄—————————— d —————————►
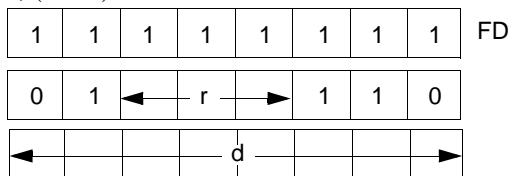
Description: The operand (lY+d) (the contents of the Index Register IY summed with a two's complement displacement integer (d) is loaded to register r, where r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected: None

Example: If the Index Register IY contains the number 25AFH, the instruction LD B, (IY+19H) causes the calculation of the sum 25AFH + 19H, which points to memory location 25C8H. If this address contains byte 39H, the instruction results in register B also containing 39H.

LD (HL), r

Operation: (HL) ← r

Op Code: LD

Operands: (HL), r
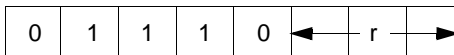
0	1	1	1	0	◄──── r ────►

Description: The contents of register r are loaded to the memory location specified by the contents of the HL register pair. The symbol r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If the contents of register pair HL specifies memory location 2146H, and the B register contains byte 29H, at execution of LD (HL), B memory address 2146H also contains 29H.

LD (IX+d), r

Operation: (IX+d) ← r

Op Code: LD

Operands: (IX+d), r
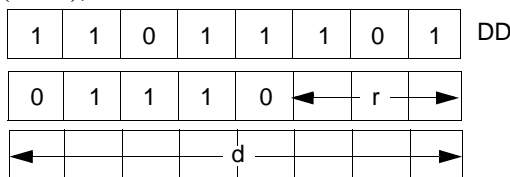
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | DD |

| 0 | 1 | 1 | 1 | 0 | ◄— | r | —► |

| ◄— | | | d | | | —► |

Description: The contents of register r are loaded to the memory address specified by the contents of Index Register IX summed with d, a two's complement displacement integer. The symbol r identifies register A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected: None

Example: If the C register contains byte 1CH, and the Index Register IX contains 3100H, then the instruction LID (IX+6H), C performs the sum 3100H + 6H and loads 1CH to memory location 3106H.

LD (IY+d), r

Operation: (lY+d) ← r

Op Code: LD

Operands: (lY+d), r
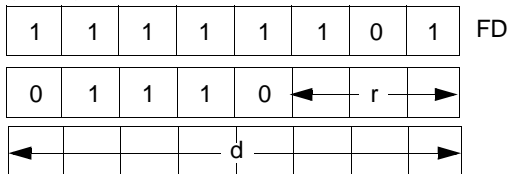
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | FD |

| 0 | 1 | 1 | 1 | 0 | ← | r | → |

| ← | | | d | | | → |

Description: The contents of resister r are loaded to the memory address specified by the sum of the contents of the Index Register IY and d, a two's complement displacement integer. The symbol r is specified according to the following table.

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected: None

Example: If the C register contains byte 48H, and the Index Register IY contains 2A11H, then the instruction LD (IY+4H),C performs the sum 2A11H + 4H, and loads 48H to memory location 2A15.

LD (HL), n

Operation: (HL) ← n

Op Code: LD

Operands: (HL), n
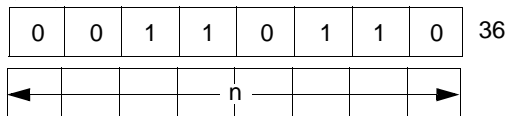
0	0	1	1	0	1	1	0	36

←				n			→

Description: Integer n is loaded to the memory address specified by the contents of the HL register pair.

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

Condition Bits Affected: None

Example: If the HL register pair contains 4444H, the instruction LD (HL), 28H results in the memory location 4444H containing byte 28H.

LD (IX+d), n

Operation: (IX+d) ← n

Op Code: LD

Operands: (IX+d), n

| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | DD |

| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 36 |

◄——————— d ———————►

◄——————— n ———————►
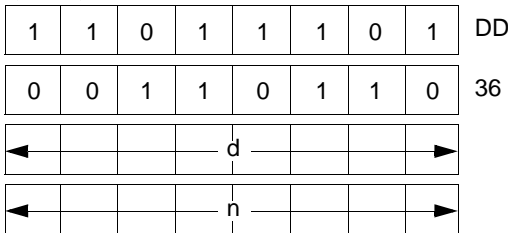
Description: The n operand is loaded to the memory address specified by the sum of the Index Register IX and the two's complement displacement operand d.

M Cycles	T States	4 MHz E.T.
5	19 (4, 4, 3,5,3)	4.75

Condition Bits Affected: None

Example: If the Index Register IX contains the number 219AH, the instruction LD (IX+5H),5AH results in byte 5AH in the memory address 219FH.

LD (lY+d), n

Operation: (lY+d) ← n

Op Code: LD

Operands: (lY+d), n
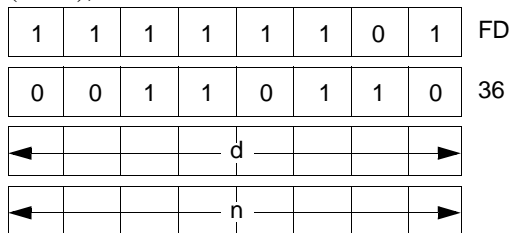
1	1	1	1	1	1	0	1	FD

0	0	1	1	0	1	1	0	36

←				d			→

←				n			→

Description: Integer n is loaded to the memory location specified by the contents of the Index Register summed with the two's complement displacement integer d.

M Cycles	T States	4 MHz E.T.
5	19 (4, 4, 3, 5, 3)	2.50

Condition Bits Affected: None

Example: If the Index Register IY contains the number A940H, the instruction LD (IY+10H), 97H results in byte 97H in memory location A950H.

LD A, (BC)

Operation: A ← (BC)

Op Code: LD

Operands: A, (BC)

0	0	0	0	1	0	1	0	0A

Description: The contents of the memory location specified by the contents of the BC register pair are loaded to the Accumulator.

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If the BC register pair contains the number 4747H, and memory address 4747H contains byte 12H, then the instruction LD A, (BC) results in byte 12H in register A.

LD A, (DE)

Operation: A ← (DE)

Op Code: LD

Operands: A, (DE)

0	0	0	1	1	0	1	0	1A

Description: The contents of the memory location specified by the register pair DE are loaded to the Accumulator.

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If the DE register pair contains the number 30A2H and memory address 30A2H contains byte 22H, then the instruction LD A, (DE) results in byte 22H in register A.

LD A, (nn)

Operation: A ← (nn)

Op Code: LD

Operands: A, (nn)

| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 3A |

◄——————— n ———————►

◄——————— n ———————►
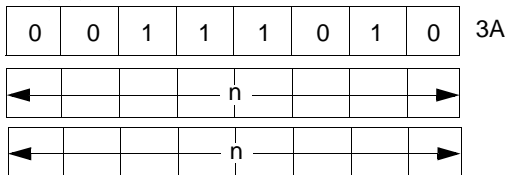
Description: The contents of the memory location specified by the operands nn are loaded to the Accumulator. The first n operand after the Op Code is the low order byte of a 2-byte memory address.

M Cycles	T States	4 MHz E.T.
4	13 (4, 3, 3, 3)	3.25

Condition Bits Affected: None

Example: If the contents of nn is number 8832H, and the content of memory address 8832H is byte 04H, at instruction LD A, (nn) byte 04H is in the Accumulator.

LD (BC), A

Operation: (BC) ← A

Op Code: LD

Operands: (BC), A

0	0	0	0	0	0	1	0	02

Description: The contents of the Accumulator are loaded to the memory location specified by the contents of the register pair BC.

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If the Accumulator contains 7AH and the BC register pair contains 1212H the instruction LD (BC), A results in 7AH in memory location 1212H.

LD (DE), A

Operation: (DE) ← A

Op Code: LD

Operands: (DE), A

0	0	0	1	0	0	1	0	12

Description: The contents of the Accumulator are loaded to the memory location specified by the contents of the DE register pair.

M cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: If the contents of register pair DE are 1128H, and the Accumulator contains byte A0H, the instruction LD (DE), A results in A0H in memory location 1128H.

LD (nn), A

Operation: $(nn) \leftarrow A$

Op Code: LD

Operands: (nn), A
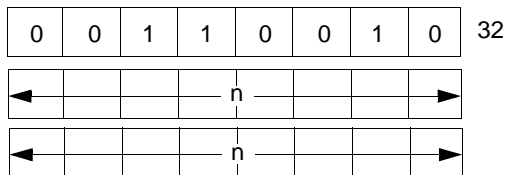
0	0	1	1	0	0	1	0	32

◄─────── n ───────►

◄─────── n ───────►

Description: The contents of the Accumulator are loaded to the memory address specified by the operand nn. The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
4	13 (4, 3, 3, 3)	3.25

Condition Bits Affected: None

Example: If the contents of the Accumulator are byte D7H, at execution of LD (3141 H), AD7H results in memory location 3141H.

LD A, I

Operation: $A \leftarrow 1$

Op Code: LD

Operands: A, I

1	1	1	0	1	1	0	1	ED

0	1	0	1	0	1	1	1	57

Description: The contents of the Interrupt Vector Register I are loaded to the Accumulator.

M Cycles	T States	MHz E.T.
2	9 (4, 5)	2.25

Condition Bits Affected:

S is set if I-Register is negative; reset otherwise

Z is set if I-Register is zero; reset otherwise

H is reset

P/V contains contents of IFF2

N is reset

C is not affected

If an interrupt occurs during execution of this instruction, the Parity flag contains a 0.

LD A, R

Operation: A, ← R

Op Code: LD

Operands: A, R

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | ED |

| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5F |

Description: The contents of Memory Refresh Register R are loaded to the Accumulator.

M Cycles	**T States**	**MHz E.T.**
2	9 (4, 5)	2.25

Condition Bits Affected:

S is set if, R-Register is negative; reset otherwise

Z is set if R-Register is zero; reset otherwise

H is reset

P/V contains contents of IFF2

N is reset

C is not affected

If an interrupt occurs during execution of this instruction, the parity flag contains a 0.

LD I,A

Operation: I ← A

Op Code: LD

Operands: I, A

1	1	1	0	1	1	0	1	ED

0	1	0	0	0	1	1	1	47

Description: The contents of the Accumulator are loaded to the Interrupt Control Vector Register, I.

M Cycles	T States	MHz E.T.
2	9 (4, 5)	2.25

Condition Bits Affected: None

LD R, A

Operation: R ← A

Op Code: LD

Operands: R, A

1	1	1	0	1	1	0	1	ED

0	1	0	0	1	1	1	1	4F

Description: The contents of the Accumulator are loaded to the Memory Refresh register R.

M Cycles	T States	MHz E.T.
2	9 (4, 5)	2.25

Condition Bits Affected: None

16-Bit Load Group

LD dd, nn

Operation: dd ← nn

Op Code: LD

Operands: dd, nn

0	0	d	d	0	0	0	1

| ◄ | | | n | | | | ► |

| ◄ | | | n | | | | ► |
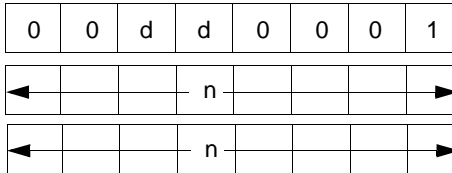
Description: The 2-byte integer nn is loaded to the dd register pair, where dd defines the BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the Op Code is the low order byte.

M Cycles	T States	4 MHz E.T.
2	10 (4, 3, 3)	2.50

Condition Bits Affected: None

Example: At execution of LD HL, 5000H the contents of the HL register pair is 5000H.

LD IX, nn

Operation: Ix ← nn

Op Code: LD

Operands: IX, nn

1	1	0	1	1	1	0	1	DD

0	0	1	0	0	0	0	1	21

◄──────── n ────────►

◄──────── n ────────►
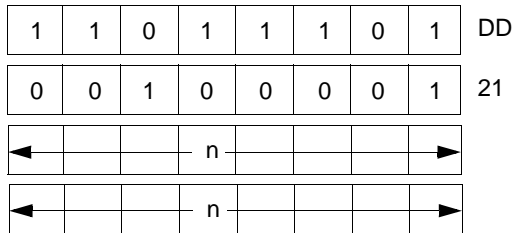
Description: Integer nn is loaded to the Index Register IX. The first n operand after the Op Code is the low order byte.

M Cycles	T States	4 MHz E.T.
4	14 (4, 4, 3, 3)	3.50

Condition Bits Affected: None

Example: At instruction LD IX, 45A2H the Index Register contains integer 45A2H.

LD IY, nn

Operation: IY ← nn

Op Code: LD

Operands: IY, nn

1	1	1	1	1	1	0	1	FD

0	0	1	0	0	0	0	1	21

◄			n				►

◄			n				►
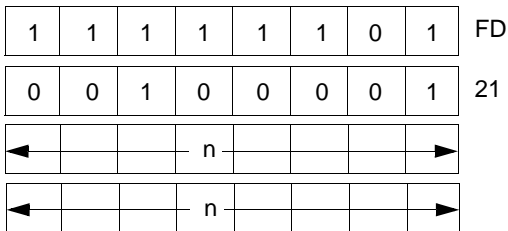
Description: Integer nn is loaded to the Index Register IY. The first n operand after the
Op Code is the low order byte.

M Cycles	T States	4 MHz E.T.
4	14 (4, 4, 3, 3)	3.50

Condition Bits Affected: None

Example: At instruction LD IY, 7733H the Index Register IY contains the integer
7733H.

LD HL, (nn)

Operation: H ← (nn+1), L ← (nn)

Op Code: LD

Operands: HL, (nn)

0	0	1	0	1	0	1	0	2A

◄—			n			—►

◄—			n			—►
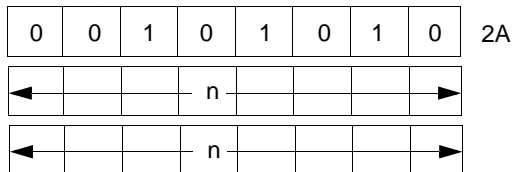
Description: The contents of memory address (nn) are loaded to the low order portion of register pair HL (register L), and the contents of the next highest memory address (nn+1) are loaded to the high order portion of HL (register H). The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
5	16 (4, 3, 3, 3, 3)	4.00

Condition Bits Affected: None

Example: If address 4545H contains 37H, and address 4546H contains A1H, at instruction LD HL, (4545H) the HL register pair contains A137H.

LD dd, (nn)

Operation: ddh ← (nn+1) ddl ← (nn)

Op Code: LD

Operands: dd, (nn)
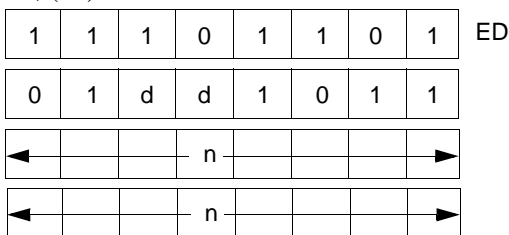
1	1	1	0	1	1	0	1	ED

0	1	d	d	1	0	1	1

◄─────── n ───────►

◄─────── n ───────►

Description: The contents of address (nn) are loaded to the low order portion of register pair dd, and the contents of the next highest memory address (nn+1) are loaded to the high order portion of dd. Register pair dd defines BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the Op Code is the low order byte of (nn).

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If Address 2130H contains 65H, and address 2131M contains 78H, at instruction LD BC, (2130H) the BC register pair contains 7865H.

LD IX, (nn)

Operation: IXh ← (nn+1), IXl ← (nn)

Op Code: LD

Operands: IX, (nn)

1	1	0	1	1	1	0	1	DD

0	0	1	0	1	0	1	0	2A

◄				n				►

◄				n				►
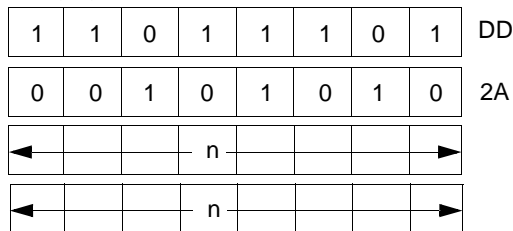
Description: The contents of the address (nn) are loaded to the low order portion of Index Register IX, and the contents of the next highest memory address (nn+1) are loaded to the high order portion of IX. The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If address 6666H contains 92H, and address 6667H contains DAH, at instruction LD IX, (6666H) the Index Register IX contains DA92H.

LD IY, (nn)

Operation: $IYh \leftarrow (nn+1), IYI \leftarrow nn)$
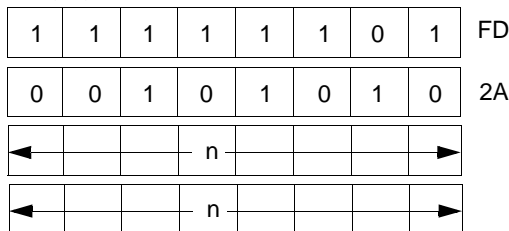
Op Code: LD

Operands: IY, (nn)

1	1	1	1	1	1	0	1	FD

0	0	1	0	1	0	1	0	2A

◄─────────── n ───────────►

◄─────────── n ───────────►

Description: The contents of address (nn) are loaded to the low order portion of Index Register IY, and the contents of the next highest memory address (nn+1) are loaded to the high order portion of IY. The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If address 6666H contains 92H, and address 6667H contains DAH, at instruction LD IY, (6666H) the Index Register IY contains DA92H.

LD (nn), HL

Operation: $(nn+1) \leftarrow H, (nn) \leftarrow L$

Op Code: LD

Operands: (nn), HL

0	0	1	0	0	0	1	0	22

←			n				→	

←			n				→	
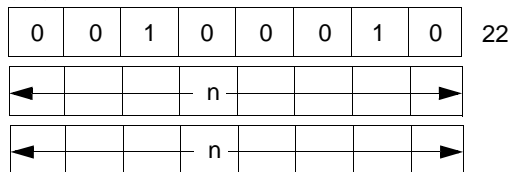
Description: The contents of the low order portion of register pair HL (register L) are loaded to memory address (nn), and the contents of the high order portion of HL (register H) are loaded to the next highest memory address (nn+1). The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
5	16 (4, 3, 3, 3, 3)	4.00

Condition Bits Affected: None

Example: If the content of register pair HL is 483AH, at instruction LD (B2291-1), HL address B229H contains 3AH, and address B22AH contains 48H.

LD (nn), dd

Operation: $(nn+1) \leftarrow ddh, (nn) \leftarrow ddl$

Op Code: LD

Operands: (nn), dd

1	1	1	0	1	1	0	1	ED

0	1	d	d	0	0	1	1

◄—				n			—►

◄—				n			—►
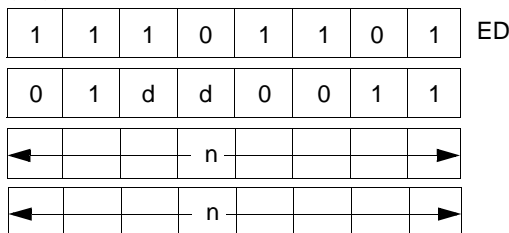
Description: The low order byte of register pair dd is loaded to memory address (nn); the upper byte is loaded to memory address (nn+1). Register pair dd defines either BC, DE, HL, or SP, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the Op Code is the low order byte of a two byte memory address.

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If register pair BC contains the number 4644H, the instruction LD (1000H), BC results in 44H in memory location 1000H, and 46H in memory location 1001H.

LD (nn), IX

Operation: $(nn+1) \leftarrow IXh, (nn) \leftarrow IXl$

Op Code: LD

Operands: (nn), IX

1	1	0	1	1	1	0	1	DD

0	0	1	0	0	0	1	0	22

←				n			→

←				n			→
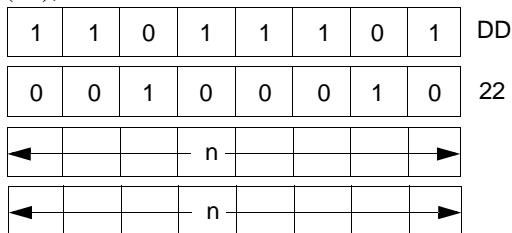
Description: The low order byte in Index Register IX is loaded to memory address (nn); the upper order byte is loaded to the next highest address (nn+1). The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If the Index Register IX contains 5A30H, at instruction LD (4392H), IX memory location 4392H contains number 30H, and location 4393H contains 5AH.

LD (nn), IY

Operation: $(nn+1) \leftarrow IYh, (nn) \leftarrow IYl$

Op Code: LD

Operands: (nn), IY

1	1	1	1	1	1	0	1	FD

0	0	1	0	0	0	1	0	22

◄───			n			───►		

◄───			n			───►		
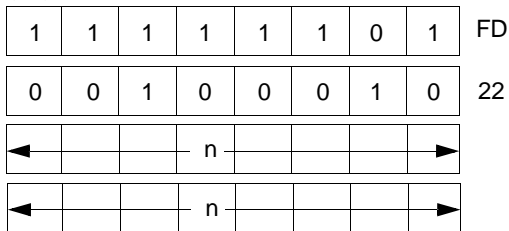
Description: The low order byte in Index Register IY is loaded to memory address (nn); the upper order byte is loaded to memory location (nn+1). The first n operand after the Op Code is the low order byte of nn.

M Cycles	T States	4 MHz E.T.
6	20 (4, 4, 3, 3, 3, 3)	5.00

Condition Bits Affected: None

Example: If the Index Register IY contains 4174H at instruction LD (8838H), IY memory location 8838H contains number 74H, and memory location 8839H contains 41H.

LD SP, HL

Operation: $SP \leftarrow HL$

Op Code: LD

Operands: SP, HL

1	1	1	r	1	0	0	1	F9

Description: The contents of the register pair HL are loaded to the Stack Pointer (SP).

M Cycles	T States	4 MHz E.T.
1	6	1.5

Condition Bits Affected: None

Example: If the register pair HL contains 442EH, at instruction LD SP, HL the Stack Pointer also contains 442EH.

LD SP, IX

Operation: SP ← - IX

Op Code: LD

Operands: SP, 1X

1	1	0	1	1	1	0	1	DD

1	1	1	1	1	0	0	1	F9

Description: The 2-byte contents of Index Register IX are loaded to the Stack Pointer (SP).

M Cycles	T States	4 MHz E.T.
2	10 (4, 6)	2.50

Condition Bits Affected: None

Example: If the contents of the Index Register IX are 98DAH, at instruction
LD SP, IX the contents of the Stack Pointer are also 98DAH.

LD SP, IY

Operation: SP ← IY

Op Code: LD

Operands: SP, IY

1	1	1	1	1	1	0	1	FD

1	1	1	1	1	0	0	1	F9

Description: The 2-byte contents of Index Register IY are loaded to the Stack Pointer SP.

M Cycles	T States	4 MHz E.T.
2	10 (4, 6)	2.50

Condition Bits Affected: None

Example: If Index Register IY contains the integer A227H, at instruction LD SP, IY the Stack Pointer also contains A227H.

PUSH qq

Operation: (SP-2) ← qqL, (SP-1) ← qqH

Op Code: PUSH

Operands: qq

1	1	q	q	0	1	0	1

Description: The contents of the register pair qq are pushed to the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current top of the Stack. This instruction first decrements SP and loads the high order byte of register pair qq to the memory address specified by the SP. The SP is decremented again and loads the low order byte of qq to the memory location corresponding to this new address in the SP. The operand qq identifies register pair BC, DE, HL, or AF, assembled as follows in the object code:

Pair	qq
BC	00
DE	01
HL	10
AF	11

M Cycles	T States	4 MHz E.T.
3	11 (5, 3, 3)	2.75

Condition Bits Affected: None

Example: If the AF register pair contains 2233H and the Stack Pointer contains 1007H, at instruction PUSH AF memory address 1006H contains 22H, memory address 1005H contains 33H, and the Stack Pointer contains 1005H.

PUSH IX

Operation: $(SP-2) \leftarrow IXL, (SP-1) \leftarrow IXH$

Op Code: PUSH

Operands: IX

1	1	0	1	1	1	0	1	DD

1	1	1	0	0	1	0	1	E5

Description: The contents of the Index Register IX are pushed to the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current top of the Stack. This instruction first decrements SP and loads the high order byte of IX to the memory address specified by SP; then decrements SP again and loads the low order byte to the memory location corresponding to this new address in SP.

M Cycles	T States	4 MHz E.T.
4	15 (4, 5, 3, 3)	3.75

Condition Bits Affected: None

Example: If the Index Register IX contains 2233H and the Stack Pointer contains 1007H, at instruction PUSH IX memory address 1006H contains 22H, memory address 1005H contains 33H, and the Stack Pointer contains 1005H.

PUSH IY

Operation: (SP-2) ← IYL, (SP-1) ← IYH

Op Code: PUSH

Operands: IY

1	1	1	1	1	1	0	1	FD

1	1	1	0	0	1	0	1	E5

Description: The contents of the Index Register IY are pushed to the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current top of the Stack. This instruction first decrements the SP and loads the high order byte of IY to the memory address specified by SP; then decrements SP again and loads the low order byte to the memory location corresponding to this new address in SP.

M Cycles	T States	4 MHz E.T.
4	15 (4, 5, 3, 3)	3.75

Condition Bits Affected: None

Example: If the Index Register IY contains 2233H and the Stack Pointer Contains 1007H, at instruction PUSH IY memory address 1006H contains 22H, memory address 1005H contains 33H, and the Stack Pointer contains 1005H.

POP qq

Operation: qqH ← (SP+1), qqL ← (SP)

Op Code: POP

Operands: qq

| 1 | 1 | q | q | 0 | 0 | 0 | 1 |

Description: The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped to register pair qq. The Stack Pointer (SP) register pair holds the 16-bit address of the current top of the Stack. This instruction first loads to the low order portion of qq, the byte at memory location corresponding to the contents of SP; then SP is incriminated and the contents of the corresponding adjacent memory location are loaded to the high order portion of qq and the SP is now incriminated again. The operand qq identifies register pair BC, DE, HL, or AF, assembled as follows in the object code:

Pair	r
BC	00
DE	01
HL	10
AF	11

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

Condition Bits Affected: None

Example: If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction POP HL results in register pair HL containing 3355H, and the Stack Pointer containing 1002H.

POP IX

Operation: IXH ← (SP+1), IXL ← (SP)

Op Code: POP

Operands: IX

| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | DD |

| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | E1 |

Description: The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped to Index Register IX. The Stack Pointer (SP) register pair holds the 16-bit address of the current top of the Stack. This instruction first loads to the low order portion of IX the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded to the high order portion of IX. The SP is incremented again.

M Cycles	T States	4 MHz E.T.
4	14 (4, 4, 3, 3)	3.50

Condition Bits Affected: None

Example: If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction POP IX results in Index Register IX containing 3355H, and the Stack Pointer containing 1002H.

POP IY

Operation: IYH ← (SP-X1), IYL ← (SP)

Op Code: POP

Operands: IY

1	1	0	1	1	1	0	1	DD

1	1	1	1	1	1	0	1	FD

Description: The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped to Index Register IY. The Stack Pointer (SP) register pair holds the 16-bit address of the current top of the Stack. This instruction first loads to the low order portion of IY the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded to the high order portion of IY. The SP is incremented again.

M Cycles	T States	4 MHz E.T.
4	14 (4, 4, 3, 3)	3.50

Condition Bits Affected: None

Example: If the Stack Pointer Contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction POP IY results in Index Register IY containing 3355H, and the Stack Pointer containing 1002H.

Exchange, Block Transfer, and Search Group

EX DE, HL

Operation: DE $\leftrightarrow$ HL

Op Code: EX

Operands: DE, HL

1	1	1	0	1	0	1	1	EB

Description: The 2-byte contents of register pairs DE and HL are exchanged.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

Example: If the content of register pair DE is the number 2822H, and the content of the register pair HL is number 499AH, at instruction EX DE, HL the content of register pair DE is 499AH, and the content of register pair HL is 2822H.

EX AF, AF'

Operation: AF ↔ AF'

Op Code: EX

Operands: AF, AF'

0	0	0	0	1	0	0	0	08

Description: The 2-byte contents of the register pairs AF and AF are exchanged.

Register pair AF consists of registers A' and F'.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

Example: If the content of register pair AF is number 9900H, and the content of register pair AF is number 5944H, at instruction EX AF, AF' the contents of AF is 5944H, and the contents of AF' is 9900H.

EXX

Operation: $(BC) \leftrightarrow (BC'), (DE) \leftrightarrow (DE'), (HL) \leftrightarrow (HL')$

Op Code: EXX

Operands: —

1	1	0	1	1	0	0	0	D9

Description: Each 2-byte value in register pairs BC, DE, and HL is exchanged with the 2-byte value in BC', DE', and HL', respectively.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

Example: If the contents of register pairs BC, DE, and HL are the numbers 445AH, 3DA2H, and 8859H, respectively, and the contents of register pairs BC', DE', and HL' are 0988H, 9300H, and 00E7H, respectively, at instruction EXX the contents of the register pairs are as follows: BC' contains 0988H; DE' contains 9300H; HL contains 00E7H; BC' contains 445AH; DE' contains 3DA2H; and HL' contains 8859H.

EX (SP), HL

Operation: H $\leftrightarrow$ (SP+1), L $\leftrightarrow$ (SP)

Op Code: EX

Operands: (SP), HL

1	1	1	0	0	0	1	1	E3

Description: The low order byte contained in register pair HL is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of HL is exchanged with the next highest memory address (SP+1).

M Cycles	T States	4 MHz E.T.
5	19 (4, 3, 4, 3, 5)	4.75

Condition Bits Affected: None

Example: If the HL register pair contains 7012H, the SP register pair contains 8856H, the memory location 8856H contains byte 11H, and memory location 8857H contains byte 22H, then the instruction EX (SP), HL results in the HL register pair containing number 2211H, memory location 8856H containing byte 12H, memory location 8857H containing byte 70H and Stack Pointer containing 8856H.

EX (SP), IX

Operation: IXH $\leftrightarrow$ (SP+1), IXL $\leftrightarrow$ (SP)

Op Code: EX

Operands: (SP), IX

1	1	0	1	1	1	0	1	DD

1	1	1	0	0	0	1	1	E3

Description: The low order byte in Index Register IX is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IX is exchanged with the next highest memory address (SP+1).

M cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 4, 3, 5)	5.75

Condition Bits Affected: None

Example: If the Index Register IX contains 3988H, the SP register pair Contains 0100H, memory location 0100H contains byte 90H, and memory location 0101H contains byte 48H, then the instruction EX (SP), IX results in the IX register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H, and the Stack Pointer containing 0100H.

EX (SP), IY

Operation: IYH $\leftrightarrow$ (SP+1), IYL $\leftrightarrow$ (SP)

Op Code: EX

Operands: (SP), IY

1	1	1	1	1	1	0	1	FD

1	1	1	0	0	0	1	1	E3

Description: The low order byte in Index Register IY is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IY is exchanged with the next highest memory address (SP+1).

M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 4, 3, 5)	5.75

Condition Bits Affected: None

Example: If the Index Register IY contains 3988H, the SP register pair contains 0100H, memory location 0100H contains byte 90H, and memory location 0101H contains byte 48H, then the instruction EX (SP), IY results in the IY register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H, and the Stack Pointer containing 0100H.

LDI

Operation: (DE) ← (HL), DE ← DE + 1, HL ← HL + 1, BC ← BC -1

Op Code: LDI

Operands: (SP), HL

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	0	0	A0

Description: A byte of data is transferred from the memory location addressed, by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is set if BC -1 ≠ 0; reset otherwise
N is reset
C is not affected

Example: If the HL register pair contains 1111H, memory location 1111H contains byte 88H, the DE register pair contains 2222H, the memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction LDI results in the following contents in register pairs and memory addresses:

HL	contains 1112H
(1111H)	contains 88H
DE	contains 2223H
(2222H)	contains 88H
BC	contains 6H

LDIR

Operation: $(DE) \leftarrow (HL), DE \leftarrow DE + 1, HL \leftarrow HL + 1, BC F\leftrightarrow BC -1$

Op Code: LDIR

Operands: B8

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | ED |

| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | B0 |

Description: This 2-byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the DE register pair. Both these register pairs are incremented and the BC (Byte Counter) register pair is decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero, the program counter is decremented by two and the instruction is repeated. Interrupts are recognized and two refresh cycles are executed after each data transfer. When BC is set to zero prior to instruction execution, the instruction loops through 64 Kbytes.

For BC $\neq$ 0:

M Cycles	**T States**	**4 MHz E.T.**
5	21 (4, 4, 3, 5, 5)	5.25

For BC = 0:

M Cycles	**T States**	**4 MHz E.T.**
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is reset
N is reset
C is not affected

Example: If the HL register pair contains 11111H, the DE register pair contains
2222H, the BC register pair contains 0003H, and memory locations have
these contents:

(1111H)	contains 88H		(2222H)	contains 66H
(1112H)	contains 36H		(2223H)	contains 59H
(1113H)	contains A5H		(2224H)	contains C5H

then at execution of LDIR the contents of register pairs and memory
locations are:

HL	contains 1114H		
DE	contains 2225H		
BC	contains 0000H		
(1111H)	contains 88H	(2222H)	contains 88H
(1112H)	contains 36H	(2223H)	contains 36H
(1113H)	contains A5H	(2224H)	contains A5H

LDD

Operation: (DE) ← (HL), DE ← DE -1, HL ← HL-1, BC ← BC-1

Op Code: LDD

Operands: —

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	0	0	A8

Description: This 2-byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these register pairs including the BC (Byte Counter) register pair are decremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is set if BC -1 $\neq$ 0; reset otherwise
N is reset
C is not affected

Example: If the HL register pair contains 1111H, memory location 1111H contains byte 88H, the DE register pair contains 2222H, memory location 2222H contains byte 66H, and the BC register pair contains 7H, then instruction LDD results in the following contents in register pairs and memory addresses:

HL	contains 1110H
(1111H)	contains 88H
DE	contains 2221H
(2222H)	contains 88H
BC	contains 6H

LDDR

Operation: $(DE) \leftarrow (HL), DE \leftarrow D \leftarrow 1, HL \leftarrow HL-1, BC \leftarrow BC-1$

Op Code: LDDR

Operands: —

1	1	1	0	1	1	0	1	ED

1	0	1	1	1	0	0	0	B8

Description: This 2-byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these registers, as well as the BC (Byte Counter), are decremented. If decrementing causes BC to go to zero, the instruction is terminated. If BC is not zero, the program counter is decremented by two and the instruction is repeated. Interrupts are recognized and two refresh cycles execute after each data transfer.

When BC is set to zero, prior to instruction execution, the instruction loops through 64 Kbytes.

For BC $\neq 0$:

M Cycles	T States	4 MHz E.T.
5	21 (4, 4, 3, 5, 5)	5.25

For BC = 0:

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is reset
N is reset

Example: If the HL register pair contains `1114H`, the DE register pair contains `2225H`, the BC register pair contains `0003H`, and memory locations have these contents:

(1114H)	contains	A5H	(2225H)	contains	C5H
(1113H)	contains	36H	(2224H)	contains	59H
(1112H)	contains	88H	(2223H)	contains	66H

Then at execution of `LDDR` the contents of register pairs and memory locations are:

HL	contains	1111H			
DE	contains	2222H			
DC	contains	0000H			
(1114H)	contains	A5H	(2225H)	contains	A5H
(1113H)	contains	36H	(2224H)	contains	36H
(1112H)	contains	88H	(2223H)	contains	88H

CPI

Operation: A- (HL), HL ← HL +1, BC ← BC -1

Op Code: CPI

Operands: —

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	0	0	A1

Description: The contents of the memory location addressed by the HL register is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. Then HL is incremented and the Byte Counter (register pair BC) is decremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if A is (HL); reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if BC -1 is not 0; reset otherwise
N is set
C is not affected

Example: If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H. At execution of CPI the Byte Counter contains 0000H, the HL register pair contains 1112H, the Z flag in the F register sets, and the P/V flag in the F register resets. There is no effect on the contents of the Accumulator or address 1111H.

CPIR

Operation: A-(HL), HL ← HL+1, BC ← BC-1

Op Code: CPIR

Operands: —

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | ED |

| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | B1 |

Description: The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. HL is incremented and the Byte Counter (register pair BC) is decremented. If decrementing causes BC to go to zero or if A = (HL), the instruction is terminated. If BC is not zero and A ≠ (HL), the program counter is decremented by two and the instruction is repeated. Interrupts are recognized and two refresh cycles are executed after each data transfer.

If BC is set to zero before instruction execution, the instruction loops through 64 Kbytes if no match is found.

For BC ≠ 0 and A ≠ (HL):

M cycles	T States	4 MHz E.T.
5	21 (4, 4, 3, 5, 5)	5.25

For BC = 0 and A = (HL):

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if A equals (HL); reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if BC -1 does not equal 0; reset otherwise
N is set
C is not affected

Example: If the HL register pair contains 1111H, the Accumulator contains F3H, the Byte Counter contains 0007H, and memory locations have these contents:

> (1111H) contains 52H
> (1112H) contains 00H
> (1113H) contains F3H

Then, at execution of CPIR the contents of register pair HL is 1114H, the contents of the Byte Counter is 0004H, the P/V flag in the F register sets, and the Z flag in the F register sets.

CPD

Operation: A -(HL), HL ← HL -1, BC ← BC -1

Op Code: CPD

Operands: —

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	0	1	A9

Description: The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and Byte Counter (register pair BC) are decremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if A equals (HL); reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if BC -1 x 0; reset otherwise
N is set
C is not affected

Example: If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H. At execution of CPD the Byte Counter contains 0000H, the HL register pair contains 1110H, the flag in the F register sets, and the P/V flag in the F register resets. There is no effect on the contents of the Accumulator or address 1111H.

CPDR

Operation: A -(HL), HL ← HL -1, BC ← BC -1

Op Code: CPDR

Operands: —

1	1	1	0	1	1	0	1	ED

1	0	1	1	1	0	0	1	B9

Description: The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and BC (Byte Counter) register pairs are decremented. If decrementing causes the BC to go to zero or if A = (HL), the instruction is terminated. If BC is not zero and A = (HL), the program counter is decremented by two and the instruction is repeated. Interrupts are recognized and two refresh cycles execute after each data transfer. When BC is set to zero, prior to instruction execution, the instruction loops through 64 Kbytes if no match is found.

For BC $\neq$ 0 and A $\neq$ (HL):

M Cycles	T States	4 MHz E.T.
5	21 (4, 4, 3, 5, 5)	5.25

For BC = 0 and A = (HL):

M Cycles	T States	4 MHz E.T.
4	16 (4, 4, 3, 5)	4.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if A = (HL); reset otherwise
H is set if borrow form bit 4; reset otherwise
P/V is set if BC -1 $\neq$ 0; reset otherwise
N is set
C is not affected

Example: If the HL register pair contains `1118H`, the Accumulator contains `F3H`, the Byte Counter contains `0007H`, and memory locations have these contents.

> (`1118H`) contains `52H`
> (`1117H`) contains `00H`
> (`1116H`) contains `F3H`

Then, at execution of `CPDR` the contents of register pair HL are `1115H`, the contents of the Byte Counter are `0004H`, the P/V flag in the F register sets, and the Z flag in the F register sets.

8-Bit Arithmetic Group

ADD A, r

Operation: $A \leftarrow A + r$

Op Code: ADD

Operands: A, r

1	0	0	0	0	←	r	→
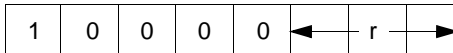
Description: The contents of register r are added to the contents of the Accumulator, and the result is stored in the Accumulator. The symbol r identifies the registers A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is set if result is negative; reset otherwise

Z is set if result is zero; reset otherwise

H is set if carry from bit 3; reset otherwise

P/V is set if overflow; reset otherwise

N is reset

C is set if carry from bit 7; reset otherwise

Example: If the contents of the Accumulator are 44H, and the contents of register C
               are 11H, at execution of ADD A,C the contents of the Accumulator are
               55H.

ADD A, n

Operation: $A \leftarrow A + n$

Op Code: ADD

Operands: A, n

1	1	0	0	0	1	1	0	C6
←			n				→	
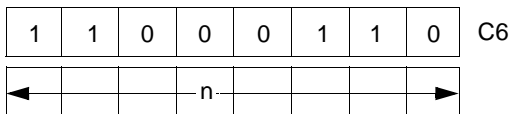
Description: The integer n is added to the contents of the Accumulator, and the results
are stored in the Accumulator.

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 7; reset otherwise

Example: If the contents of the Accumulator are 23H, at execution of ADD A, 33H
the contents of the Accumulator are 56H.

ADD A, (HL)

Operation: $A \leftarrow A + (HL)$

Op Code: ADD

Operands: A, (HL)

1	0	0	0	0	1	1	0	86

Description: The byte at the memory address specified by the contents of the HL register pair is added to the contents of the Accumulator, and the result is stored in the Accumulator.

M Cycles	**T States**	**4 MHz E.T.**
2	7 (4, 3)	1.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 7; reset otherwise

Example: If the contents of the Accumulator are A0H, and the content of the register pair HL is 2323H, and memory location 2323H contains byte 08H, at execution of ADD A, (HL) the Accumulator contains A8H.

ADD A, (IX + d)

Operation: A ← A + (IX+d)

Op Code: ADD

Operands: A, (IX + d)

1	1	0	1	1	1	0	1	DD

1	0	0	0	0	1	1	0	86

◄———————— d ————————►
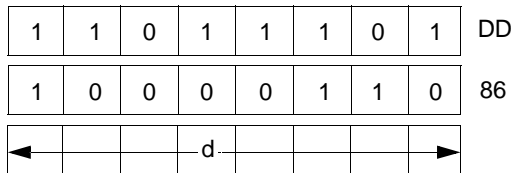
Description: The contents of the Index Register (register pair IX) is added to a two's complement displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M Cycles	T States	4 MHz E.T.
5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 7; reset otherwise

Example: If the Accumulator contents are 11H, the Index Register IX contains 1000H, and if the contents of memory location 1005H is 22H, at execution of ADD A, (IX + 5H) the contents of the Accumulator are 33H.

ADD A, (IY + d)

Operation: $A \leftarrow A + (ID+d)$

Op Code: ADD

Operands: A, (IY + d)

1	1	1	1	1	1	0	1	FD

1	0	0	0	0	1	1	0	86

◄				d			►	
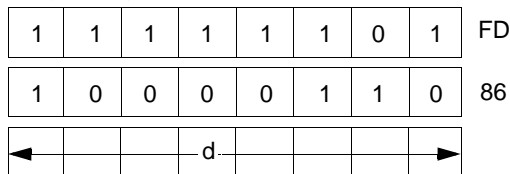
Description: The contents of the Index Register (register pair IY) is added to a two's complement displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator, and the result is stored in the Accumulator.

M Cycles	T States	4 MHz E.T.
5	19(4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3: reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 7; reset otherwise

Example: If the Accumulator contents are 11H, the Index Register Pair IY contains 1000H, and if the content of memory location 1005H is 22H, at execution of ADD A, (IY + 5H) the contents of the Accumulator are 33H.

ADC A, s

Operation: $A \leftarrow A + s + CY$

Op Code: ADC

Operands: A, s

This s operand is any of r, n, (HL), (IX+d), or (lY+d) as defined for the analogous ADD instruction. These possible Op Code/operand combinations are assembled as follows in the object code:

ADC A,r	1	0	0	0	1	◄──── r* ────►		

ADC A,n	1	1	0	0	1	1	1	0	CE
	◄──────── n ────────►							

ADC A, (HL)	1	0	0	0	1	1	1	0	8E

ADC A, (IX+d)	1	1	0	1	1	1	1	0	DD
	1	0	0	0	1	1	1	0	8E
	◄──────── d ────────►							

ADC A, (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	0	0	1	1	1	0	8E
	◄──────── d ────────►							

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The s operand, along with the Carry Flag (C in the F register) is added to the contents of the Accumulator, and the result is stored in the Accumulator.

Instruction	M Cycle	T States	4 MHz E.T.
ADC A, r	1	4	1.00
ADC A, n	2	7 (4, 3)	1.75
ADC A, (HL)	2	7 (4, 3)	1.75
ADC A, (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
ADC A, (lY+d)	5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 7: reset otherwise

Example: If the Accumulator contents are 16H, the Carry Flag is set, the HL register pair contains 6666H, and address 6666H contains 10H, at execution of ADC A, (HL) the Accumulator contains 27H.

SUB s

Operation: A ← A - s

Op Code: SUB

Operands: s

This s operand is any of r, n, (HL), (IX+d), or (lY+d) as defined for the analogous ADD instruction. These possible Op Code/operand combinations are assembled as follows in the object code:

SUB r	1	0	0	1	0	◄─── r* ───►		

SUB n	1	1	0	1	0	1	1	0	D6
	◄────────── n ──────────►								

SUB (HL)	1	0	0	1	0	1	1	0	96

SUB (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	0	1	0	1	1	0	96
	◄────────── d ──────────►								

SUB (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	0	1	0	1	1	0	96
	◄────────── d ──────────►								

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The s operand is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

Instruction	M Cycle	T States	4 MHz E.T.
SUB r	1	4	1.00
SUB n	2	7 (4, 3)	1.75
SUB (HL)	2	7 (4, 3)	1.75
SUB (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
SUB (lY+d)	5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if overflow; reset otherwise
N is set
C is set if borrow; reset otherwise

Example: If the Accumulator contents are 29H, and register D contains 11H, at execution of SUB D the Accumulator contains 18H.

SBC A, s

Operation: $A \leftarrow A - s - CY$

Op Code: SBC

Operands: A, s

The s operand is any of r, n, (HL), (IX+d), or (lY+d) as defined for the analogous ADD instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

SBC A, r	1	0	0	1	1	◄— r* —►			

SBC A, n	1	1	0	1	1	1	1	0	DE
	◄—			n			—►		

SBC A, (HL)	1	0	0	1	1	1	1	0	9E

SBC A, (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	0	1	1	1	1	0	9E
	◄—			d			—►		

SBC A, (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	0	1	1	1	1	0	9E
	◄—			d			—►		

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The s operand, along with the Carry flag (C in the F register) is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T.
SBC A, r	1	4	1.00
SBC A, n	2	7(4, 3)	1.75
SBC A, (HL)	2	7 (4, 3)	1.75
SBC A, (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
SBC A, (lY+d)	5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is reset if overflow; reset otherwise
N is set
C is set if borrow; reset otherwise

Example: If the Accumulator contains 16H, the carry flag is set, the HL register pair contains 3433H, and address 3433H contains 05H, at execution of SBC A, (HL) the Accumulator contains 10H.

AND s

Operation: $A \leftarrow A \wedge s$

Op Code: AND

Operands: s

The s operand is any of r, n, (HL), (IX+d), or (lY+d), as defined for the analogous ADD instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

AND r*	1	0	1	0	0	← r* →		

AND n	1	1	1	0	0	1	1	0	E6
	← n →								

AND (HL)	1	0	1	0	0	1	1	0	A6

AND (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	1	0	0	1	1	0	A6
	← d →								

AND (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	1	0	0	1	1	0	A6
	← d →								
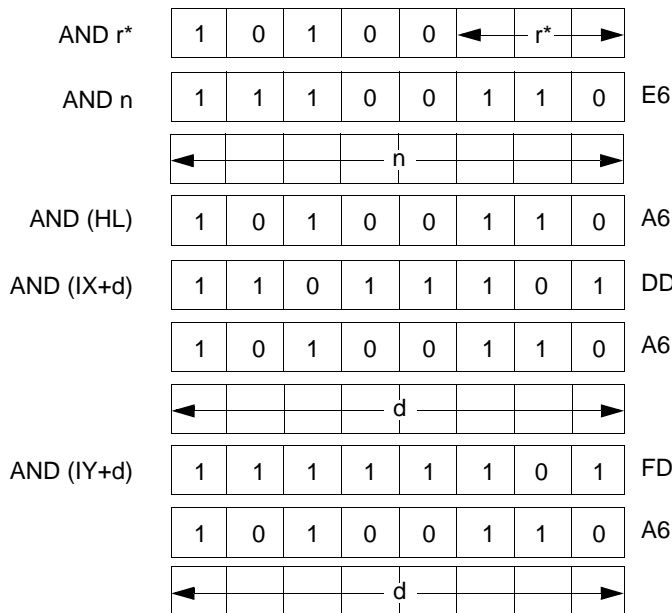
*r identifies registers B, C, D, E, H, L, or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: A logical AND operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T.
AND r	1	4	1.00
AND n	2	7 (4, 3)	1.75
AND (HL)	2	7 (4, 3)	1.75
AND (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
AND (IX+d)	5	19 (4, 4, 3. 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set
P/V is reset if overflow; reset otherwise
N is reset
C is reset

Example: If the B register contains 7BH (0111 1011), and the Accumulator contains C3H (1100 0011), at execution of AND B the Accumulator contains 43H (0100 0011).

OR s

Operation: $A \leftarrow A \vee s$

Op Code: OR

Operands: s

The s operand is any of r, n, (HL), (IX+d), or (lY+d), as defined for the analogous ADD instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

OR r*	1	0	1	1	0	←	r*	→	

OR n	1	1	1	1	0	1	1	0	F6

←				n				→	

OR (HL)	1	0	1	1	0	1	1	0	B6

OR (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	1	1	0	1	1	0	B6

←				d				→	

OR (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	1	1	0	1	1	0	B6

←				d				→	
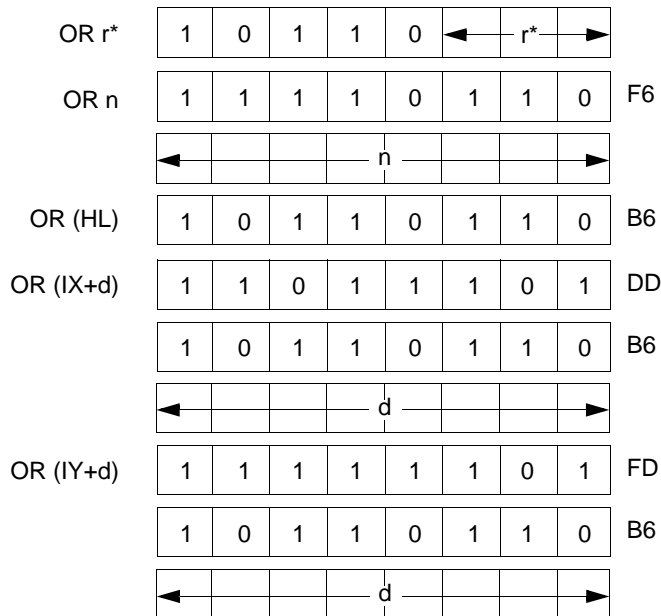
*r identifies registers B, C-, D, E, H, L, or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: A logical OR operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Instruction	M cycles	T States	4 MHz E.T.
OR r	1	4	1.00
OR n	2	7 (4, 3)	1.75
OR (HL)	2	7 (4, 3)	1.75
OR (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
OR (1Y+d)	5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if overflow; reset otherwise
N is reset
C is reset

Example: If the H register contains 48H (0100 0100), and the Accumulator contains 12H (0001 0010), at execution of OR H the Accumulator contains 5AH (0101 1010).
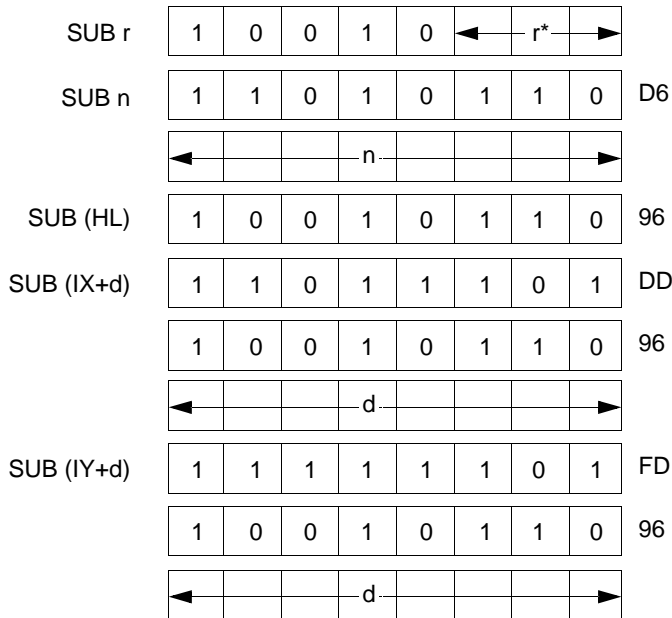
XOR s

Operation: $A \leftarrow A \oplus s$

Op Code: XOR

Operands: s

The s operand is any of r, n, (HL), (IX+d), or (lY+d), as defined for the analogous ADD instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

OR r*	1	0	1	1	0	◄— r* —►		

OR n	1	1	1	1	0	1	1	0	F6

◄———————— n ————————►								

OR (HL)	1	0	1	1	0	1	1	0	B6

OR (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	1	1	0	1	1	0	B6

◄———————— d ————————►								

OR (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	1	1	0	1	1	0	B6

◄———————— d ————————►								
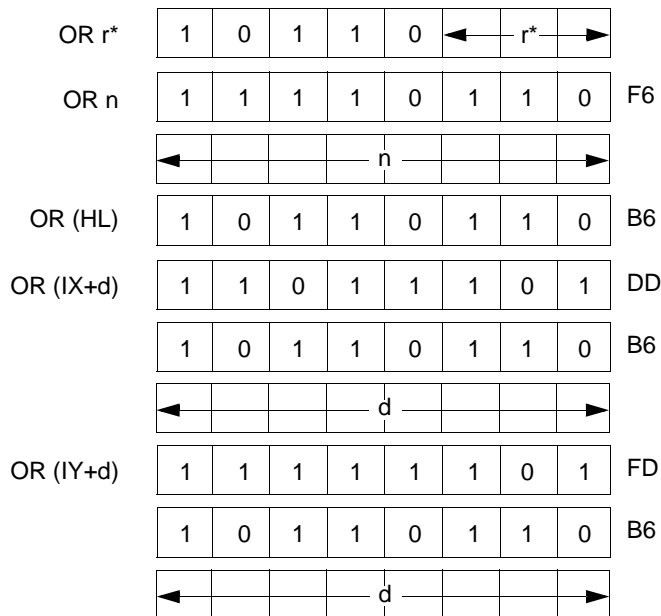
*r identifies registers B, C, D, E, H, L, or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The logical exclusive-OR operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T.
XOR r	1	4	1.00
XOR n	2	7 (4, 3)	1.75
XOR (HL)	2	7 (4, 3)	1.75
XOR (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
XOR (lY+d)	5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise
N is reset
C is reset

Example: If the Accumulator contains 96H (1001 0110), at execution of XOR 5DH (5DH = 0101 1101) the Accumulator contains CBH (1100 1011).

CP s

Operation: A - s

Op Code: CP

Operands: s

The s operand is any of r, n, (HL), (IX+d), or (lY+d), as defined for the analogous ADD instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

CP r*	1	0	1	1	1	◄—— r* ——►		

CP n	1	1	1	1	1	1	1	0	FE
	◄————————— n —————————►								

CP (HL)	1	0	1	1	1	1	1	0	BE

CP (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	1	1	1	1	1	0	BE
	◄————————— d —————————►								

CP (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	1	1	1	1	1	0	BE
	◄————————— d —————————►								
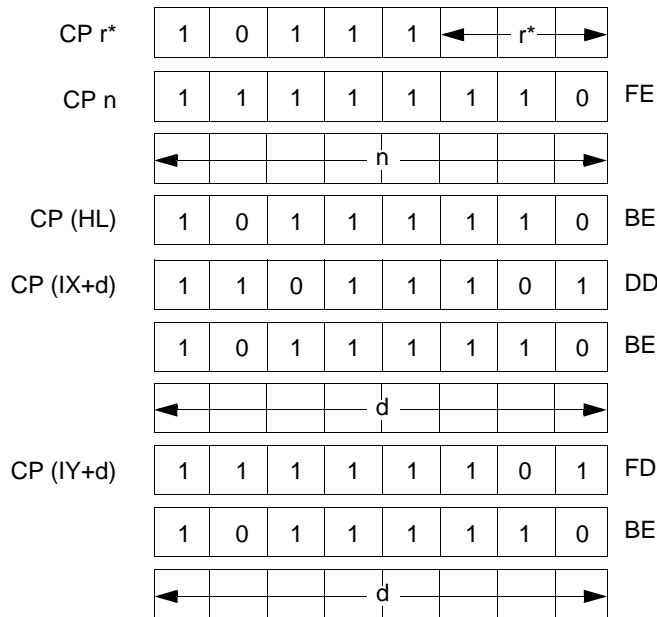
*r identifies registers B, C, D, E, H, L, or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The contents of the s operand are compared with the contents of the Accumulator. If there is a true compare, the Z flag is set. The execution of this instruction does not affect the contents of the Accumulator.

Instruction	M Cycles	T States	4 MHz E.T.
CP r	1	4	1.00
CP n	2	7(4, 3)	1.75
CP (HL)	2	7 (4, 3)	1.75
CP (IX+d)	5	19 (4, 4, 3, 5, 3)	4.75
CP (lY+d)	5	19 (4, 4, 3, 5, 3)	4.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if overflow; reset otherwise
N is set
C is set if borrow; reset otherwise

Example: If the Accumulator contains 63H, the HL register pair contains 6000H, and memory location 6000H contains 60H, the instruction CP (HL) results in the PN flag in the F register resetting.

INC r

Operation: r ← r + 1

Op Code: INC

Operands: r

| 0 | 0 | ◄——— r ———► | 1 | 0 | 0 |
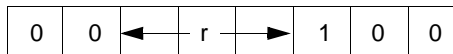
Description: Register r is incremented and register r identifies any of the registers A, B, C, D, E, H, or L, assembled as follows in the object code.

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if r was 7FH before operation; reset otherwise
N is reset
C is not affected

Example: If the contents of register D are 28H, at execution of INC D the contents of register D are 29H.

INC (HL)

Operation: $(HL) \leftarrow (HL) + 1$

Op Code: INC

Operands: (HL)

0	0	1	1	0	1	0	0	34

Description: The byte contained in the address specified by the contents of the HL register pair is incremented.

	M Cycles	**T States**	**4 MHz E.T.**
	3	11 (4, 4, 3)	2.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if (HL) was 7FH before operation; reset otherwise
N is reset
C is not affected

Example: If the contents of the HL register pair are 3434H, and the contents of address 3434H are 82H, at execution of INC (HL) memory location 3434H contains 83H.

INC (IX+d)

Operation: $(IX+d) \leftarrow (IX+d) + 1$

Op Code: INC

Operands: (IX+d)

1	1	0	1	1	1	0	1	DD

0	0	1	1	0	1	0	0	34

←				d			→
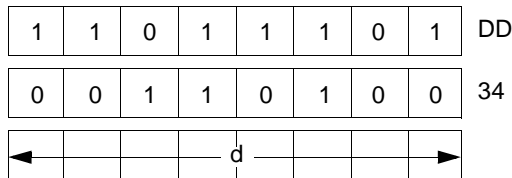
Description: The contents of the Index Register IX (register pair IX) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if (IX+d) was 7FH before operation; reset otherwise
N is reset
C is not affected

Example: If the contents of the Index Register pair IX are 2020H, and the memory location 2030H contains byte 34H, at execution of INC (IX+10H) the contents of memory location 2030H is 35H.

INC (lY+d)

Operation: $(lY+d) \leftarrow (lY+d) + 1$

Op Code: INC

Operands: (lY+d)

1	1	1	1	1	1	0	1	FD

0	0	1	1	0	1	0	0	34

◄				d			►
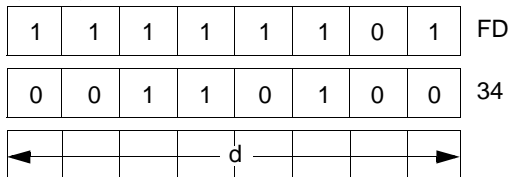
Description: The contents of the Index Register IY (register pair IY) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if carry from bit 3; reset otherwise
P/V is set if (lY+d) was 7FH before operation; reset otherwise
N is reset
C is not affected

Example: If the contents of the Index Register pair IY are 2020H, and the memory location 2030H contain byte 34H, at execution of INC (IY+10H) the contents of memory location 2030H are 35H.

DEC m

Operation: m ← m- 1

Op Code: DEC

Operands: m

The m operand is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous INC instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

DEC r*	0	0	◄── r ──►	1	0	1	

DEC (HL)	0	0	1	1	0	1	0	1	35

DEC (IX+d)	1	1	0	1	1	1	0	1	DD
	0	0	1	1	0	1	0	1	35
	◄──────── d ────────►								

DEC (IY+d)	1	1	1	1	1	1	0	1	FD
	0	0	1	1	0	1	0	1	35
	◄──────── d ────────►								

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The byte specified by the m operand is decremented.

Instruction	M Cycles	T States	4 MHz E.T.
DEC r	1	4	1.00
DEC (HL)	3	11 (4, 4, 3)	2.75
DEC (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
DEC (lY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if borrow from bit 4, reset otherwise
P/V is set if m was 80H before operation; reset otherwise
N is set
C is not affected

Example: If the D register contains byte 2AH, at execution of DEC D register D
contains 29H.

General-Purpose Arithmetic and CPU Control Groups

DAA

Operation:

Op Code: DAA

0	0	1	0	0	1	1	1	27

**Description:** This instruction conditionally adjusts the Accumulator for BCD addition and subtraction operations. For addition (ADD, ADC, INC) or subtraction (SUB, SBC, DEC, NEG), the following table indicates the operation performed:

Operation	C Before DAA	Hex Value In Upper Digit (bit 7-4)	H Before DAA	Hex Value In Lower Digit (bit 3-0)	Number Added To Byte	C After DAA
	0	9-0	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADD	0	A-F	0	0-9	60	1
ADC	0	9-F	0	A-F	66	1
INC	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB	0	0-9	0	0-9	00	0
SBC	0	0-8	1	6-F	FA	0
DEC	1	7-F	0	0-9	A0	1
NEG	1	6-7	1	6-F	9A	1

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is set if most-significant bit of Accumulator is 1 after operation; reset otherwise

Z is set if Accumulator is zero after operation; reset otherwise

H, see instruction

P/V is set if Accumulator is even parity after operation; reset otherwise

N is not affected

C, see instruction

Example: If an addition operation is performed between 15 (BCD) and 27 (BCD), simple decimal arithmetic gives this result:

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

But when the binary representations are added in the Accumulator according to standard binary arithmetic.

$$
\begin{array}{ll}
0001 & \underline{0101} \\
+\,0010 & \overline{0111} \\
\hline
0011 & 1100 \qquad = 3C
\end{array}
$$

the sum is ambiguous. The DAA instruction adjusts this result so that the correct BCD representation is obtained:

$$
\begin{array}{ll}
0011 & 1100 \\
+\,0000 & \underline{0110} \\
\hline
0100 & 0010 \qquad = 42
\end{array}
$$

CPL

Operation: $A \leftarrow \overline{A}$

Op Code: CPL

0	0	1	0	1	1	1	1	2F

Description: The contents of the Accumulator (register A) are inverted (one's complement).

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is not affected
Z is not affected
H is set
P/V is not affected
N is set
C is not affected

Example: If the contents of the Accumulator are 1011 0100, at execution of CPL the Accumulator contents are 0100 1011.

NEG

Operation: A ← 0 - A

Op Code: NEG

1	1	1	0	1	1	0	1	ED

0	1	0	0	0	1	0	0	44

Description: The contents of the Accumulator are negated (two's complement). This is the same as subtracting the contents of the Accumulator from zero. Note that 80H is left unchanged.

M Cycles	**T States**	**4 MHz E.T.**
2	8 (4, 4)	2.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is 0; reset otherwise
H is set if borrow from bit 4; reset otherwise
P/V is set if Accumulator was 80H before operation; reset otherwise
N is set
C is set if Accumulator was not 00H before operation; reset otherwise

Example: If the contents of the Accumulator are

1	0	0	1	1	0	0	0

at execution of NEG the Accumulator contents are

0	1	1	0	1	0	0	0

CCF

Operation: $CY \leftarrow \overline{CY}$

Op Code: CCF

0	0	1	1	1	1	1	1	3F

Description: The Carry flag in the F register is inverted.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is not affected
Z is not affected
H, previous carry is copied
P/V is not affected
N is reset
C is set if CY was 0 before operation; reset otherwise

SCF

Operation: $CY \leftarrow 1$

Op Code: SCF

0	0	1	1	0	1	1	1

37

Description: The Carry flag in the F register is set.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is set

NOP

Operation: —

Op Code: NOP

0	0	0	0	0	0	0	0	00

Description: The CPU performs no operation during this machine cycle.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

HALT

Operation: —

Op Code: HALT

0	1	1	1	0	1	1	0	76

Description: The HALT instruction suspends CPU operation until a subsequent interrupt or reset is received. While in the HALT state, the processor executes NOPs to maintain memory refresh logic.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

DI

Operation: IFF $\leftarrow 0$

Op Code: DI

1	1	1	1	0	0	1	1	F3

Description: DI disables the maskable interrupt by resetting the interrupt enable flip-flops (IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

Example: When the CPU executes the instruction DI the maskable interrupt is disabled until it is subsequently re-enabled by an EI instruction. The CPU does not respond to an Interrupt Request (INT) signal.

EI

Operation: IFF ← 1

Op Code: EI

1	1	1	1	1	0	1	1	FB

Description: The enable interrupt instruction sets both interrupt enable flip flops (IFFI and IFF2) to a logic 1, allowing recognition of any maskable interrupt. Note that during the execution of this instruction and the following instruction, maskable interrupts are disabled.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

Example: When the CPU executes instruction El RETI the maskable interrupt is enabled at execution of the RETI instruction.

IM 0

Operation: —

Op Code: IM

Operands: 0

1	1	1	0	1	1	0	1	ED

0	1	0	0	0	1	1	0	46

Description: The IM 0 instruction sets interrupt mode 0. In this mode, the interrupting device can insert any instruction on the data bus for execution by the CPU. The first byte of a multi-byte instruction is read during the interrupt acknowledge cycle. Subsequent bytes are read in by a normal memory read sequence.

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

Condition Bits Affected: None

IM 1

Operation: —

Op Code: IM

Operands: 1

1	1	1	0	1	1	0	1	ED

0	1	0	1	0	1	1	0	56

Description: The IM 1 instruction sets interrupt mode 1. In this mode, the processor responds to an interrupt by executing a restart to location 0038H.

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

Condition Bits Affected: None

IM 2

Operation: —

Op Code: IM

Operands: 2

1	1	1	0	1	1	0	1	ED

0	1	0	1	1	1	1	0	5E

Description: The IM 2 instruction sets the vectored interrupt mode 2. This mode allows an indirect call to any memory location by an 8-bit vector supplied from the peripheral device. This vector then becomes the least-significant eight bits of the indirect pointer, while the I register in the CPU provides the most-significant eight bits. This address points to an address in a vector table that is the starting address for the interrupt service routine.

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

Condition Bits Affected: None

16-Bit Arithmetic Group

ADD HL, ss

Operation: HL ← HL + ss

Op Code: ADD

Operands: HL, ss

0	0	s	s	1	0	0	1

Description: The contents of register pair ss (any of register pairs BC, DE, HL, or SP) are added to the contents of register pair HL and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M Cycles	T States	4 MHz E.T.
3	11 (4, 4, 3)	2.75

Condition Bits Affected:

S is not affected
Z is not affected
H is set if carry out of bit 11; reset otherwise
P/V is not affected
N is reset
C is set if carry from bit 15; reset otherwise

Example: If register pair HL contains the integer 4242H, and register pair DE contains 1111H, at execution of ADD HL, DE the HL register pair contains 5353H.

ADC HL, ss

Operation: $HL \leftarrow HL + ss + CY$

Op Code: ADC

Operands: HL, ss

1	1	1	0	1	1	0	1	ED

0	1	s	s	1	0	1	0

Description: The contents of register pair ss (any of register pairs BC, DE, HL, or SP) are added with the Carry flag (C flag in the F register) to the contents of register pair HL, and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M Cycles	T States	4 MHz E.T.
4	15 (4, 4, 4, 3)	3.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
R is set if carry out of bit 11;. reset otherwise
P/V is set if overflow; reset otherwise
N is reset
C is set if carry from bit 15; reset otherwise

Example: If the register pair BC contains 2222H, register pair HL contains 5437H, and the Carry Flag is set, at execution of ADC HL, BC the contents of HL are 765AH.

SBC HL, ss

Operation: HL ← HI - ss - CY

Op Code: SBC

Operands: HL, ss

1	1	1	0	1	1	0	1	ED

0	1	s	s	0	0	1	0

Description: The contents of the register pair ss (any of register pairs BC, DE, HL, or SP) and the Carry Flag (C flag in the F register) are subtracted from the contents of register pair HL, and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M Cycles	T States	4 MHz E.T.
4	15 (4, 4, 4, 3)	3.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is set if a borrow from bit 12; reset otherwise
P/V is set if overflow; reset otherwise
N is set
C is set if borrow; reset otherwise

Example: If the contents of the HL, register pair are 9999H, the contents of register pair DE are 1111H, and the Carry flag is set. At execution of SBC HL, DE the contents of HL are 8887H.

ADD IX, pp

Operation: IX ← IX + pp

Op Code: ADD

Operands: IX, pp

1	1	0	1	1	1	0	1	DD

0	0	p	p	1	0	0	1

Description: The contents of register pair pp (any of register pairs BC, DE, IX, or SP) are added to the contents of the Index Register IX, and the results are stored in IX. Operand pp is specified as follows in the assembled object code.

Register Pair	pp
BC	00
DE	01
IX	10
SP	11

M Cycles	T States	4 MHz E.T.
4	15 (4, 4, 4, 3)	3.75

Condition Bits Affected:

S is not affected
Z is not affected
H is set if carry out of bit 11; reset otherwise
P/V is not affected
N is reset
C is set if carry from bit 15; reset otherwise

Example: If the contents of Index Register IX are 333H, and the contents of register pair BC are 5555H, at execution of ADD IX, BC the contents of IX are 8888H.

ADD IY, rr

Operation: IY ← IY + rr

Op Code: ADD

Operands: IY, rr

1	1	1	1	1	1	0	1	FD

0	0	r	r	1	0	0	1

Description: The contents of register pair rr (any of register pairs BC, DE, IY, or SP) are added to the contents of Index Register IY, and the result is stored in IY. Operand rr is specified as follows in the assembled object code.

Register Pair	rr
BC	00
DE	01
IY	10
SP	11

M Cycles	T States	4 MHz E.T.
4	15 (4, 4, 4, 3)	3.75

Condition Bits Affected:

S is not affected
Z is not affected
H is set if carry out of bit 11; reset otherwise
P/V is not affected
N is reset
C is set if carry from bit 15; reset otherwise

Example: If the contents of Index Register IY are 333H, and the contents of register pair BC are 555H, at execution of ADD IY, BC the contents of IY are 8888H.

INC ss

Operation: ss ← ss + 1

Op Code: INC

Operands: ss

0	0	s	s	0	0	1	1

Description: The contents of register pair ss (any of register pairs BC, DE, HL, or SP) are incremented. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M Cycles	T States	4 MHz E.T.
1	6	1.50

Condition Bits Affected: None

Example: If the register pair contains 1000H, after the execution of INC HL, HL contains 1001H.

INC IX

Operation: $IX \leftarrow IX + 1$

Op Code: INC

Operands: IX

1	1	0	1	1	1	0	1	DD

0	0	1	0	0	0	1	1	23

Description: The contents of the Index Register IX are incremented.

M Cycles	T States	4 MHz E.T.
2	10 (4, 6)	2.50

Condition Bits Affected: None

Example: If the Index Register IX contains the integer 3300H. at execution of INC IX the contents of Index Register IX are 3301H.

INC IY

Operation: $IY \leftarrow IY + 1$

Op Code: INC

Operands: IY

1	1	1	1	1	1	0	1	FD

0	0	1	0	0	0	1	1	23

Description: The contents of the Index Register IY are incremented.

M Cycles	T States	4 MHz E.T.
2	10 (4, 6)	2.50

Condition Bits Affected: None

Example: If the contents of the Index Register are 2977H, at execution of INC IY the contents of Index Register IY are 2978H.

DEC ss

Operation: ss ← ss - 1

Op Code: DEC

Operands: ss

0	0	s	s	1	0	1	1

Description: The contents of register pair ss (any of the register pairs BC, DE, HL, or SP) are decremented. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M Cycles	T States	4 MHz E.T.
1	6	1.50

Condition Bits Affected: None

Example: If register pair HL contains 1001H, at execution of DEC HL the contents of HL are 1000H.

DEC IX

Operation: IX ← IX - 1

Op Code: DEC

Operands: IX

1	1	0	1	1	1	0	1	DD

0	0	1	0	1	0	1	1	2B

Description: The contents of Index Register IX are decremented.

M Cycles	T States	4 MHz E.T.
2	10 (4, 6)	2.50

Condition Bits Affected: None

Example: If the contents of Index Register IX are 2006H, at execution of DEC IX the contents of Index Register IX are 2005H.

DEC IY

Operation: IY ← IY - 1

Op Code: DEC

Operands: IY

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | FD |

| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 2B |

Description: The contents of the Index Register IY are decremented.

M Cycles	T States	4 MHz E.T.
2	10 (4, 6)	2.50

Condition Bits Affected: None

Example: If the contents of the index Register IY are 7649H, at execution of DEC IY the contents of index Register IY are 7648H.

Rotate and Shift Group

RLCA

Operation:

A
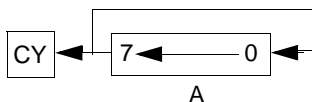
Op Code: RLCA

Operands: —
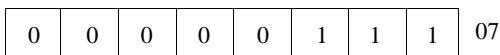
0	0	0	0	0	1	1	1

07

Description: The contents of the Accumulator (register A) are rotated left 1-bit position. The sign bit (bit 7) is copied to the Carry flag and also to bit 0. Bit 0 is the least-significant bit.

M cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 7 of Accumulator

Example: If the contents of the Accumulator are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0
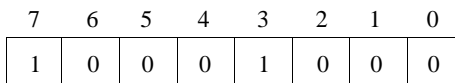
at execution of RLCA the contents of the Accumulator and Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1
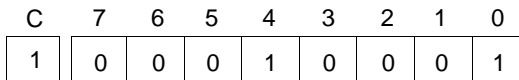
RLA

Operation:
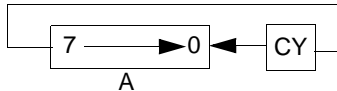
Op Code: RLA

Operands: —
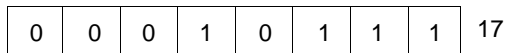
0	0	0	1	0	1	1	1	17

Description: The contents of the Accumulator (register A) are rotated left 1-bit position through the Carry flag. The previous content of the Carry flag is copied to bit 0. Bit 0 is the least-significant bit.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: Condition Bits Affected

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 7 of Accumulator

Example: If the contents of the Accumulator and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	1	0
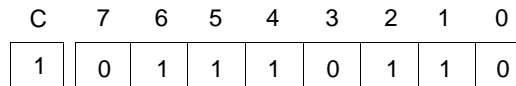
at execution of RLA the contents of the Accumulator and the Carry flag are

C	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0	1

RRCA

Operation:

A
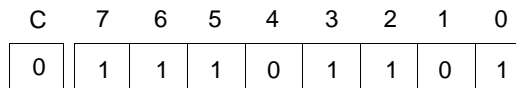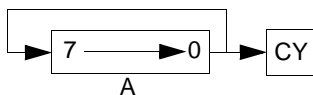
Op Code: RRCA

Operands: —

| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0F |
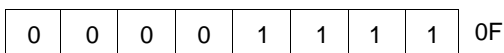
Description: The contents of the Accumulator (register A) are rotated right 1-bit position. Bit 0 is copied to the Carry flag and also to bit 7. Bit 0 is the least-significant bit.

M Cycles	**T States**	**4 MHz E.T.**
1	4	1.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 0 of Accumulator

Example: If the contents of the Accumulator are

7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1

at execution of RRCA the contents of the Accumulator and the Carry flag are

7	6	5	4	3	2	1	0	C
1	0	0	1	1	0	0	0	1
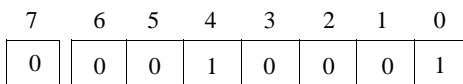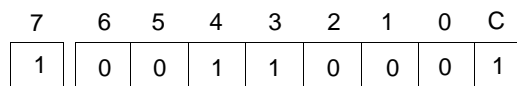
RRA

Operation:

Op Code: RRA

Operands: —
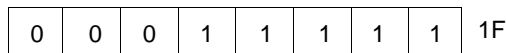
0	0	0	1	1	1	1	1	1F

Description: The contents of the Accumulator (register A) are rotated right 1-bit position through the Carry flag. The previous content of the Carry flag is copied to bit 7. Bit 0 is the least-significant bit.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected:

S is not affected
Z is not affected
H is reset
P/V is not affected
N is reset
C is data from bit 0 of Accumulator

Example: If the contents of the Accumulator and the Carry Flag are

7	6	5	4	3	2	1	0	C
1	1	1	0	0	0	0	1	0
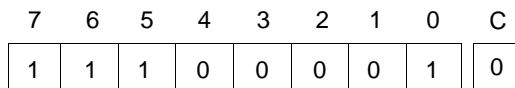
at execution of RRA the contents of the Accumulator and the Carry flag are

7	6	5	4	3	2	1	0	C
0	1	1	1	0	0	0	0	1

RLC r

Operation:

| CY | ← | 7 ← | 0 | ← |

r
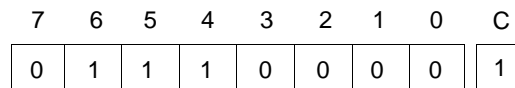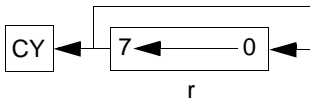
Op Code: RLC

Operands: r

| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | CB

| 0 | 0 | 0 | 0 | 0 | ← | r | → |
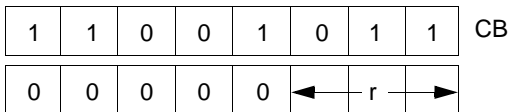
Description: The contents of register r are rotated left 1-bit position. The content of bit 7 is copied to the Carry flag and also to bit 0. Operand r is specified as follows in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise
N is reset
C is data from bit 7 of source register

Example: If the contents of register r are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

at execution of RLC r the contents of register r and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLC (HL)

Operation:

(HL)
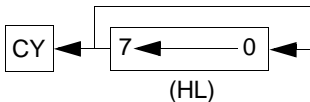
Op Code: RLC

Operands: (HL)

1	1	0	0	1	0	1	1	CB

0	0	0	0	0	1	1	0	06

Description: The contents of the memory address specified by the contents of register pair HL are rotated left 1-bit position. The content of bit 7 is copied to the Carry flag and also to bit 0. Bit 0 is the least-significant bit.

M Cycles	T States	4 MHz E.T.
4	15 (4, 4, 4, 3)	3.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise
N is reset
C is data from bit 7 of source register

Example: If the contents of the HL register pair are 2828H, and the contents of memory location 2828H are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

at execution of `RLC(HL)` the contents of memory location 2828H and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLC (IX+d)

Operation:

(IX+d)
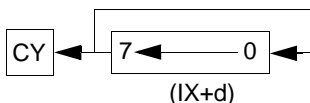
Op Code: RLC

Operands: (IX+d)

1	1	0	1	1	1	0	1	DD

1	1	0	0	1	0	1	1	CB

←			d				→	

0	0	0	0	0	1	1	0	06

Description: The contents of the memory address specified by the sum of the contents of the Index Register IX and a two's complement displacement integer d, are rotated left 1-bit position. The content of bit 7 is copied to the Carry flag and also to bit 0. Bit 0 is the least-significant bit.

M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise
N is reset
C is data from bit 7 of source register

Example: If the contents of the Index Register IX are 1000H, and the contents of memory location 1022H are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

at execution of `RLC (IX+2H)` the contents of memory location `1002H` and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLC (IY+d)

Operation:
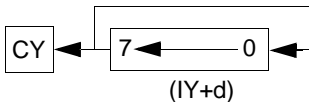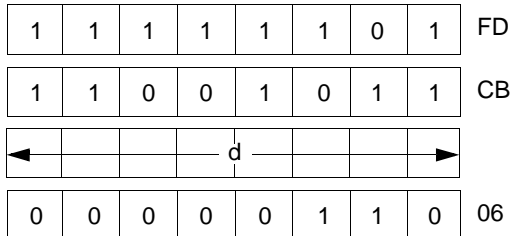
(IY+d)

Op Code: RLC

Operands: (lY+d)

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | FD |

| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | CB |

| ← | | | d | | | | → | |

| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 06 |

Description: The contents of the memory address specified by the sum of the contents of the Index Register IY and a two's complement displacement integer d are rotated left 1-bit position. The content of bit 7 is copied to the Carry flag and also to bit 0. Bit 0 is the least-significant bit.

| M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise
N is reset
C is data from bit 7 of source register

Example: If the contents of the Index Register IY are `1000H`, and the contents of memory location `1002H` are

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

at execution of `RLC (IY+2H)` the contents of memory location `1002H` and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RL m

Operation:
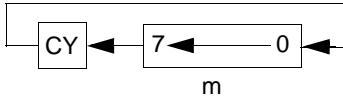
Op Code: PL

Operands: m

The m operand is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous PLC instructions. These possible Op Code/operand combinations are specified as follows in the assembled object code:

RL r*	1	1	0	0	1	0	1	1	CB
	0	0	0	1	0	←	r*	→	

RL (HL)	1	1	0	0	1	0	1	1	CB
	0	0	0	1	0	1	1	0	16

RL (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	←				d			→	
	0	0	0	1	0	1	1	0	16

RL (IY+d)	1	1	1	1	1	1	0	1	FB
	1	1	0	0	1	0	1	1	CB
	←				d			→	
	0	0	0	1	0	1	1	0	16

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The contents of the m operand are rotated left 1-bit position. The content of bit 7 is copied to the Carry flag and the previous content of the Carry flag is copied to bit 0.

Instruction	M Cycles	T States	4 MHz E.T.
RL r	2	8 (4, 4)	2.00
RL (HL)	4	15(4, 4, 4, 3)	3.75
RL (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
RL (IY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise
N is reset
C is data from bit 7 of source register

Example: If the contents of register D and the Carry flag are

C	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1	1

at execution of `RL D` the contents of register D and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	1	0

RRC m

Operation:
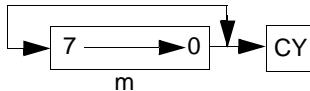
Op Code: RRC

Operands: m

The m operand is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous RLC instructions. These possible Op Code/operand combinations are specified as follows in the assembled object code:

RRC r*	1	1	0	0	1	0	1	1	CB
	0	0	0	0	1	◄— r* —►			
RRC (HL)	1	1	0	0	1	0	1	1	CB
	0	0	0	0	1	1	1	0	OE
RRC (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	◄—			d			—►		
	0	0	0	0	1	1	1	0	OE
RRC (IY+d)	1	1	1	1	1	1	0	1	FB
	1	1	0	0	1	0	1	1	CB
	◄—			d			—►		
	0	0	0	0	1	1	1	0	OE

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The contents of the m operand are rotated right 1-bit position. The content of bit 0 is copied to the Carry flag and also to bit 7. Bit 0 is the least-significant bit.

Instruction	M cycles	T States	4 MHz E.T.
RRC r	2	8 (4, 4)	2.00
RRC (HL)	4	15 (4, 4, 4, 3)	3.75
RRC (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
RRC (lY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise,
N is reset
C is data from bit 0 of source register

Example: If the contents of register A are

7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1

at execution of `RRC A` the contents of register A and the Carry flag are

7	6	5	4	3	2	1	0	C
1	0	0	1	1	0	0	0	1

RR m

Operation:

7 ———► 0 ► CY

m
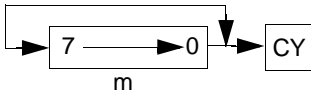
Op Code: RR

Operands: m

The m operand is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous RLC instructions. These possible Op Code/operand combinations are specified as follows in the assembled object code:

RR r*	1	1	0	0	1	0	1	1	CB
	0	0	0	0	1	◄—	r*	—►	

RR (HL)	1	1	0	0	1	0	1	1	CB
	0	0	0	0	1	1	1	0	1E

RR (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	◄—				d			—►	
	0	0	0	1	1	1	1	0	1E

RR (IY+d)	1	1	1	1	1	1	0	1	FD
	1	1	0	0	1	0	1	1	CB
	◄—				d			—►	
	0	0	0	1	1	1	1	0	1E

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: The contents of operand m are rotated right 1-bit position through the Carry flag. The content of bit 0 is copied to the Carry flag and the previous content of the Carry flag is copied to bit 7. Bit 0 is the least-significant bit.

Instruction	M Cycles	T States	4 MHz E.T.
RR r	2	8 (4, 4)	2.00
RR (HL)	4	15 (4, 4, 4, 3)	3.75
RR (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
RR (lY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity even; reset otherwise,
N is reset
C is data from bit 0 of source register

Example: If the contents of the HL register pair are 4343H, and the contents of memory location 4343H and the Carry flag are

7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	1	0

at execution of `RR (HL)` the contents of location 4343H and the Carry flag are

7	6	5	4	3	2	1	0	C
0	1	1	0	1	1	1	0	1

SLA m

Operation:
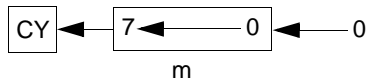
Op Code: SLA

Operands: m

The m operand is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous RLC instructions. These possible Op Code/operand combinations are specified as follows in the assembled object code :

SLA r*	1	1	0	0	1	0	1	1	CB
	0	0	1	0	0	←	r*	→	

SLA (HL)	1	1	0	0	1	0	1	1	CB
	0	0	1	0	0	1	1	0	26

SLA (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	←				d			→	
	0	0	1	0	0	1	1	0	26

SLA (IY+d)	1	1	1	1	1	1	0	1	FD
	1	1	0	0	1	0	1	1	CB
	←				d			→	
	0	0	1	0	0	1	1	0	26

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: An arithmetic shift left 1-bit position is performed on the contents of operand m. The content of bit 7 is copied to the Carry flag. Bit 0 is the least-significant bit.

Instruction	M Cycles	T States	4 MHz E.T.
SLA r	2	8 (4, 4)	2.00
SLA (HL)	4	15 (4, 4, 4, 3)	3.75
SLA (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
SLA (IY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity is even; reset otherwise
N is reset
C is data from bit 7

Example: If the contents of register L are

7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1

at execution of `SLA L` the contents of register L and the Carry flag are

C	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1	0

SRA m

Operation:

$$7 \longrightarrow 0 \longrightarrow CY$$

m
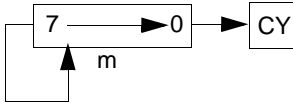
Op Code: SRA

Operands: m

The m operand is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous PLC instructions. These possible Op Code/operand combinations are specified as follows in the assembled object code:

SRA r*	1	1	0	0	1	0	1	1	CB
	0	0	1	0	0	←	r*	→	

SRA (HL)	1	1	0	0	1	0	1	1	CB
	0	0	1	0	1	1	1	0	2E

SRA (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	←				d			→	
	0	0	1	0	1	1	1	0	2E

SRA (IY+d)	1	1	1	1	1	1	0	1	FD
	1	1	0	0	1	0	1	1	CB
	←				d			→	
	0	0	1	0	1	1	1	0	2E

*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description: An arithmetic shift right 1-bit position is performed on the contents of operand m. The content of bit 0 is copied to the Carry flag and the previous content of bit 7 is unchanged. Bit 0 is the least-significant bit.

Instruction	M Cycles	T States	4 MHz E.T.
SRA r	2	8 (4, 4)	2.00
SRA (HL)	4	15 (4, 4, 4, 3)	3.75
SRA (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
SRA (lY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected:

S is set if result is negative; reset otherwise
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity is even; reset otherwise
N is reset
C is data from bit 0 of source register

Example: If the contents of the Index Register IX are 1000H, and the contents of memory location 1003H are

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0

at execution of `SRA (IX+3H)` the contents of memory location 1003H
and the Carry flag are

7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	0	0

SRL m

Operation:
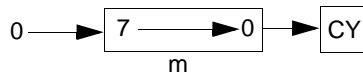
Op Code: SRL

Operands: m

The operand m is any of r, (HL), (IX+d), or (lY+d), as defined for the analogous RLC instructions. These possible Op Code/operand combinations are specified as follows in the assembled object code

SRL r*	1	1	0	0	1	0	1	1	CB

	0	0	1	1	1	←	r*	→	

SRL (HL)	1	1	0	0	1	0	1	1	CB

	0	0	1	1	1	1	1	0	3E

SRL (IX+d)	1	1	0	1	1	1	0	1	DD

	1	1	0	0	1	0	1	1	CB

	←				d			→	

	0	0	1	1	1	1	1	0	3E

SRL (IY+d)	1	1	1	1	1	1	0	1	FD

	1	1	0	0	1	0	1	1	CB

	←				d			→	

	0	0	1	1	1	1	1	0	3E
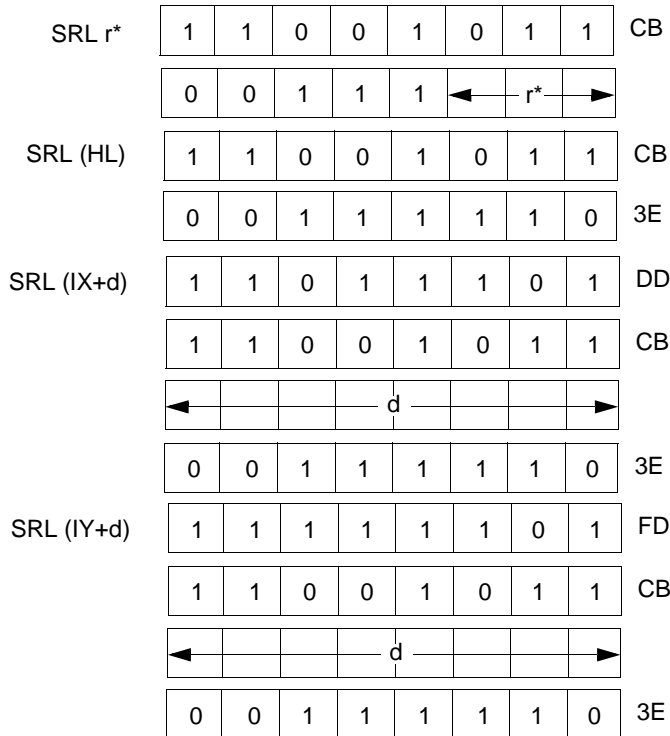
*r identifies registers B, C, D, E, H, L, or A assembled as follows in the object code field above:

Description: The contents of operand m are shifted right 1-bit position. The content of bit 0 is copied to the Carry flag, and bit 7 is reset. Bit 0 is the least-significant bit.

Instruction	M Cycles	T States	4 MHz E.T.
SRL r	2	8 (4, 4)	2.00
SRL (HL)	4	15 (4, 4, 4, 3)	3.75
SRL (1X+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

SRL (lY+d)   6   23 (4, 4, 3, 5, 4, 3)   5.75

Condition Bits Affected:

S is reset
Z is set if result is zero; reset otherwise
H is reset
P/V is set if parity is even; reset otherwise
N is reset
C is data from bit 0 of source register

Example: If the contents of register B are

7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	1

at execution of SRL B the contents of register B and the Carry flag are

7	6	5	4	3	2	1	0	C
0	1	0	0	0	1	1	1	1

RLD

Operation:

A | 7 4 | 3 0 | 7 4 | 3 0
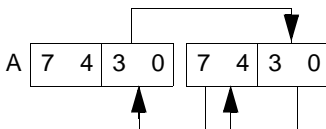
Op Code: RLD

Operands: —

1	1	1	0	1	1	0	1	ED

0	1	1	0	1	1	1	1	6F

Description: The contents of the low order four bits (bits 3, 2, 1, and 0) of the memory location (HL) are copied to the high order four bits (7, 6, 5, and 4) of that same memory location; the previous contents of those high order four bits are copied to the low order four bits of the Accumulator (register A); and the previous contents of the low order four bits of the Accumulator are copied to the low order four bits of memory location (HL). The contents of the high order bits of the Accumulator are unaffected.

> Note: (HL) means the memory location specified by the contents of the HL register pair.

M Cycles	T States	4 MHz E.T.
5	18 (4, 4, 3, 4, 3)	4.50

Condition Bits Affected:

S is set if Accumulator is negative after operation; reset otherwise
Z is set if Accumulator is zero after operation; reset otherwise
H is reset
P/V is set if parity of Accumulator is even after operation; reset otherwise
N is reset
C is not affected

Example: If the contents of the HL register pair are 5000H, and the contents of the
Accumulator and memory location 5000H are

7	6	5	4	3	2	1	0	
0	1	1	1	1	0	1	0	Accumulator

7	6	5	4	3	2	1	0	
0	0	1	1	0	0	0	1	(5000H)

at execution of RLD the contents of the Accumulator and memory location
5000H are

7	6	5	4	3	2	1	0	
0	1	1	1	0	0	1	1	Accumulator

7	6	5	4	3	2	1	0	
0	0	0	1	1	0	1	0	(5000H)

RRD

Operation:

A | 7 4 3 0 | 7 4 3 0 | (HL)
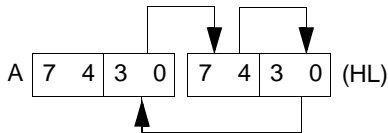
Op Code: RRD

Operands: —

1	1	1	0	1	1	0	1	ED

0	1	1	0	0	1	1	1	67

Description: The contents of the low order four bits (bits 3, 2, 1, and 0) of memory location (HL) are copied to the low order four bits of the Accumulator (register A). The previous contents of the low order four bits of the Accumulator are copied to the high order four bits (7, 6, 5, and 4) of location (HL); and the previous contents of the high order four bits of (HL) are copied to the low order four bits of (HL). The contents of the high order bits of the Accumulator are unaffected.

(HL) means the memory location specified by the contents of the HL register pair.

M Cycles	T States	4 MHz E.T.
5	18 (4, 4, 3, 4, 3)	4.50

Condition Bits Affected:

S is set if Accumulator is negative after operation; reset otherwise
Z is set if Accumulator is zero after operation; reset otherwise
H is reset
P/V is set if parity of Accumulator is even after operation; reset otherwise
N is reset
C is not affected

Example: If the contents of the HL register pair are 5000H, and the contents of the
Accumulator and memory location 5000H are

7	6	5	4	3	2	1	0	
1	0	0	0	0	1	0	0	Accumulator

7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	0	(5000H)

at execution of RRD the contents of the Accumulator and memory location
5000H are

7	6	5	4	3	2	1	0	
1	0	0	0	0	0	0	0	Accumulator

7	6	5	4	3	2	1	0	
0	1	0	0	0	0	1	0	(5000H)

Bit Set, Reset, and Test Group

BIT b, r

Operation: $Z \leftarrow \overline{rb}$

Op Code: BIT

Operands: b, r

1	1	0	0	1	0	1	1	CB

0	1	← b →	← r →
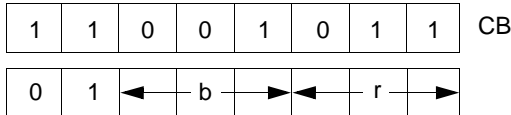
Description: This instruction tests bit b in register r and sets the Z flag accordingly. Operands b and r are specified as follows in the assembled object code:

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

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	4.50

Condition Bits Affected:

S is unknown
Z is set if specified bit is 0; reset otherwise
H is set
P/V is unknown
N is reset
C is not affected

Example: If bit 2 in register B contains 0, at execution of BIT 2, B the Z flag in the
F register contains 1, and bit 2 in register B remains 0. Bit 0 in register B is
the least-significant bit.

BIT b, (HL)

Operation: $Z \leftarrow (\overline{HL})b$

Op Code: BIT

Operands: b, (HL)

1	1	0	0	1	0	1	1	CB

0	1		b		1	1	0

Description: This instruction tests bit b in the memory location specified by the contents of the HL register pair and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
1	111

M Cycles	T States	4 MHz E.T.
3	12 (4, 4, 4) 4	3.00

Condition Bits Affected:

S is unknown
Z is set if specified Bit is 0; reset otherwise
H is set
P/V is unknown
H is reset
C is not affected

Example: If the HL register pair contains 4444H, and bit 4 in the memory location 444H contains 1, at execution of BIT 4, (HL) the Z flag in the F register contains 0, and bit 4 in memory location 4444H still contains 1. Bit 0 in memory location 4444H is the least-significant bit.

BIT b, (IX+d)

Operation: $Z \leftarrow \overline{(IX+d)}b$

Op Code: BIT

Operands: b, (IX+d)
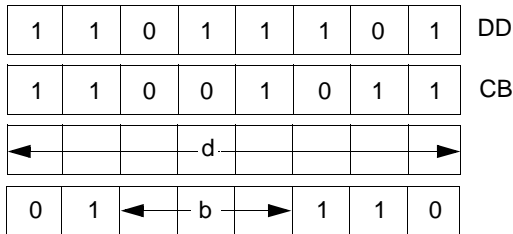
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | DD |

| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | CB |

| ← | | | d | | | | → |

| 0 | 1 | ← | b | → | 1 | 1 | 0 |

Description: This instruction tests bit b in the memory location specified by the contents of register pair IX combined with the two's complement displacement d and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code.

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M cycles	T States	4 MHz E.T.
5	20 (4, 4, 3, 5, 4)	5.00

Condition Bits Affected:

S is unknown
Z is set if specified Bit is 0; reset otherwise
H is set
P/V is unknown
N is reset
C is not affected

Example: If the contents of Index Register IX are `2000H`, and bit 6 in memory location `2004H` contains 1, at execution of `BIT 6, (IX+4H)` the Z flag in the F register contains 0, and bit 6 in memory location `2004H` still contains 1. Bit 0 in memory location `2004H` is the least-significant bit.

BIT b, (IY+d)

Operation: $Z \leftarrow \overline{(IY+d)b}$

Op Code: BIT

Operands: b, (lY+d)

1	1	1	1	1	1	0	1	FD

1	1	0	0	1	0	1	1	CB

◄──────────── d ────────────►

0	1	◄── b ──►	1	1	0
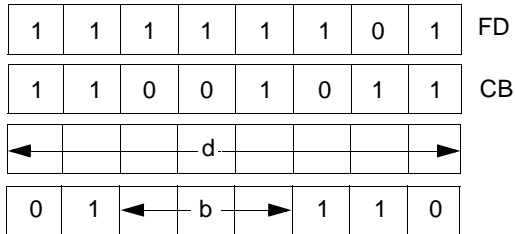
Description: This instruction tests bit b in the memory location specified by the content of register pair IY combined with the two's complement displacement d and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code.

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M Cycles	T States	4 MHz E.T.
5	20 (4, 4, 3, 5, 4)	5.00

Condition Bits Affected:

S is unknown
Z is set if specified Bit is 0; reset otherwise
H is set
P/V is unknown
H is reset
C is not affected

Example: If the contents of Index Register are 2000H, and bit 6 in memory location 2004H contains 1, at execution of BIT 6, (IY+4H) the Z flag and the F register still contain 0, and bit 6 in memory location 2004H still contains 1. Bit 0 in memory location 2004H is the least-significant bit.

SET b, r

Operation: rb ← 1

Op Code: SET

Operands: b, r

1	1	0	0	1	0	1	1	CB

1	1	◄— b —►	◄— r —►
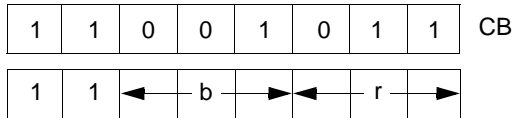
Description: Bit b in register r (any of registers B, C, D, E, H, L, or A) is set. Operands b and r are specified as follows in the assembled object code:

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

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

Condition Bits Affected: None

Example: At execution of SET 4, A bit 4 in register A sets. Bit 0 is the least-significant bit.

SET b, (HL)

Operation: (HL)b ← 1

Op Code: SET

Operands: b, (HL)

1	1	0	0	1	0	1	1	CB

1	1	◄— b —►	◄— r —►
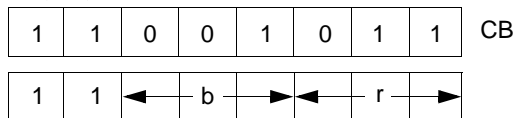
Description: Bit b in the memory location addressed by the contents of register pair HL is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M Cycles	T States	4 MHz E.T.
4	15 (4, 4, 4, 3)	3.75

Condition Bits Affected: None

Example: If the contents of the HL register pair are 3000H, at execution of SET 4, (HL) bit 4 in memory location 3000H is 1. Bit 0 in memory location 3000H is the least-significant bit.

SET b, (IX+d)

Operation: (IX+d)b ← 1

Op Code: SET

Operands: b, (IX+d)

Description: Bit b in the memory location addressed by the sum of the contents of the IX register pair and the two's complement integer d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected: None

Example: If the contents of Index Register are 2000H, at execution of SET 0, (IX + 3H) bit 0 in memory location 2003H is 1. Bit 0 in memory location 2003H is the least-significant bit.

SET b, (IY+d)

Operation: (IY + d) b ← 1

Op Code: SET

Operands: b, (IY + d)

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | FD |

| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | CB |

| ← | | | d | | | | → | |

| 1 | 1 | ← | b | → | 1 | 1 | 0 |

Description: Bit b in the memory location addressed by the sum of the contents of the IY register pair and the two's complement displacement d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M Cycles	T States	4 MHz E.T.
6	23 (4, 4, 3, 5, 4, 3)	5.75

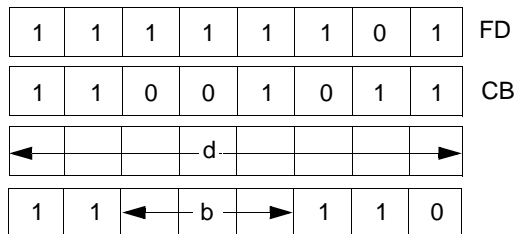**Condition Bits Affected:** None

Example: If the contents of Index Register IY are 2000H, at execution of SET 0, (IY+3H) bit 0 in memory location 2003H is 1. Bit 0 in memory location 2003H is the least-significant bit.

RES b, m

Operation: sb ← 0

Op Code: RES

Operands: b, m

Operand b is any bit (7 through 0) of the contents of the m operand, (any of r, (HL), (IX+d), or (lY+d)) as defined for the analogous SET instructions. These possible Op Code/operand combinations are assembled as follows in the object code:

RES b, rn

1	1	0	0	1	0	1	1	CB

1	0	◄—	b	—►	◄—	r	—►

RES b, (HL)

1	1	0	0	1	0	1	1	CB

1	0	◄—	b	—►	1	1	0

RES b, (IX+d)

1	1	0	1	1	1	0	1	DD

1	1	0	0	1	0	1	1	CB

◄—				d			—►

1	0	◄—	b	—►	1	1	0

RES b, (IY+d)

1	1	1	1	1	1	0	1	FD

1	1	0	0	1	0	1	1	CB

◄—				d			—►

1	0	◄—	b	—►	1	1	0
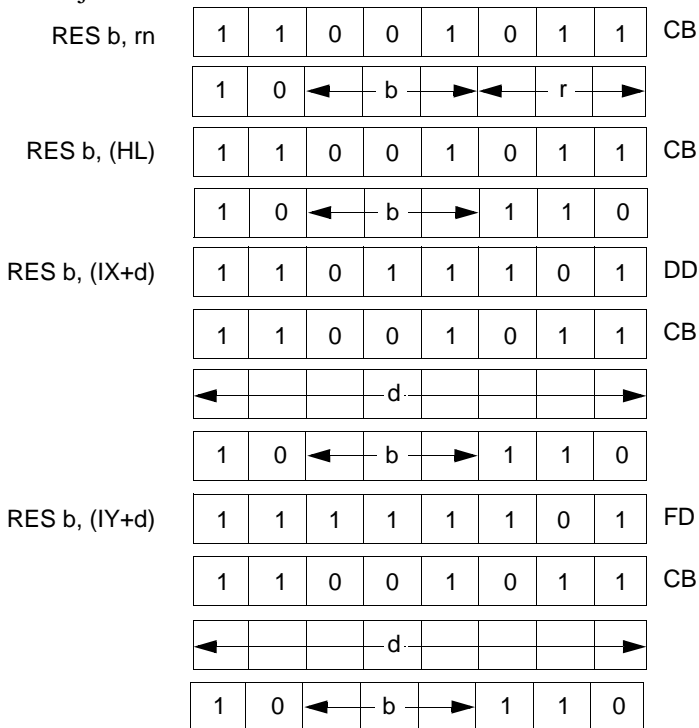
Bit	b	Register	r
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

Description: Bit b in operand m is reset.

Instruction	M Cycles	T States	4 MHz E.T.
RES r	4	8 (4, 4)	2.00
RES (HL)	4	15 (4, 4, 4, 3)	3.75
RES (IX+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75
RES (IY+d)	6	23 (4, 4, 3, 5, 4, 3)	5.75

Condition Bits Affected: None

Example: At execution of RES 6, D, bit 6 in register 0 resets. Bit 0 in register D is the least-significant bit.

Jump Group

JP nn

Operation: PC ← nn

Op Code: JP

Operands: nn

| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | C3 |

| ◄ | | | n | | | | ► |

| ◄ | | | n | | | | ► |
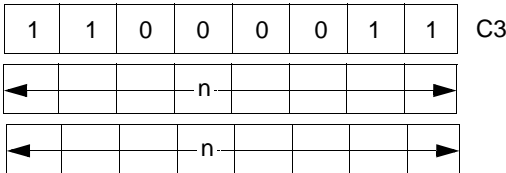
> Note: The first operand in this assembled object code is the low order byte of a two-byte address.

Description: Operand nn is loaded to register pair PC (Program Counter). The next instruction is fetched from the location designated by the new contents of the PC.

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

Condition Bits Affected: None

JP cc, nn

Operation: IF cc true, PC ← nn

Op Code: JP

Operands: cc, nn

1	1	←	00	→	0	1	0

←			n				→

←			n				→

The first n operand in this assembled object code is the low order byte of a 2-byte memory address.

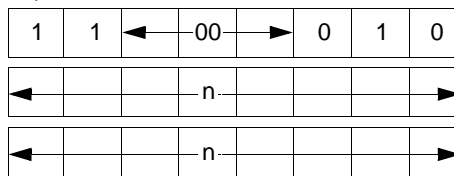**Description:** If condition cc is true, the instruction loads operand nn to register pair PC (Program Counter), and the program continues with the instruction beginning at address nn. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status that corresponds to condition bits in the Flag Register (register F). These eight status are defined in the table below that also specifies the corresponding cc bit fields in the assembled object code.

cc	Condition	Relevant Flag
000	NZ non zero	Z
001	Z zero	Z
010	NC no carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

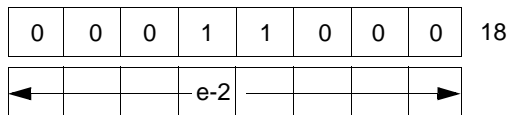**Condition Bits Affected:** None

Example: If the Carry flag (C flag in the F register) is set and the contents of address 1520 are 03H, at execution of JP C, 1520H the Program Counter contains 1520H, and on the next machine cycle the CPD fetches byte 03H from address 1520H.

JR e

Operation: PC ← PC + e

Op Code: JR

Operands: e

0	0	0	1	1	0	0	0	18

◄──────── e-2 ────────►

Description: This instruction provides for unconditional branching to other segments of a program. The value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. This jump is measured from the address of the instruction Op Code and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

M Cycles	**T States**	**4 MHz E.T.**
3	12 (4, 3, 5)	3.00

Condition Bits Affected: None

Example: To jump forward five locations from address 480, the following assembly language statement is used   JR $+5

The resulting object code and final PC value is shown below:

Location	**Instruction**
480	18
481	03
482	-
483	-
484	-
485	← PC after jump

JR C, e

Operation: If C = 0, continue

 If C = 1, PC ← PC+ e

Op Code: JR

Operands: C, e
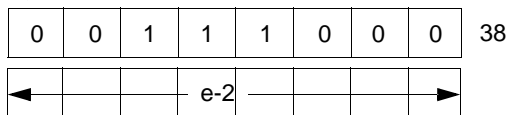
0	0	1	1	1	0	0	0	38

←			e-2				→

Description: This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to a 1, the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction Op Code and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a 0, the next instruction executed is taken from the location following this instruction. If condition is met

M Cycles	T States	4 MHz E.T.
3	12 (4, 3, 5)	3.00

If condition is not met:

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: The Carry flag is set and it is required to jump back four locations from 480. The assembly language statement is JR C, $ - 4

The resulting object code and final PC value is shown below:

Location	Instruction
47C	← PC after jump
47D	-
47E	-
47F	-
480	38
481	FA (two's complement - 6)

JR NC, e

Operation: If C = 1, continue
If C = 0, PC ← PC + e

Op Code: JR

Operands: NC, e
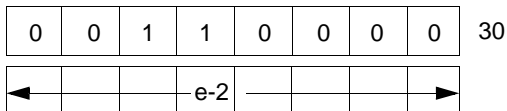
0	0	1	1	0	0	0	0	30

◄			e-2				►

Description: This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to 0, the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction Op Code and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a 1, the next instruction executed is taken from the location following this instruction.

If the condition is met:

M Cycles	T States	4 MHz E.T.
3	12 (4, 3, 5)	3.00

If the condition is not met:

M Cycles	T States	4 MHz E.T.
7	7 (4, 3)	1.75

Condition Bits Affected: None

Example: The Carry Flag is reset and it is required to repeat the jump instruction. The assembly language statement is JR NC, $

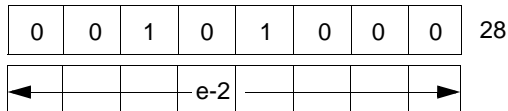The resulting object code and PC after the jump are:

Location	Instruction
480	30 ← PC after jump
481	00

JR Z, e

Operation: If $Z = 0$, continue
                  If $Z = 1$, $PC \leftarrow PC + e$

Op Code: JR

Operands: Z, e

0	0	1	0	1	0	0	0	28

◄──			e-2				──►

Description: This instruction provides for conditional branching to other segments of a
                   program depending on the results of a test on the Zero Flag. If the flag is
                   equal to a 1, the value of the displacement e is added to the Program
                   Counter (PC) and the next instruction is fetched from the location
                   designated by the new contents of the PC. The jump is measured from the
                   address of the instruction Op Code and has a range of -126 to +129 bytes.
                   The assembler automatically adjusts for the twice incremented PC.

   If the Zero Flag is equal to a 0, the next instruction executed is taken from
   the location following this instruction. If the condition is met:

M Cycles	T States	4 MHz E.T.
3	12 (4, 3, 5)	3.00

   If the condition is not met;

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: The Zero Flag is set and it is required to jump forward five locations
                from address 300. The following assembly language statement is used
                JR Z , $ + 5

   The resulting object code and final PC value is:

Location	Instruction
300	28
301	03
302	-
303	-
304	-
305	← PC after jump

JR NZ, e

Operation: If Z = 1, continue
If Z = 0, PC ← pc + e

Op Code: JR

Operands: NZ, e
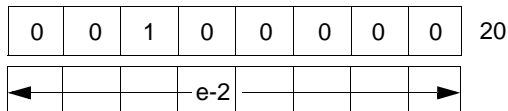
0	0	1	0	0	0	0	0	20

◄———————— e-2 ————————►

Description: This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a 0, the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction Op Code and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a 1, the next instruction executed is taken from the location following this instruction.

If the condition is met:

M Cycles	T States	4 MHz E.T.
3	12 (4, 3, 5)	3.00

If the condition is not met:

M Cycles	T States	4 MHz E.T.
2	7 (4, 3)	1.75

Condition Bits Affected: None

Example: The Zero Flag is reset and it is required to jump back four locations from 480. The assembly language statement is JR NZ, $ - 4

The resulting object code and final PC value is:

Location	**Instruction**
47C	← PC after jump
47D	-
47E	-
47F	-
480	20
481	FA (two's complement - 6)

JP (HL)

Operation: pc ← hL

Op Code: JP

Operands: (HL)

1	1	1	0	1	0	0	1	E9

Description: The Program Counter (register pair PC) is loaded with the contents of the HL register pair. The next instruction is fetched from the location designated by the new contents of the PC.

M Cycles	T States	4 MHz E.T.
1	4	1.00

Condition Bits Affected: None

Example: If the contents of the Program Counter are 1000H, and the contents of the HL register pair are 4800H, at execution of JP (HL) the contents of the Program Counter are 4800H.

JP (IX)

Operation: pc ← IX

Op Code: JP

Operands: (IX)

1	1	0	1	1	1	0	1	DD

1	1	1	0	1	0	0	1	E9

Description: The Program Counter (register pair PC) is loaded with the contents of the IX Register Pair. The next instruction is fetched from the location designated by the new contents of the PC.

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

Condition Bits Affected: None

Example: If the contents of the Program Counter are `1000H`, and the contents of the IX Register Pair are `4800H`, at execution of `JP (IX)` the contents of the Program Counter are `4800H`.

JP (IY)

Operation: PC ← IY

Op Code: JP

Operands: (IY)

1	1	1	1	1	1	0	1	FD

1	1	1	0	1	0	0	1	E9

Description: The Program Counter (register pair PC) is loaded with the contents of the IY Register Pair. The next instruction is fetched from the location designated by the new contents of the PC.

M Cycles	T States	4 MHz E.T.
2	8 (4, 4)	2.00

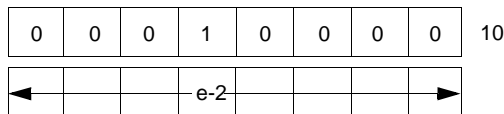**Condition Bits Affected:** None

Example: If the contents of the Program Counter are 1000H, and the contents of the IY Register Pair are 4800H, at execution of JP (IY) the contents of the Program Counter are 4800H.

DJNZ, e

Operation: -

Op Code: DJNZ

Operands: e

| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 10 |

| ← | | | e-2 | | | | → |

Description: This instruction is similar to the conditional jump instructions except that a register value is used to determine branching. The B register is decremented, and if a non zero value remains, the value of the displacement e is added to the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction Op Code and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the result of decrementing leaves B with a zero value, the next instruction executed is taken from the location following this instruction.

if B ≠ 0:

M Cycles	**T States**	**4 MHz E.T.**
3	13 (5,3, 5)	3.25

If B = 0:

M Cycles	**T States**	**4 MHz E.T.**
2	8 (5, 3)	2.00

Condition Bits Affected: None

Example: A typical software routine is used to demonstrate the use of the DJNZ instruction. This routine moves a line from an input buffer (INBUF) to an output buffer (OUTBUF). It moves the bytes until it finds a CR, or until it has moved 80 bytes, whichever occurs first.

```
                LD   8, 80        ;Set up counter
                LD   HL, Inbuf    ;Set up pointers
                LD   DE, Outbuf

        LOOP:   LID  A, (HL)      ;Get next byte from
                                  ;input buffer
                LD   (DE), A      ;Store in output buffer
                CP   ODH          ;Is it a CR?
                JR   Z, DONE      ;Yes finished
                INC  HL           ;Increment pointers
                INC  DE
                DJNZ LOOP         ;Loop back if 80
                                  ;bytes have not
                                  ;been moved
        DONE:
```
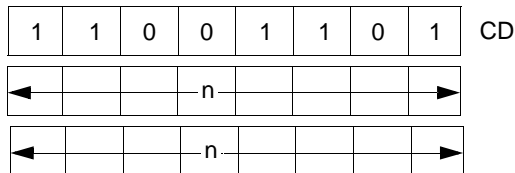
Call And Return Group

CALL nn

Operation: (SP-1) ← PCH, (SP-2) ← PCL, PC ← nn

Op Code: CALL

Operands: nn

1	1	0	0	1	1	0	1	CD

◄———————— n ————————►

◄———————— n ————————►

The first of the two n operands in the assembled object code above is the least-significant byte of a 2-byte memory address.

Description: The current contents of the Program Counter (PC) are pushed onto the top of the external memory stack. The operands nn are then loaded to the PC to point to the address in memory where the first Op Code of a subroutine is to be fetched. At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back to the PC. The push is accomplished by first decrementing the current contents of the Stack Pointer (register pair SP), loading the high-order byte of the PC contents to the memory address now pointed to by the SP; then decrementing SP again, and loading the low order byte of the PC contents to the top of stack.

Because this is a 3-byte instruction, the Program Counter was incremented by three before the push is executed.

M Cycles	T States	4 MHz E.T.
5	17 (4, 3, 4, 3, 3)	4.25

Condition Bits Affected: None

Example: If the contents of the Program Counter are `1A47H`, the contents of the Stack Pointer are `3002H`, and memory locations have the contents:

> `1A47H`   contains `CDH`
> `IA48H`   contains `35H`
> `1A49H`   contains `21H`

If an instruction fetch sequence begins, the 3-byte instruction `CD3521H` is fetched to the CPU for execution. The mnemonic equivalent of this is `CALL 2135H`. At execution of this instruction, the contents of memory address `3001H` is `1AH`, the contents of address `3000H` is `4AH`, the contents of the Stack Pointer is `3000H`, and the contents of the Program Counter is `2135H`, pointing to the address of the first Op Code of the subroutine now to be executed.

CALL cc, nn

Operation: IF cc true: (sp-1) ← PCH

(sp-2) ← PCL, pc ← nn

Op Code: CALL

Operands: cc, nn

1	1	←	cc	→	1	0	0

←			n				→

←			n				→
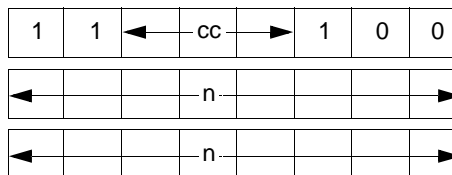
Note: The first of the two n operands in the assembled object code above is the least-significant byte of the 2-byte memory address.

Description: If condition cc is true, this instruction pushes the current contents of the Program Counter (PC) onto the top of the external memory stack, then loads the operands nn to PC to point to the address in memory where the first Op Code of a subroutine is to be fetched. At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back to PC. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. The stack push is accomplished by first decrementing the current contents of the Stack Pointer (SP), loading the high-order byte of the PC contents to the memory address now pointed to by SP; then decrementing SP again, and loading the low order byte of the PC contents to the top of the stack.

Because this is a 3-byte instruction, the Program Counter was incremented by three before the push is executed.

Condition cc is programmed as one of eight status that corresponds to condition bits in the Flag Register (register F). These eight status are

defined in the table below, which also specifies the corresponding cc bit
fields in the assembled object code:

cc	Condition	Relevant Flag
000	NZ non zero	Z
001	Z zero	Z
010	NC non carry	C
011	C carry	Z
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

If cc is true:

M Cycles	T States	4 MHz E.T.
5	17 (4, 3, 4, 3, 3)	4.25

If cc is false:

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

Condition Bits Affected: None

Example: If the C Flag in the F register is reset, the contents of the Program Counter
are 1A47H, the contents of the Stack Pointer are 3002H, and memory
locations have the contents:

Location	Contents
1A47H	D4H
1448H	35H
1A49H	21H

then if an instruction fetch sequence begins, the 3-byte instruction
D43521H is fetched to the CPU for execution. The mnemonic equivalent of
this is CALL NC, 2135H. At execution of this instruction, the contents of
memory address 3001H is 1AH, the contents of address 3000H is 4AH, the

contents of the Stack Pointer is 3000H, and the contents of the Program Counter is 2135H, pointing to the address of the first Op Code of the subroutine now to be executed.

RET

Operation: pCL ← (sp), pCH ← (sp+1)

Op Code: RET

| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | C9 |

Description: The byte at the memory location specified by the contents of the Stack Pointer (SP) register pair is moved to the low order eight bits of the Program Counter (PC). The SP is now incremented and the byte at the memory location specified by the new contents of this instruction is fetched from the memory location specified by the PC. This instruction is normally used to return to the main line program at the completion of a routine entered by a CALL instruction.

M Cycles	T States	4 MHz E.T.
3	10 (4, 3, 3)	2.50

**Condition Bits Affected:** None

Example: If the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location of memory location 2001H are 18H. At execution of RET the contents of the Stack Pointer is 2002H, and the contents of the Program Counter is 18B5H, pointing to the address of the next program Op Code to be fetched.

RET cc

Operation: If cc true: PCL ← (sp), pCH ← (sp+1)

Op Code: RET

Operands: cc

| 1 | 1 | ◄— cc —► | 0 | 0 | 0 |
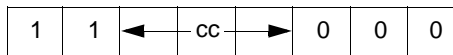
Description: If condition cc is true, the byte at the memory location specified by the contents of the Stack Pointer (SP) register pair is moved to the low order eight bits of the Program Counter (PC). The SP is incremented and the byte at the memory location specified by the new contents of the SP are moved to the high order eight bits of the PC. The SP is incremented again. The next Op Code following this instruction is fetched from the memory location specified by the PC. This instruction is normally used to return to the main line program at the completion of a routine entered by a CALL instruction. If condition cc is false, the PC is simply incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status that correspond to condition bits in the Flag Register (register F). These eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code.

	cc	**Condition**	**Relevant Flag**
	000	NZ non zero	Z
	001	Z zero	Z
	010	NC non carry	C
	011	C carry	C
	100	PO parity odd	P/V
	101	PE parity even	P/V
	110	P sign positive	S
	111	M sign negative	S

If cc is true:

M Cycles	**T States**	**4 MHz E.T.**
3	11 (5, 3, 3)	2.75

If cc is false:

M Cycles	**T States**	**4 MHz E.T.**
1	5	1.25

Condition Bits Affected: None

Example: If the S flag in the F register is set, the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H. At execution of RET M the contents of the Stack Pointer is 2002H, and the contents of the Program Counter is 18B5H, pointing to the address of the next program Op Code to be fetched.

RETI

Operation: Return from Interrupt

Op Code: RETI

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | ED |

| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 4D |

Description: This instruction is used at the end of a maskable interrupt service routine to:

- Restore the contents of the Program Counter (PC) (analogous to the RET instruction)

- Signal an I/O device that the interrupt routine is completed. The RETI instruction also facilitates the nesting of interrupts, allowing higher priority devices to temporarily suspend service of lower priority service routines. However, this instruction does not enable interrupts that were disabled when the interrupt routine was entered. Before doing the RETI instruction, the enable interrupt instruction (EI) should be executed to allow recognition of interrupts after completion of the current service routine.

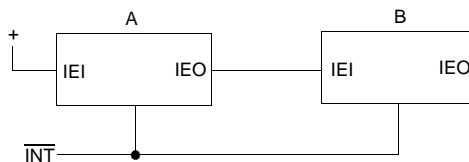| M Cycles	T States	4 MHz E.T.
4	14 (4, 4, 3, 3)	3.50

Condition Bits Affected: None

Example: Given: Two interrupting devices, with A and B connected in a daisy-chain configuration and A having a higher priority than B.

B generates an interrupt and is acknowledged. The interrupt enable out, IEO, of B goes Low, blocking any lower priority devices from interrupting while B is being serviced. Then A generates an interrupt, suspending service of B. The IEO of A goes Low, indicating that a higher priority device is being serviced. The A routine is completed and a RETI is issued resetting the IEO of A, allowing the B routine to continue. A second RETI is issued on completion of the B routine and the IE0 of B is reset (high) allowing lower priority devices interrupt access.

RETN

Operation: Return from non maskable interrupt

Op Code: RETN

1	1	1	0	1	1	0	1	ED

0	1	0	0	0	1	0	1	45

Description: This instruction is used at the end of a non-maskable interrupts service routine to restore the contents of the Program Counter (PC) (analogous to the RET instruction). The state of IFF2 is copied back to IFF1 so that maskable interrupts are enabled immediately following the RETN if they were enabled before the nonmaskable interrupt.

M Cycles	T States	4 MHz E.T.
4	14 (4, 4, 3, 3)	3.50

Condition Bits Affected: None

Example: If the contents of the Stack Pointer are 1000H, and the contents of the Program Counter are 1A45H, when a non maskable interrupt (NMI) signal is received, the CPU ignores the next instruction and instead restarts to memory address 0066H. The current Program Counter contents of 1A45H is pushed onto the external stack address of 0FFFH and 0FFEH, high order-byte first, and 0066H is loaded onto the Program Counter. That address begins an interrupt service routine that ends with a RETN instruction. Upon the execution of RETN the former Program Counter contents are popped off the external memory stack, low order first, resulting in a Stack Pointer contents again of 1000H. The program flow continues where it left off with an Op Code fetch to address 1A45H, order-byte first, and 0066H is loaded onto the Program Counter. That address begins an interrupt service routine that ends with a RETN instruction. At execution of RETN the former Program Counter contents are popped off the external memory stack, low order first, resulting in a Stack Pointer contents again of
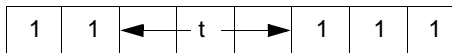
1000H. The program flow continues where it left off with an Op Code fetch to address 1A45H.

RST p

Operation: (SP-1) ← PCH, (SP-2) ← PCL, PCH ← 0, PCL ← P

Op Code: RST

Operands: p

1	1	◄— t —►	1	1	1

Description: The current Program Counter (PC) contents are pushed onto the external memory stack, and the page zero memory location given by operand p is loaded to the PC. Program execution then begins with the Op Code in the address now pointed to by PC. The push is performed by first decrementing the contents of the Stack Pointer (SP), loading the high-order byte of PC to the memory address now pointed to by SP, decrementing SP again, and loading the low order byte of PC to the address now pointed to by SP. The Restart instruction allows for a jump to one of eight addresses indicated in the table below. The operand p is assembled to the object code using the corresponding T state.

Because all addresses are in page zero of memory, the high order byte of PC is loaded with 00H. The number selected from the p column of the table is loaded to the low order byte of PC.

p	t
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

M Cycles	T States	4 MHz E.T.
3	11 (5, 3, 3)	2.75

Example: If the contents of the Program Counter are 15B3H, at execution of
RST 18H (Object code 1101111) the PC contains 0018H, as the address
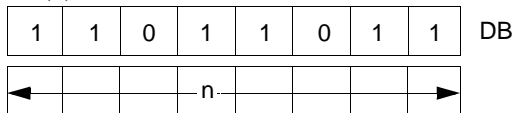of the next Op Code fetched.

Input and Output Group

IN A, (n)

Operation: A ← (n)

Op Code: IN

Operands: A, (n)

1	1	0	1	1	0	1	1	DB

|←————————— n —————————→|

Description: The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also appear on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the Accumulator (register A) in the CPU.

M Cycles	T States	4 MHz LT.
3	11 (4, 3, 4)	2.75

Condition Bits Affected: None

Example: If the contents of the Accumulator are 23H, and byte 7BH is available at the peripheral device mapped to I/O port address 01H. At execution of INA, (01H) the Accumulator contains 7BH.
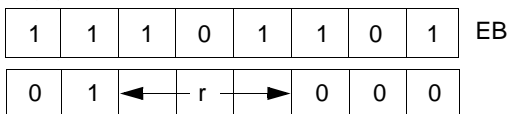
IN r (C)

Operation: r ← (C)

Op Code: IN

Operands: r, (C)

1	1	1	0	1	1	0	1	EB

0	1	◄── r ──►			0	0	0

Description: The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to register r in the CPU. Register r identifies any of the CPU registers shown in the following table, which also indicates the corresponding 3-bit r field for each. The flags are affected, checking the input data.

Register	r	
Flag	110	Undefined Op Code, set the flag
B	000	
C	001	
D	010	
E	011	
H	100	
L	101	
A	111	

M Cycles	T States	4 MHz E.T.
3	12 (4, 4, 4)	3.00

Condition Bits Affected:

S is set if input data is negative; reset otherwise
Z is set if input data is zero; reset otherwise
H is reset
P/V is set if parity is even; reset otherwise
N is reset
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 10H, and byte 7BH is available at the peripheral device mapped to I/O port address 07H. After execution of IN D, (C) register D contains 7BH.

INI

Operation: (HL) ← (C), B ← B -1, HL ← HL + 1

Op Code: INI

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | ED |

| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | A2 |

Description: The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are then placed on the address bus and the input byte is written to the corresponding location of memory. Finally, the byte counter is decremented and register pair HL is incremented.

M Cycles	**T States**	**4 MHz E.T.**
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

S is unknown
Z is set if B–1 = 0, reset otherwise
H is unknown
P/V is unknown
N is set
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and byte 7BH is available at the peripheral device mapped to I/O port address 07H. At execution of INI memory location 1000H contains 7BH, the HL register pair contains 1001H, and register B contains 0FH.

INIR

Operation: $(HL) \leftarrow (C), B \leftarrow B - 1, HL \leftarrow HL + 1$

Op Code: INIR

1	1	1	0	1	1	0	1	ED

1	0	1	1	0	0	1	0	B2

Description: The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written to the corresponding location of memory. Then register pair HL is incremented, the byte counter is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Interrupts are recognized and two refresh cycles execute after each data transfer.

Note: if B is set to zero prior to instruction execution, 256 bytes of data are input.

If B ≠ 0:

M Cycles	T States	4 MHz E.T.
5	21 (4, 5, 3, 4, 5)	5.25

If B = 0:

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

S is unknown
Z is set
H is unknown
P/V is unknown

N is set
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port of address 07H:

> 51H
> A9H
> 03H

then at execution of INIR the HL register pair contains 1003H, register B contains zero, and memory locations contain the following:

> 1000H   contains 51H
> 1001H   contains A9H
> 1002H   contains 03H

IND

Operation: $(HL) \leftarrow (C), B \leftarrow B -1, HL \leftarrow HL -1$

Op Code: IND

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	1	0	AA

Description: The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written to the corresponding location of memory. Finally, the byte counter and register pair HL are decremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

S is unknown
Z is set if B–1 = 0; reset otherwise
H is unknown
P/V is unknown
N is set
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and byte 7BH is available at the peripheral device mapped to I/O port address 07H. At execution of IND memory location 1000H contains 7BH, the HL register pair contains 0FFFH, and register B contains 0FH.

INDR

Operation: (HL) ← (C), B ← 131, HL ← HL1

Op Code: INDR

1	1	1	0	1	1	0	1	ED

1	0	1	1	1	0	1	0	BA

Description: The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written to the corresponding location of memory. Then HL and the byte counter are decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Interrupts are recognized and two refresh cycles are executed after each data transfer.

When B is set to zero prior to instruction execution, 256 bytes of data are input.

If B ≠ 0

M Cycles	T States	4 MHz E.T.
5	21 (4, 5, 3, 4, 5)	5.25

If B = 0:

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

S is unknown
Z is set
H is unknown
P/V is unknown
N is set

C is not affected

Example: If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port address 07H:

        51H
        A9H
        03H

then at execution of INDR the HL register pair contains 0FFDH, register B contains zero, and memory locations contain the following:
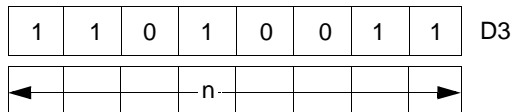
        0FFEH   contains 03H
        0FFFH   contains A9H
        1000H   contains 51H

OUT (n), A

Operation: (n) ← A

Op Code: OUT

Operands: (n), A

| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | D3 |

| ← | | | | n | | | → |

Description: The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator (register A) also appear on the top half (A8 through A15) of the address bus at this time. Then the byte contained in the Accumulator is placed on the data bus and written to the selected peripheral device.

M Cycles	T States	4 MHz E.T.
3	11 (4, 3, 4)	2.75

Condition Bits Affected: None

Example: If the contents of the Accumulator are 23H, at execution of OUT (01H), byte 23H is written to the peripheral device mapped to I/O port address 01H.
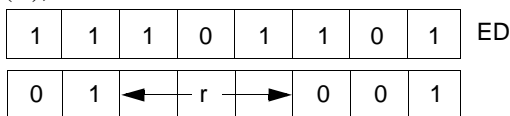
OUT (C), r

Operation: (C) ← r

Op Code: OUT

Operands: (C), r

1	1	1	0	1	1	0	1	ED

0	1	← r →	0	0	1

Description: The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then the byte contained in register r is placed on the data bus and written to the selected peripheral device. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding three-bit r field for each that appears in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M Cycles	T States	4 MHz E.T.
3	12 (4, 4, 4)	3.00

Condition Bits Affected: None

Example: If the contents of register C are 01H, and the contents of register D are 5AH, at execution of OUT (C),D byte 5AH is written to the peripheral device mapped to I/O port address 01H.

OUTI

Operation: $(C) \leftarrow (HL), B \leftarrow B - 1, HL \leftarrow HL + 1$

Op Code: OUTI

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | ED |

| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | A3 |

Description: The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus. The byte to be output is placed on the data bus and written to a selected peripheral device. Finally, the register pair HL is incremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

S is unknown
Z is set if B–1 = 0; reset otherwise
H is unknown
P/V is unknown
N is set
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 100014, and the contents of memory address 1000H are 5914, then after thee execution of OUTI register B contains 0FH, the HL register pair contains 1001H, and byte 59H is written to the peripheral device mapped to I/O port address 07H.

OTIR

Operation: $(C) \leftarrow (HL), B \leftarrow B -1, HL \leftarrow HL + 1$

Op Code: OTIR

1	1	1	0	1	1	0	1	ED

1	0	1	1	0	0	1	1	B3

Description: The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next, the byte to be output is placed on the data bus and written to the selected peripheral device. Then register pair HL is incremented. If the decremented B register is not zero, the Program Counter (PC) is decremented by two and the instruction is repeated. If B has gone to zero, the instruction is terminated. Interrupts are recognized and two refresh cycles are executed after each data transfer.

> Note: When B is set to zero prior to instruction execution, the instruction outputs 256 bytes of data.

If $B \neq 0$:

M Cycles	T States	4 MHz E.T.
5	21 (4, 5, 3, 4, 5)	5.25

If $B = 0$:

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

S is unknown
Z is set
H is unknown
P/V is unknown
N is set
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

>   1000H   contains 51H
>   1001H   contains A9H
>   1002H   contains 03H

then at execution of OTIR the HL register pair contains 1003H, register B contains zero, and a group of bytes is written to the peripheral device mapped to I/O port address 07H in the following sequence:

>   51H
>   A9H
>   03H

OUTD

Operation: $(C) \leftarrow (HL), B \leftarrow B -1, HL \leftarrow HL -1$

Op Code: OUTD

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	1	1	AB

Description: The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next, the byte to be output is placed on the data bus and written to the selected peripheral device. Finally, the register pair HL is decremented.

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3. 4)	4.00

Condition Bits Affected:

S is unknown
Z is set if B–1 = 0; reset otherwise
H is unknown
P/V is unknown
N is set
C is not affected

Example: If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory location 1000H are 59H, at execution of OUTD register B contains 0FH, the HL register pair contains 0FFFH, and byte 59H is written to the peripheral device mapped to I/O port address 07H.

OTDR

Operation: $(C) \leftarrow (HL), B \leftarrow B - 1, HL \leftarrow HL - 1$

Op Code: OTDR

1	1	1	0	1	1	0	1	ED

1	0	1	1	1	0	1	1	BB

Description: The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next, the byte to be output is placed on the data bus and written to the selected peripheral device. Then, register pair HL is decremented and if the decremented B register is not zero, the Program Counter (PC) is decremented by two and the instruction is repeated. If B has gone to zero, the instruction is terminated. Interrupts are recognized and two refresh cycles are executed after each data transfer.

> Note: When B is set to zero prior to instruction execution, the instruction outputs 256 bytes of data.

If $B \neq 0$:

M Cycles	T States	4 MHz E.T.
5	21 (4, 5, 3, 4, 5)	5.25

If $B = 0$:

M Cycles	T States	4 MHz E.T.
4	16 (4, 5, 3, 4)	4.00

Condition Bits Affected:

> S is unknown
> Z is set
> H is unknown
> P/V is unknown
> N is set
> C is not affected

Example: If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

> 0FFEH   contains 51H
> 0FFFH   contains A9H
> 1000H   contains 03H

then at execution of OTDR the HL register pair contain 0FFDH, register B contains zero, and a group of bytes is written to the peripheral device mapped to I/O port address 07H in the following sequence:

> 03H
> A9H
> 51H