MOSTEK®

Z80 MICROCOMPUTER SYSTEM

Programming Manual

Z80 PROGRAMMING MANUAL V2.0

Z80
PROGRAMMING MANUAL
V2.0
(MK78515)

TABLE OF CONTENTS

TABLE OF CONTENTS (Contd.)

TABLE OF CONTENTS (Contd.)

LIST OF FIGURES

LIST OF TABLES

PREFACE

This manual is designed to help the user program the Z80 microcomputer in assembly language. It also serves as a standard for the Z80 Assembly language.

It is assumed that the user has a background in logic and some experience with programming.

The manual consists mainly of a brief general description of the Z80 CPU architecture from the programmer's point of view and a detailed description of the Z80 instruction set. The description of the instruction set includes a description of the set's main features, specific information about assembly language syntax, and detailed descriptions of each of the Z80 opcodes. The manual also contains several appendices. Appendix A is an alphabetical list of the Z80 opcodes. Appendix B provides details of the Mostek assembler standard pseudo-ops. Appendix C describes the Mostek standard Z80 object code format. Appendix D provides binary, hexadecimal, and ASCII reference tables.

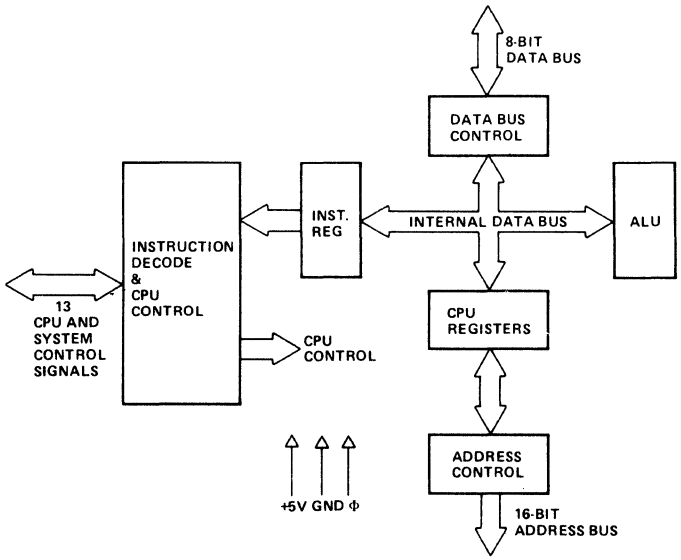
SECTION 1

Z80 CPU ARCHITECTURE

1-1. INTRODUCTION.

1-2. A block diagram of the internal architecture of the Z80 CPU is shown in Figure 1-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.

Figure 1-1. Z80 CPU Block Diagram



1-3. CPU REGISTERS.

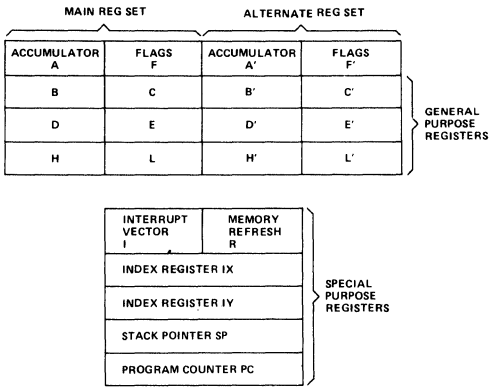
1-4. The Z80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 1-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z80 registers are implemented using static RAM. The registers include a set of special purpose registers, two sets of accumulator and flag registers, and two sets of six general purpose registers which may be used individually as 8-bit registers or in pairs as 16-bit registers.

1-5. SPECIAL PURPOSE REGISTERS.

1-6. 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.

1-7. 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 into 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.

Figure 1-2. Z80 CPU Register Configuration



1-8. Two Index Registers (IX & 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 displacment 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.

1-9. 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 to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

1-10. Memory Refresh Register (R). The Z80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. This 7-bit register is automatically incremented after each instruction fetch. 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 totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer.

1-11. ACCUMULATOR AND FLAG REGISTERS. The CPU includes two independent 8-bit accumulators and associated 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 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 that he wishes to work with a single exchange instruction so that he may easily work with either pair.

1-12. GENERAL PURPOSE REGISTERS. There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. 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 with through a single exchange command for the entire set. In systems where fast interrupt response is required one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange command need be executed to go between the 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 by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

1-13. ARITHMETIC & LOGIC UNIT (ALU).

1-14. 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 on the internal data bus. The type of 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

1-15. INSTRUCTION REGISTER AND CPU CONTROL.

1-16. As each instruction is fetched from memory, it is placed in the instructio register and decoded. The control section performs this function and then generate and supplies all of the control signals necessary to read or write data from or t the registers, controls the ALU and provides all required external control signals

SECTION 2

Z80 INSTRUCTION SET

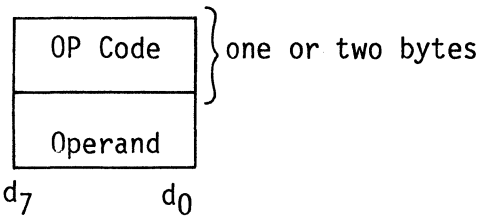
2-1. INTRODUCTION.

2-2. The Z80 instruction set of 158 instructions can best be described by first discussing in general the main features. These features include its addressing modes; status and flags; interrupt modes; load and exchange instructions; block transfer and search instructions; arithmetic and logical instructions; rotate and shift instructions; bit manipulation instructions; jump, call, and return instructions; input/output instruction; and miscellaneous instructions. Included in the discussion of the addressing modes are descriptions of subroutines and subroutine use of the stack. Following this general description of the instruction set, specific information about the syntax of the assembly language is provided. Then each instruction is described in detail in alphabetical order.

2-3. INSTRUCTION SET FEATURES.

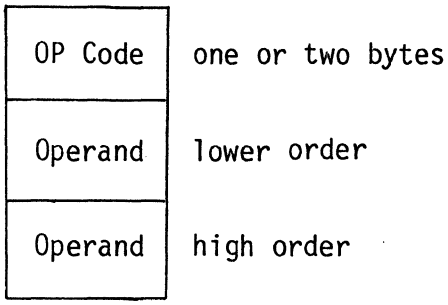
2-4. ADDRESSING MODES. Most of the Z80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. The following paragraphs gives a brief summary of the types of addressing used in the Z80. 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 destination.

2-5. Immediate Addressing. In this mode of addressing the byte following the Op code in memory contains the actual operand.



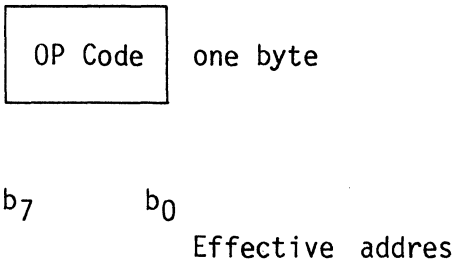
Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

2-6. Immediate Extended Addressing. This mode is merely an extension of immediate addressing in that the two bytes following the OP code are the operand.

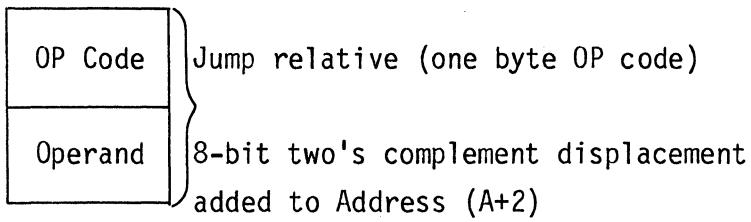


Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

2-7. Modified Page Zero Addressing. The Z80 has a special single byte call instruction to any of 8 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.



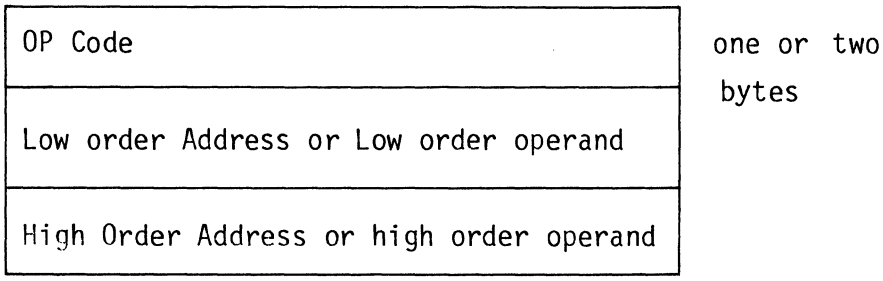
2-8. Relative Addressing. Relative addressing use one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.



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

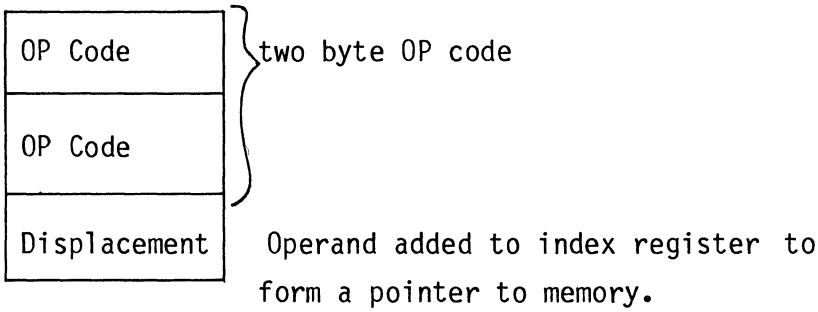
total displacement of +129 to -126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

2-9. 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.



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. When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to 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, (1200) refers to the contents of memory at location 1200.

2-10. Indexed Addressing. In this type of addressing, the byte of data following the Op code contains a displacement which 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.



2-11. An example of an indexed instruction would be 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 since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

2-12. The two index registers in the Z80 are referred to as IX and IY. To indicate indexed addressing the notation:

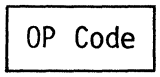
(IX+d) or (IY+d)

is used. Here d is the displacment specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

2-13. 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 would be to load the data in register B into register C.

2-14. 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 the destination of the results.

2-15. Register Indirect Addressing. This type of addressing specifies a 16-bit CUP 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.



one or two bytes

An example of this type of instruction would be 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 have 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 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.

2-16. 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, indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

2-17. Stack Pointer Addressing. Memory locations may be addressed in the 16-bit stack pointer register (SP). There are two stack operations which may be performed:

1. PUSH, which puts data into a stack,
2. POP, which retrieves data from a stack.

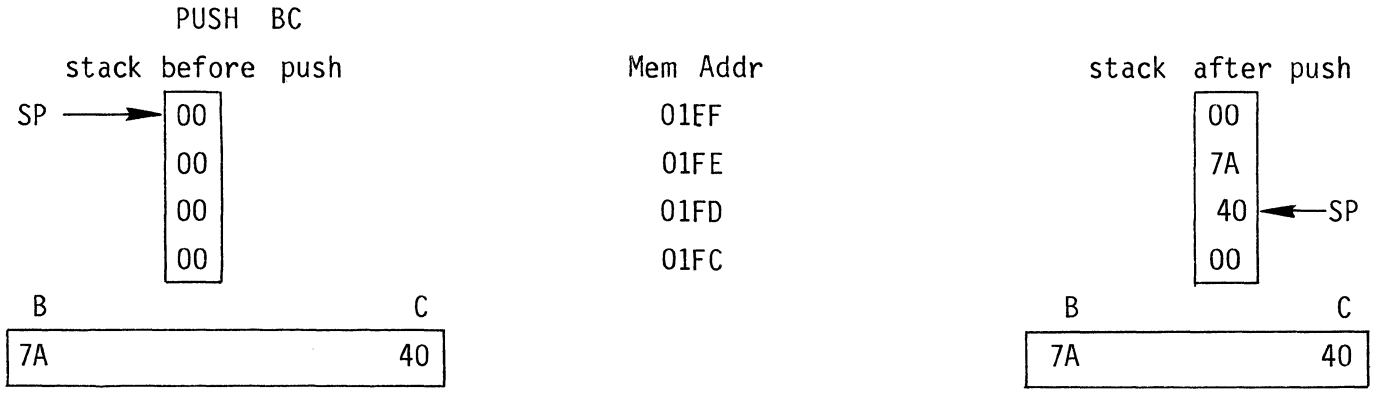
Note that the stack area must reside in read/write memory. The stack pointer is initialized to the top location in the stack at the start of a program. In a stack operation a 16-bit register pair is transferred to or from the stack.

2-18. For the PUSH operation the contents of the register pair are transferred to the Stack:

1. The most significant 8-bits of data are stored at the memory address less one than the contents of the stack pointer.

2. The least significant 8 bits of data are stored at the memory address less two than the contents of the stack pointer.

3. The stack pointer is automatically decremented by two.

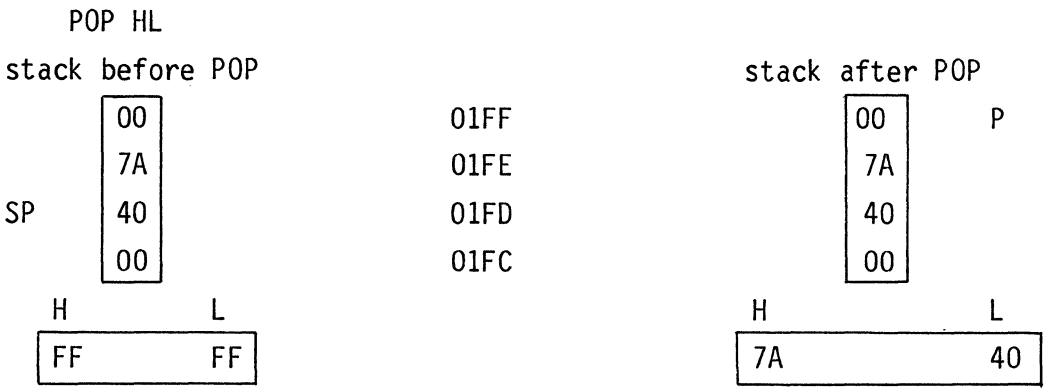


2-19. For the POP operation, 16 bits of data are taken from the stack and placed in the 16-bit register pair:

1. The second register of the pair (or the least significant byte of the pair) is loaded from the memory address held in the stack pointer.
2. The first register of the pair (or the most significant byte of the pair)

is loaded from the memory address one greater than the address held in the stack pointer.

3. The stack pointer is automatically incremented by two.



2-20. Subroutine Addressing.

2-21. Subroutines are blocks of instructions that can be called during the execution of a sequence of instructions. Subroutines can be called from main programs or from other subroutines. A subroutine is entered by the CALL opcode as in:

CALL REWIND

2-22. Parameters such as those used by the macros are not used with subroutines. When a call instruction is encountered during execution of a program, the PC is changed to the first instruction of the subroutine. The subsequent address of the invoking program is pushed on the stack. Control will return to this point when the subroutine is finished. The processor continues to execute the subroutine until it encounters a RET (return) instruction. At this point the return address is popped off the stack into the PC, and the processor returns to the address of the instruction following the CALL, to continue execution from that point.

2-23. Subroutines of any size can be invoked from programs or other subroutines of any size, without restriction. Care must be taken when nesting subroutines (subroutines within subroutines) that pushes and pops remain balanced at each level. If the processor encounters a RET with an un-popped push on the stack, the PC will be set to a meaningless address rather than to the next instruction following the CALL.

2-24. Tradeoffs must be considered between:

1. Using a block of code repetitively in line , and

2. calling the block repetitively as a subroutine.

2-25. Program size can usually by saved by using the subroutine. If the repetitive block contains N bytes and it is repeated on M occasions in the program,

1. MxN bytes would be used in direct programming, while
2. 3M (for CALLS)
   + n (for the block)
   + 1 (for the RET)
   =3M+N+1 bytes would be required if using a subroutine.

2-26. For example, for a block of 20 bytes used 5 times, in-line programming would require 100 bytes while a subroutine would require 36.

2-27. An added advantage of subroutines is that with careful naming, program structures become clearer, easier to read and easier to debug and maintain. Subroutines written for one purpose can be employed elsewhere in the programs requiring the same functions.

2-28. Subroutines differ from Macros in several ways:

1. Subroutine code is assembled into an object program only once although it may be called many times. Macro code is assembled in line every place the macro is used.
2. Registers and pointers required by a subroutine must be set up before the CALL. No parameters are used and no argument string can be issued. Macros, through their use of parameters, can modify the settings of registers on each occurrence.

2-29. Subroutine Use Of The Stack. When a call to a subroutine is executed, the contents of the program counter are pushed onto the stack automatically. Recall that the program counter contains the next memory address to be executed. After the PC is pushed onto the stack, the starting address of the subroutine is placed into the PC and then branch to the subroutine, a return instruction pops the address off the stack into the PC, and control is transferred to the memory address after the call. These operations are automatic when the CALL and RET instructions are executed.

2-30. Note that parameters can be passed to a subroutine because the stack and stack pointer can be manipulated and updated by special Z80 instructions.

2-31. The save type of operation as described for a subroutine also occurs for external interrupts monitored by the CPU.

2-32. **Z80 STATUS INDICATORS (FLAGS).**

2-33. Flag Register. The flag register (F and F') supplies information to the user regarding the status of the Z80 at any given time. The bit positions for each flag are shown below:

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

WHERE:

C = CARRY FLAG
N = ADD/SUBTRACT FLAG
P/V = PARITY/OVERFLOW FLAG
H = HALF-CARRY FLAG
Z = ZERO FLAG
S = SIGN FLAG
X = NOT USED

Each of the Z80 Flag Registers contains 6 bits of status information which are set or reset by CPU operations. (Bits 3 and 5 are not used.) Four of these bits are testable (C,P/V,Z and S) for use with conditional jump, call or return instructions. Two flags are not testable (H,N) and are used for BCD arithmetic.

2-34. Carry Flag (C). The carry bit is set or reset depending on the operation being performed. For ADD instructions that generate a carry and SUBTRACT instructions that generate a borrow, the Carry Flag will be set. The Carry Flag is reset by an ADD that does not generate a carry and a SUBTRACT that does not generate a borrow. This saved carry facilitates software routines for extended precision arithmetic. Also, the 'DAA' instruction will set the Carry Flag if the conditions for making the decimal adjustment are met.

2-35. For instructions RLA, RRA, RL and RR, the carry bit is used as a link bet-

ween the LSB and MSB for any register or memory location. During instructions RLCA, RLC s and SLA s, the carry contains the last value shifted out of bit 7 of any register or memory locations. During instructions RRCA, RRC s, SRA s and SRL s the carry contains the last value shifted out of bit 0 of any register or memory location.

2-36. For the logical instructions AND s, OR s and XOR s, the carry will be reset.

2-37. The Carry Flag can also be set (SCF) and complemented (CCF).

2-38. Add/Subtract Flag (N). This flag is used by the decimal adjust accumulator instruction (DAA) to distinguish between ADD and SUBTRACT instructions. For all ADD instructions, N will be set to a 1.

2-39. Parity/Overflow Flag. This flag is set to a particular state depending on the operation being performed.

2-40. For arithmetic operations, this flag indicates an overflow condition when the result in the Accumulator is greater than the maximum possible number (+127) or less than the minimum possible number(-128). This overflow condition can be determined by examining the sign bits of the operands.

2-41. For addition, operands with different signs will 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 adding of the two numbers together has resulted in a number that exceeds +127 and the two positive operands cause a negative number (-95) which is incorrect. The overflow flag is therefore set.

2-42. For subtraction, overflow can occur for operands of unlike signs. Operands of like sign will 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 therefore set. Another method for predicting an overflow is to observe the carry into and out of the sign bit. If there is a carry in and no carry out, or if there is no carry in and a carry out, then overflow has occurred.

2-43. This flag is also used with logical operations and rotate instructions to indicate the parity of the result. The number of 1 bits in a byte are counted. If the total is odd, ODD parity (P=0) is flagged. If the total is even, EVEN parity is flagged (P=1).

2-44. During search instructions CPI, CPIR, CPD, and CPDR and block transfer instructions LDI, LDIR, LDD, and LDDR, the P/V flag monitors the state of the byte count register (BC). When decrementing, the byte counter results in a zero value, the flag is reset to 0; otherwise, the flag is a 1.

2-45. During LD A,I and LD A,R instructions, the P/V flag will be set with the contents of the interrupt enable flip-flop (IFF2) for storage or testing.

2-46. When inputting a byte from an I/O device, IN r,(C), the flag will be adjusted to indicate the parity of the data.

2-47. Half Carry Flag (H). The Half Carry Flag (H) will be set or reset depending on the carry and borrow status between bits 3 and 4 of an 8-bit arithmetic operation. This flag is used by decimal adjust accumulator instruction (DAA) to correct the result of a packed BCD add or subtract operation. The H flag will be set (1) or reset (0) according to the following table:

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

2-48. Zero Flag (Z). The Zero Flag (Z) is set or reset if the result generated by the execution of certain instructions is a zero.

2-49. For 8-bit arithmetic and logical operations, the Z flag will be set to a 1 if the resulting byte in the Accumulator is zero. If the byte is not zero, the Z flag is reset to 0.

2-50. For compare (search) instructions, the Z flag will be set to a '1' if a comparison is made between the value in the Accumulator and the memory location pointed to by the contents of the register pair HL.

2-51. When testing a bit in a register or memory location, the Z flag will contain the complemented state of the indicated bit (see Bit b,s).

2-52. When inputting or outputting a byte between a memory location and an I/O device (INI;IND:OUTI and OUTD), if the result of B-1 is zero, the Z flag is set; otherwise, it is reset. Also for byte inputs from I/O devices using IN r,(C), the Z flag is set to indicate a zero byte input.

2-53. Sign Flag (S). 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, binary two's 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 two's complement of the equivalent positive number. The total range for negative numbers is from -1 to -128.

2-54. When inputting a byte from an I/O device to a register, IN r, (C), the S flag will indicate either positive (S=0) or negative (S=1) data. The state of the four testable flags is specified as follows:

FLAG	ON CONDITION	OFF CONDITION
Carry	C	NC
Zero	Z	NZ
Sign	M (minus)	P (plus)
Parity	PE (even)	PO (odd)

2-55. INTERRUPTS. The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

2-56. Interrupt Types.

Non-Maskable

A non-maskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

Maskable

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

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only needs to supply a single byte instruction. Alternatively, another 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 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. After the application of RESET the CPU will automatically enter interrupt Mode 0.

Mode 1

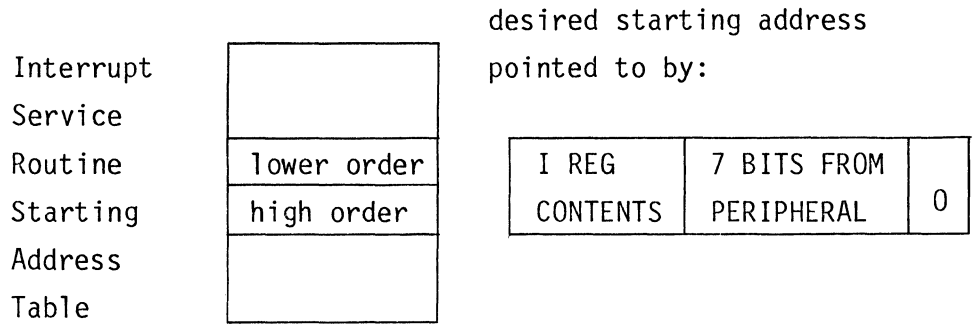
When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non-maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 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.

With 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 8 bits of this pointer are formed from the contents of the I Register. The I Register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that 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. Actually only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since 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.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once 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 does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address).

Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80 PIO, Z80 SIO, and Z80 CTC manuals for details.

2-57. Interrupt Enable - Disable.

2-58. The Z80 CPU has two interrupt inputs, a software maskable interrupt and a non-maskable interrupt. The non-maskable interrupt (NMI) can not be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved from very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt (INT) can be selectively enabled or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constants that

do not allow it to be interrupted. In the Z80 CPU there are enable flip flops (called IFF_1 and IFF_2) that are set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF_1 is reset, an interrupt cannot be accepted by the CPU. Table 2-1 summarizes the effect of the different instructions on the two enable flip flops.

2-59. There are two enable flip flops, called IFF_1 and IFF_2.

IFF_1	IFF_2
Actually disables interrupts from being accepted.	Temporary storage location for IFF_1.

The state of IFF_1 is used to actually inhibit interrupts while IFF_2 is used as a temporary storage location for IFF_1. The purpose of storing the IFF_1 will be subsequently explained.

2-60. A reset to the CPU will force both IFF_1 and IFF_2 to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF_1 and IFF_2 to the enable state. When an interrupt is accepted by the CPU, both IFF_1 and IFF_2 are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF_1 and IFF_2 are always equal.

2-61. The purpose of IFF_2 is to save the status of IFF_1 when a non-maskable interrupt occurs. When a non-maskable interrupt is accepted, IFF_1 is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non-maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF_1 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 IFF_2 is copied into the parity flag where it can be tested or stored.

2-62. A second method of restoring the status of IFF_1 is thru the execution of a Return from Non-Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non-maskable interrupt service routine is complete, the contents of IFF_2 are

now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non-maskable interrupt will be restored automatically.

Table 2-1. Interrupt Enable/Disable Flip Flops

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A,I	.	.	IFF_2 --> Parity flag
LD A,R	.	.	IFF_2 --> Parity flag
Accept NMI	0	.	
RETN	IFF_2	.	IFF_2 --> IFF_1
RETI	.	.	
Accept INT	0	0	"." indicates no change

2-63. LOAD AND EXCHANGE INSTRUCTIONS. These instructions move data to and from registers, such as load B from D, load C from memory, store HL into memory, push IX into stack, and exchange AF with AF'.

2-64. BLOCK TRANSFER AND SEARCH INSTRUCTIONS. This group includes several useful instructions.

2-65. The load and increment instruction moves one byte of data from memory pointed to by HL to another memory location pointed to by DE. Both register pairs are automatically incremented and the byte counter (BC) is decremented. This instruction is extremely valuable in moving blocks of data.

2-66. Another instruction repeats the load and increment instruction automatically until the byte counter reaches zero. Thus, in one instruction, a block of data, up to 64K bytes in length, can be moved anywhere in memory.

2-67. The compare and increment instruction compares the contents of the accumulator with that of memory pointed to by HL. The appropriate flag bits are set, HL is automatically incremented, and the byte counter is decremented.

2-68. The compare, increment, and repeat instruction repeats the above instruction until either a match is found or the counter reaches zero.

2-69. ARITHMETIC AND LOGICAL INSTRUCTIONS. These instructions include all the adds

and subtracts, increments,compares, exclusive-ors, etc. The Z80 features the indexed addressing mode and double precision add with carry and subtract with carry.

2-70. ROTATE AND SHIFT INSTRUCTIONS. The Z80 included four rotate accumulator instructions and logical shifts and arithmetic shifts. There are also two rotate digit instructions which are applicable to BCD arithmetic. With these a digit (4 bits) can be rotated with two digits in a memory location.

2-71. BIT MANIPULATION INSTRUCTIONS. There are three basic bit manipulation operations: test bit, set bit, and reset bit.

2-72. JUMP, CALL, AND RETURN. The Z80 has numerous conditional and unconditional jumps, calls, and returns. In addition, the Z80 has several jump relative instructions using relative addressing.

2-73. INPUT/OUTPUT INSTRUCTIONS.

2-74. The Z80 allows for a standard common I/O routine for all devices by including I/O instructions that use the C register to contain the IO device address. Therefore one I/O routine can be used with device address placed into register C before entering the routine. Also instead of being restricted to inputting or outputting to and from the accumulator only, any register can be used.

2-75. The Z80 has eight block tranfer I/O instructions which are similar to the memory block transfer instructions. HL is the memory pointer, C is the device pointer, and B is the byte counter. Therefore, an I/O block transfer can handle up to 256 bytes. Essentially, these commands are a processor implementation of direct memory access (DMA), invoked by a software sequence.

2-76. MISCELLANEOUS FEATURES. The Z80 instruction set also includes a no-operation instruction.

2-77. **Z80 ASSEMBLY LANGUAGE SYNTAX.**

2-78. INTRODUCTION.

2-79. The assembly language of the Z80 is designed to minimize the number of different opcodes corresponding to the set of basic machine operations and to provide for a consistent description of instruction operands. The nomenclature has been defined with special emphasis on mnemonic value and readability.

2-80. An assembly language program, or source program, consists of statements in a sequence which defines the user's program. The statements consist of:

1. labels,
2. opcodes or pseudo-ops
3. operands, and
4. comments.

2-81. Certain rules define how assembly language statements are to appear. A statement has four separate and distinct parts or fields.

```
      LABEL      OPCODE      OPERANDS      COMMENT

e.g.: LOOP:      LD          HL, VALUE     ;GET VALUE
```

2-82. The first field is the LABEL field. The label is a name used to reference the program counter, another label, or a constant.

2-83. The second field is the OPCODE field. It specifies the operation to be performed. There are 74 Z80 opcodes and several pseudo-ops that are standard for the Z80. The standard pseudo-ops are described in Appendix B.

2-84. The third field is the OPERAND field. It provides address or data information for the OPCODE field. There may be zero or more operands in the operand field depending on the requirements of the opcode field.

2-85. The fourth field is the COMMENT field. It is used to document a program. The comment field may appear in a statement without the other fields. Comments are ignored by an assembler, but they are printed in the assembly listing.

2-86. Each of the above parts, or fields, must be separated from each other by one or more commas, tabs, or blanks. If more than one operand appears, they must be separated from each other by one or more commas.

2-87. LABELS

2-88. A label is a symbol representing up to 16 bits of information and is used to specify an address or data. By using labels effectively, the user can write assembly language programs more rapidly and make fewer errors.

2-89. A label is composed of one or more characters. If more than 6 characters ar used for the label, only the first 6 will be recognized by a standard assembler The first character of a label must not be a number (0-9) or a restricted character The remaining characters connot include a restricted character. The restricted char acters are:

Control characters (0-2FH,7FH)
Space
' ( ) * = , - . /
: ; < = >

Note that single dollar sign ($) is reserved to represent the program counter.

2-90. A label can start in any column if followed by a colon (:). It does no require a colon if started in column one.

2-91. OPCODES. The bulk of this manual describes the Z80 opcodes. Opcodes are to 4 characters long and describe Z80 instructions.

2-92. STANDARD OPERANDS. There may be zero or more operands present in a statemen depending upon the opcode used. An operand which appears in a statement may tak one of the following forms.

1. A generic operand, such as the letter A, which stands for the accumulator
2. A constant. The constant must be in the range 0 through 0FFFFH. It can be i the following forms:

Decimal - Any number may be denoted as decimal by following it with th letter 'D'. E.g., 35,249D. However, the assembler will conside any number which is undesignated as decimal.
Hexadecimal - must begin with a number (0-9) and end with the letter 'H' E.g., 0AF1H
Octal - must end with the letter 'Q' or 'O'. E.g., 377Q, 277O
Binary - must end with the letter 'B. e.g., 0110111B
ASCII - letters enclosed in quote marks will be converted to their ASCI equivalent value. E.g., 'A' = 41H

3. A label which appears elsewhere in the program. Note that labels canno be defined by labels which have not yet appeared in the user program for 2-pas assemblers.

E.g.:

```
L EQU H
H EQU I
I EQU 7  IS NOT ALLOWED.
I EQU 7
H EQU I
L EQU H  IS ALLOWED.
```

4. The symbol $ is used to represent the value of the program counter of the current instruction.

5. Expressions. Expression evaluation capability is a function of the features of a particular assembler. In general, arithmetic and logical expressions are allowed, and parentheses may be used to assure correct evaluation.

2-93. OPERAND NOTATION. The following notation is used in the assembly language:

1) r specifies any one of the following registers: A,B,C,D,E,H,L.
2) (HL) specifies the contents of memory at the location addressed by the contents of the register pair HL.
3) n specifies a one-byte expression in the range 0 to 255. nn specifies a two byte expression in the range 0 to 65535.
4) d specifies a one-byte expression in the range (-128,127).
5) (nn) specifies the contents of memory at the location addressed by the two-byte expression nn.
6) b specifies an expression in the range (0.7).
7) e specifies a one-byte expression in the range (-126,129).
8) cc specifies the state of the flags for conditional JR and JP instructions.
9) qq specifies any one of the register pairs BC, DE, HL or AF.
10) ss specifies any one of the following register pairs: BC, DE, HL, SP.
11) pp specifies any one of following register pairs: BC,DE, IX, SP.
12) rr specifies any one fo the following register pairs: BC, DE, IY, SP.
13) Specifies any of r,n, (HL, (IX+d), (IY+d).
14) dd specifies any one of the following register pairs: BC, DE, HL, SP.
15) m specifies any of r, (HL), (IX+d), (IY+d).

NOTE: The enclosing of an expression wholly in parentheses indicates a memory address. The contents of the memory address equivalent to the expression value will be used as the operand value.

2-94. In doing relative addressing, the current value of the program counter must be subtracted from the label if a branch is to be made to that label address. E.g.:

```
JR NC,LOOP-$
       ...will jump relative to 'LOOP'
```

2-95. COMMENTS. A comment is defined as any string of characters following a semicolon. Comments are ignored by an assembler, but they are printed on the assembly listing. Comments can begin in any column:

```
;
; this is a comment
;
```

2-96. UPPER/LOWER CASE.

NOTE: MOSTEK assemblers allow the user of lower case letters for labels and comments.

2-97. OPCODES - DETAILED DESCRIPTIONS.

2-98. INTRODUCTION. This section describes each Z80 opcode (instruction) in detail. The opcodes are presented in alphabetical order, one per page. Each instruction is introduced by its mnemonic opcode and symbolic operands. Then follows a brief description, operation, valid operand combinations, machine code, detailed description, condition bits affected, and one or more examples.

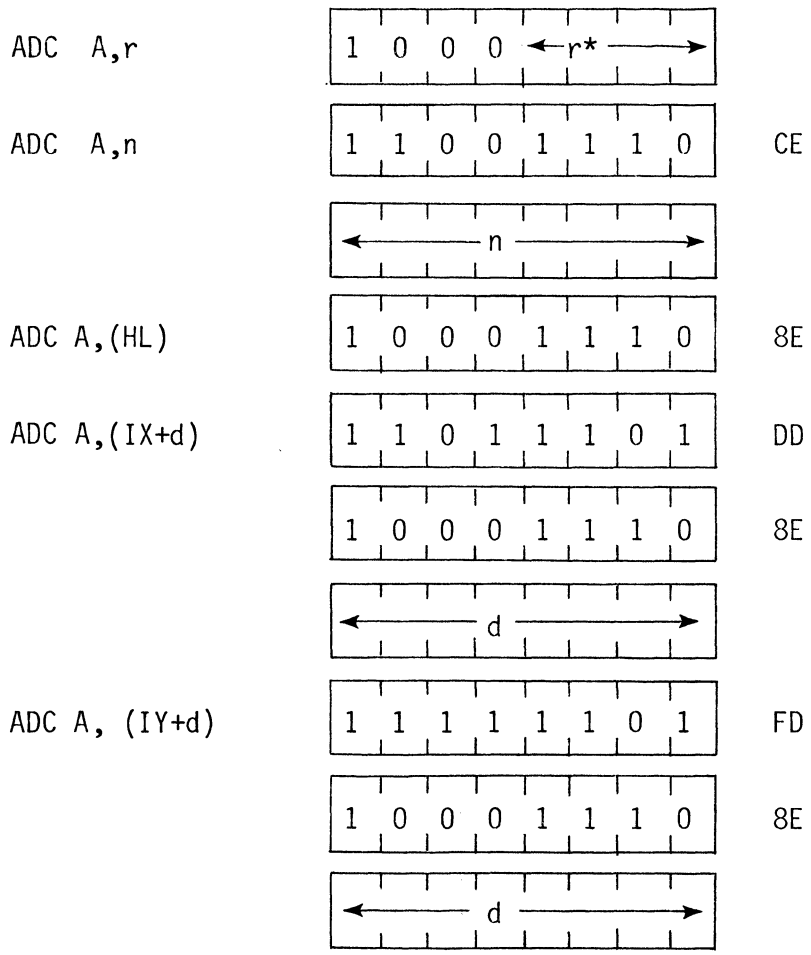
ADC A, s

Operation. A ← A + s + CY

Format:

Opcode	Operands
ADC	A,s

The s operand is any of r,n, (HL),IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:



r identifies registers B,C,D,E,H,L or A assembled as follows in the object code ield 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 CYCLES	T STATES
ADC A,r	1	4
ADC A,n	2	7(4,3)
ADC A,(HL)	2	7(4,3)
ADC A, (IX+d)	5	19(4,4,3,5,3)
ADC A,(IY+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Set if carry from Bit 3; reset otherwise

P/V: Set if overflow: reset otherwise

N: Reset

C: Set if carry from Bit 7; reset otherwise

Example:

If the Accumulator contains 16H, the Carry Flag is set, the HL register pair contains 6666H, and address 6666H contains 10H, after the execution of

```
ADC A,(HL)
```

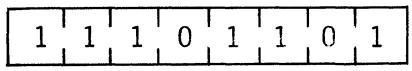
the Accumulator will contain 27H.

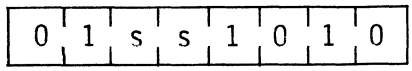
ADC HL, ss

Operation: HL←HL+ss+CY

Format:

Opcode	Operands
ADC	HL,ss





Description:

The contents of register pair ss (any of register pairs DC, 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: 4 T STATES: 15(4,4,4,3)

Condition Bits affected:

S. Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Set if carry out of Bit 11; reset otherwise

P/V: Set if overflow; reset otherwise

N: Reset

C: 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, after the execution of

```
ADC HL, BC
```

the contents of HL will be 765AH.

ADD A, (HL)

Operation: A← A + (HL)

Format:

Opcode	Operands	
ADD	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: 2 T STATES: 7(4,3)

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are AOH, and the content of the register pair HL is 2323H, and memory location 2323H contains byte 08H after the execution of

ADD A, (HL)

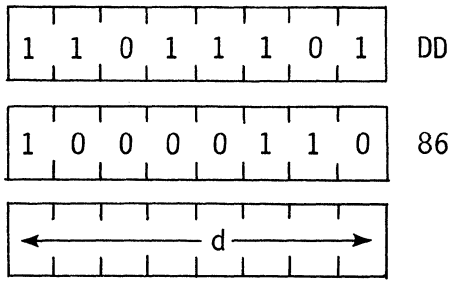
the Accumulator will contain A8H.

ADD A, (IX+d)

Operation: A← A + (IX+d)

Format:

Opcode	Operands
ADD	A,(IX+d)



Description:

The contents of the Index Register (register pair IX) is added to a displacemer d to point to an address in memory. The contents of this address is then added t the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected:

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

Example:

If the Accumulator contents are 11H, the Index Register IX contains 1000H, and if th content of memory location

1005H is 22H, after the execution of

ADD A, (IX+5H)

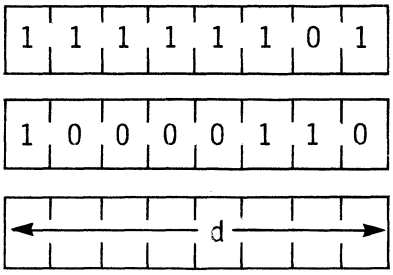
the contents of the Accumulator will be 33H.

ADD A, (IY+d)

Operation: A ← A+(IY+d)

Format:

Opcode	Operands
ADD	A,(IY+d)



Description:

The contents of the Index Register (register pair IY) are added to a displacem
d to point to an address in memory. The contents of this address is then add
to the contents of the Accumulator and the result is stored in the Accumulato

M CYCLES: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected:

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

Example:

If the Accumulator contents are 11H, the Index Register pair IY contains 1000H
and if the content of memory

location 1005H is 22H, after the execution of

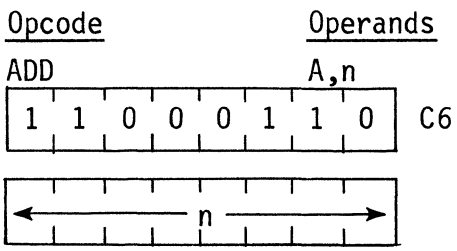
```
ADD A,(IY+5H)
```

the contents of the Accumulator will be 33H.

ADD A, n

Operation: A ← A + n

Format.



Description:

The integer n is added to the contents of the Accumulator and the results are store in the Accumulator.

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are 23H, after the execution of

```
ADD A,33H
```

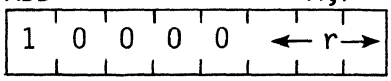
the contents of the Accumulator will be 56H.

ADD A, r

Operation: A ← A + r

Format:

Opcode	Operands
ADD	A,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: 1 T STATES: 4

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero: reset otherwise

H: Set if carry from Bit 3; reset otherwise

P/V: Set if overflow; reset otherwise

N: Reset

C: Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 44H, and the contents of register C are 11H after the execution of

```
ADD A,C
```

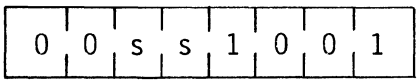
the contents of the Accumulator will be 55H.

ADD HL, ss

Operation: HL ← HL + ss

Format:

Opcode	Operands
ADD	HL,ss



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: 3 T STATES: 11(4,4,3)

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set if carry out of Bit 11; reset otherwise
P/V: Not affected
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If register pair HL contains the integer 4242H and register pair DE contains 1111H, after the execution of

```
ADD HL,DE
```

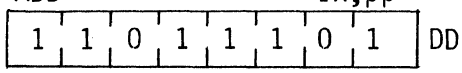
the HL register will contain 5353H

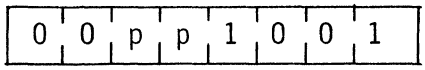
ADD IX, pp

Operation: IX ← IX + pp

Format:

Opcode	Operands
ADD	IX,pp





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: 4 T STATES: 15(4,4,4,3)

Condition Bits Affected:

S; Not affected
Z: Not affected
H: Set if carry out of Bit 11; reset otherwise
P/V: Not affected
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IX are 3333H and the contents of register pair BC are 5555H, after the execution of

```
ADD IX,BC
```

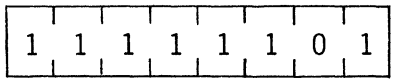
the contents of IX will be 8888H.

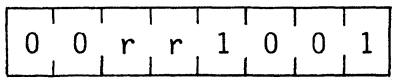
ADD IY, rr

Operation: IY ← IY + rr

Format:

Opcode	Operands
ADD	IY,rr





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: 4 T STATES: 15(4,4,4,3)

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set if carry out of Bit 11; reset otherwise
P/V: Not affected
N: Reset
C: Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IY are 3333H and the contents of register pair are 5555H, after the execution of

```
ADD IY,BC
```

the contents of IY will be 8888H.

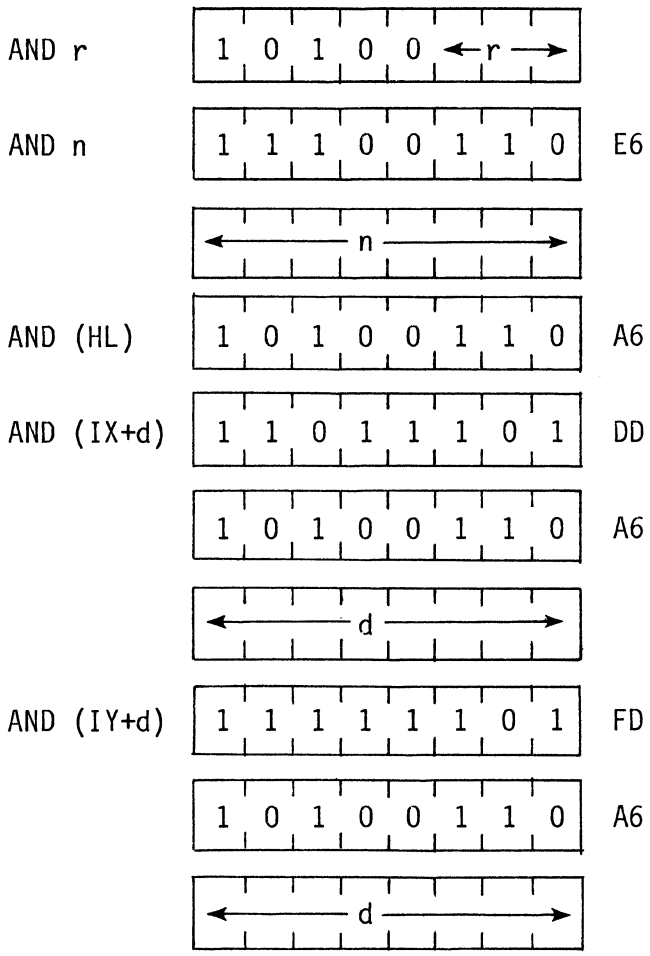
AND s

Operation: A ← A s

Format:

Opcode	Operands
AND	s

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



*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:

A logical AND operation, bit by bit, 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
AND r	1	4
AND n	2	7 (4,3)
AND (HL)	2	7 (4,3)
AND (IX+d)	5	19(4,4,3,5,3)
AND (IY+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

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

Example: If the B register contains 7BH (01111011) and the Accumulator contains C3H (11000011) after the execution of

AND B

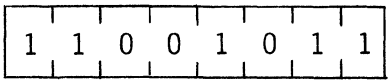
the Accumulator will contain 43H (01000011).

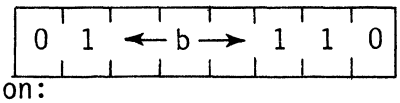
BIT b, (HL)

Operation: $Z \leftarrow (HL)_b$

Format:

Opcode	Operands
BIT	b,(HL)





Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory address pointed to by the HL register pair. 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: 3 T STATES: 12(4,4,4)

Condition Bits Affected:

S: Unknown
Z: Set if specified bit is 0; reset otherwise
H: Set
P/V: Unknown
H: Reset
C: Not affected

Example:

If the HL register pair contains 4444H, and bit 4 in the memory location 4444H contains 1, after the execution of

```
BIT 4, (HL)
```

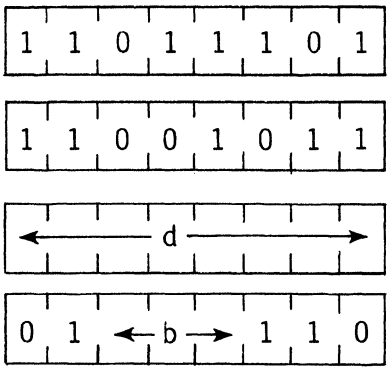
the Z flag in the F register will contain 0, and bit 4 in memory location 4444H will still contain 1. (Bit 0 in memory location 4444H is the least significant bit.)

BIT b, (IX+d)

Operation: $Z \leftarrow \overline{(IX+d)_b}$

Format:

Opcode	Operands
BIT	b,(IX+d)



Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents of register pair IX (Index Register IX) and the two's complement displacement integer d. 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: 5 T STATES: 20(4,4,3,5,4)

Condition Bits Affected:

S: Unknown
Z: Set if specified bit is 0; reset otherwise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If the contents of Index Register IX are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

```
BIT 6, (IX+4H)
```

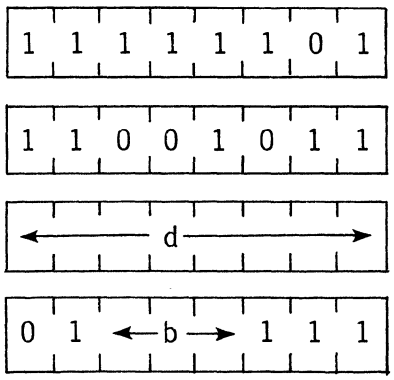
the Z flag in the F register will contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit).

BIT b, (IY+d)

Operation: $Z \leftarrow \overline{(IY+d)_b}$

Format:

Opcode	Operands
BIT	b,(IY+d)



Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents of register pair IY (Index Register IY) and the two's complement displacement integer d. 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: 5 T STATES: 20(4,4,3,5,4)

Condition Bits Affected:

S: Unknown
Z: Set if specified bit is 0; reset other wise
H: Set
P/V: Unknown
N: Reset
C: Not affected

Example:

If the contents of Index Register are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

```
BIT 6, (IY+4H)
```

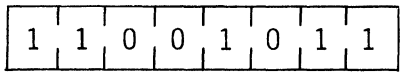
the Z flag in the F register will contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit).

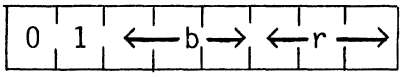
BIT b, r

Operation: $Z \leftarrow r_b$

Format:

Opcode	Operands
BIT	b,r





Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the r - register. 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: 2 T STATES: 8 (4,4)

Condition Bits Affected:

S: Unknown
Z: Set if specified bit is 0; reset otherwise
H: Set
P/V: Unknown
H: Reset
C: Not affected

Example:

If bit 4 in the B-register contains 1, after the execution of

```
BIT 4, B
```

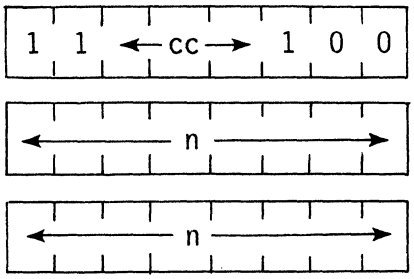
the Z flag in the F register will contain 0, and bit 4 in the B register will still contain 1. (Bit 0 in the B-register is the least significant bit.)

CALL cc, nn

Operation: IF cc TRUE: $(SP-1) \leftarrow PC_H$
$(SP-2) \leftarrow PC_L$, $PC \leftarrow nn$

Format:

Opcode	Operands
CALL	cc,nn



Note: The first of the two n operands in the assembled object code above is the least significant byte of the two-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 into PC to point to the address in memory where the first opcode 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 into 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 into the memory address now pointed to by SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of the stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before the push is executed. Condition cc is programmed

as one of eight status which corresponds to condition bits in the Flag Registe (register F). These eight status are defined in the table below, which also specifie 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: 5 T STATES: 17(4,3,4,3,3)

If cc is false:

M CYCLES: 3 T STATES: 10(4,3,3)

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
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction D43521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

CALL NC,2135H

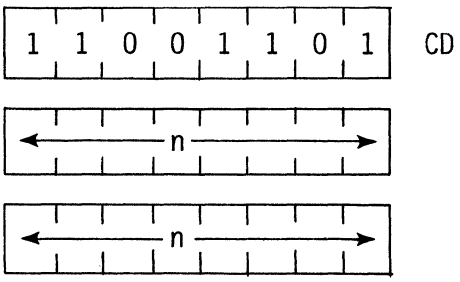
After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

CALL nn

Operation: (SP-1) ← PC_H, (SP-2) ←PC_L, PC ←nn

Format:

Opcode	Operands
CALL	nn



Note: The first of the two n operands in the assembled object code above is the least significant byte of a two-byte memory address.

Description:

After pushing the current contents of the Program Counter (PC) onto the top of the external memory stack, the operands nn are loaded into PC to point to the address in memory where the first opcode 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 into 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 into the memory address now pointed to by the SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before the push is executed.

M CYCLES: 5 T STATES: 17(4,3,4,3,3,)

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:

Location	Contents
1A47H	CDH
1A48H	35H
1A49H	21H

Then if an instruction fetched sequence begins, the three-byte instruction CD3521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

```
CALL 2135H
```

After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H willbe 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

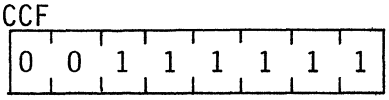
CCF

Operation: CY ← CY

Format:

Opcode

CCF



Description:

The c flag in the F register is inverted.

M CYCLES: 1 T STATES: 4

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Previous carry will be copied
P/V: Not affected
N: Reset
C: Set if CY was 0 before operation; reset otherwise

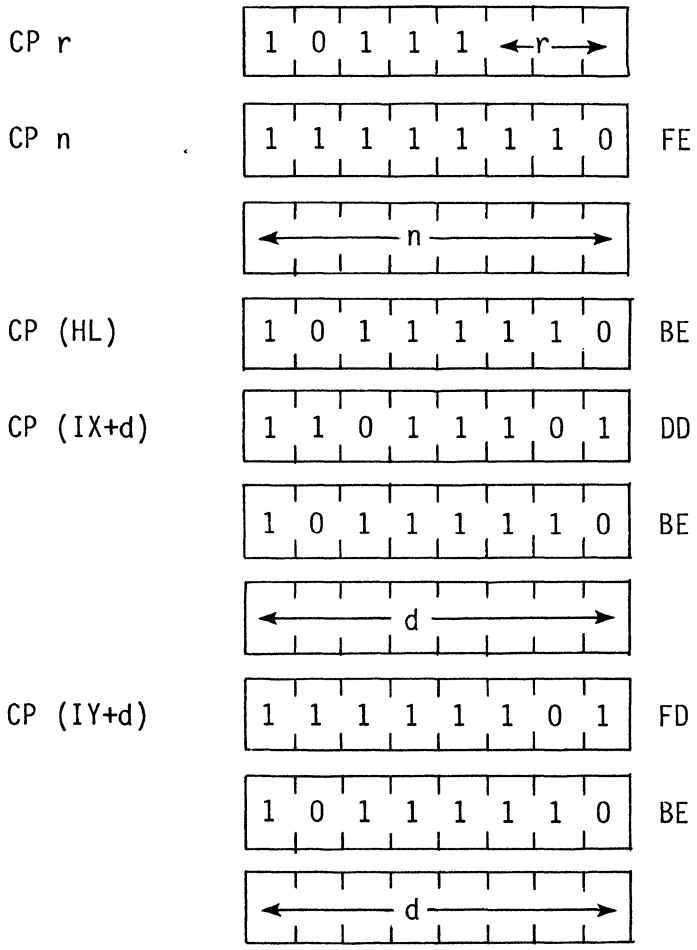
CP s

Operation: A - s

Format:

Opcode	Operands
CP	s

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



*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 s operand are compared with the contents of the Accumulator. there is a true compare, a flag is set.

INSTRUCTION	M CYCLES	T STATES
CP r	1	4
CP n	2	7(4,3)
CP (HL)	2	7(4,3)
CP (IX+d)	5	19(4,4,3,5,3)
CP (IY+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if there is a borrow from Bit 4; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if there is a borrow; reset otherwise

Example:
If the Accumulator contains 63H, the HL register pair contains 6000H and memor location 6000H contains 60H, the instruction

CP (HL)

will result in the P/V flag in the F register being reset.

CPD

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

Format:

Opcode	Operands
CPD	
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 the Byte Counter (register pair BC) are decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if A=(HL);
reset otherwise

H: Set if there is a borrow from
Bit 4; reset otherwise

P/V: Set if BC-1≠0;
reset otherwise

N: Set

C: 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, then after the execution of

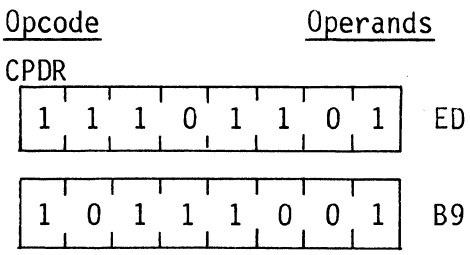
CPD

the Byte Counter will contain 0000H, the HL register pair will contain 1110H, the flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

CPDR

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

Format:



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 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes, if no match is found. Also, interrupts will be recognized and two refresh cycles will be executed after each data comparison.

For BC≠0 and A≠(HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5)

For BC=0 or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if A=(HL);
reset otherwise
H: Set if there is a borrow from
Bit 4; reset otherwise
P/V: Set if BC-1≠0;
reset otherwise
N: Set
C: Not affected

Example:

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

(1118H): 52H
(1117H): 00H
(1116H): F3H

then after the execution of

CPDR

the contents of register pair HL will be 1115H, the contents of the Byte Counte will be 0004H, the P/V flag in the F register will be set, and the Z flag in the register will be set.

CPI

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

Format:

Opcode	Operands
CPI	
1 1 1 0 1 1 0 1	ED
1 0 1 0 0 0 0 1	A1

Description:

The contents of the memory location addressed by the HL register pair are 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: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if A=(HL);
reset otherwise

H: Set if there is a borrow from
Bit 4; reset otherwise

P/V: Set if BC-1≠0;
reset otherwise

N: Set

C: 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, then after the execution of

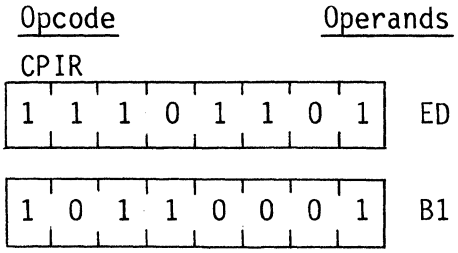
CPI

the Byte Counter will contain 0000H, the HL register pair will contain 1112H, th Z flag in the F register will be set, and the P/V flag in the F register will b reset. There will be no effect on the contents of the Accumulator or address 1111H

CPIR

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

Format:



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 is incremented and the Byte Counter (register pair BC) is 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 2 and the instruction is repeated. Note that if BC is set to zero before instruction execution, the instruction will loop through 64K bytes, if no match if found. Also, interrupts will be recognized and two refresh cycles will be executed after each data comparison.

For BC≠0 and A≠(HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5)

For BC=0 or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise
Z: Set if A=(HL);
reset otherwise
H: Set if there is a borrow from
Bit 4; reset otherwise
P/V: Set if BC-1≠0;
reset otherwise
N: Set
C: 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) : 52H
(1112H) : 00H
(1113H) : F3H

then after the execution of

CPIR

the contents of register pair HL will be 1114H, the contents of the Byte Counter will be 0004H, the P/V flag in the F register will be set and the Z flag in the F register will be set.

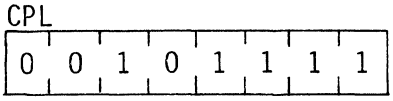
CPL

Operation: A ←A

Format:

Opcode

CPL



2F

Description:

Contents of the Accumulator (register A) are inverted (1's complement).

M CYCLES: 1 T STATES: 4

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are 1011 0100, after the execution of

CPL

the Accumulator contents will be 0100 1011.

DAA

Operation: —

Format:

Opcode

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 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	0-9	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-F	1	6-F	9A	1

M CYCLES: 1 T STATES: 4

Condition Bits Affected:

S: Set if most significant bit of Acc. is 1 after operation; reset otherwise

Z: Set if Acc. is zero after operation; reset otherwise

H: See instruction

P/V: Set if Acc. is even parity after operation; reset otherwise

N: Not affected

C: See instruction

Example:

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

```
 15
+27
 42
```

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

```
 0001  0101
+0010  0111
 0011  1100 =3C
```

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

```
 0011  1100
+0000  0110
 0100  0010 =42
```

DEC IX

Operation: IX ← IX -1

Format:

Opcode	Operands
DEC	IX

1	1	0	1	1	1	0	1	DD
0	0	1	0	1	0	1	1	2B

Description:

The contents of the Index Register IX are decremented.

M Cycles: 2 T STATES: 10 (4,6)

Condition Bits Affected: None

Example:

If the contents of the Index Register IX are 7649H, after execution of

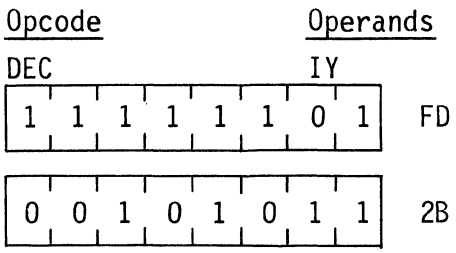
```
DEC IX
```

the contents of Index Register IX will be 7648H.

DEC IY

Operation: IY ← IY -1

Format:



Description:

The contents of the Index Register IY are decremented.

M CYCLES: 2 T STATES: 10(4,6)

Condition Bits Affected: None

Example:

If the contents of the Index Register IY are 7649H, then after the execution of :

```
DEC IY
```

the contents of Index Register IY will be 7648H.

DEC m

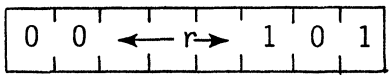
Operation: m ← m-1

Format:

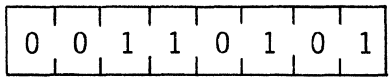
Opcode	Operands
DEC	m

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

DEC r

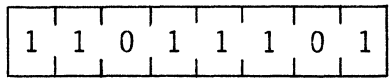


DEC (HL)

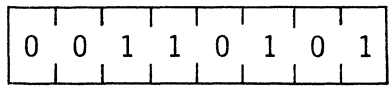


35

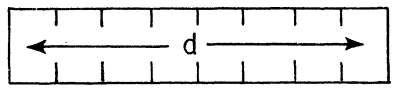
DEC (IX+d)



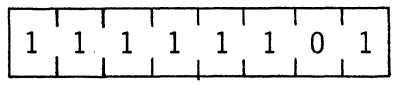
DD



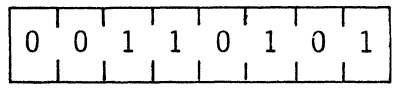
35



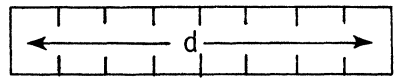
DEC (IY+d)



FD



35



*r identifies register B,C,D,E,H,L or A assembled as follows in th 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
DEC R	1	4
DEC (HL)	3	11(4,4,3)
DEC (IX+d)	6	23(4,4,3,5,4,3)
DEC (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero: reset otherwise

H: Set if there is a borrow from Bit 4, reset otherwise

P/V: Set if m was 80H before operation; reset otherwise

N: Set

C: Not affected

Example:

If the D register contains byte 2AH, after the execution of

```
DEC D
```

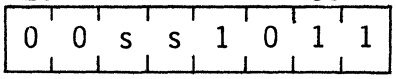
register D will contain 29H.

DEC ss

Operation: ss ← ss -1

Format:

Opcode	Operands
DEC	ss



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.

Pair	ss
BC	00
DE	01
HL	10
SP	11

M CYCLES: 1 T STATES: 6

Condition Bits Affected: None

Example:

If register pair HL contains 1001H, after the execution of

```
DEC HL
```

the contents of HL will be 1000H.

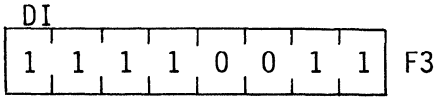
DI

Operation: IFF ← Ø

Format:

Opcode

DI



Description:

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

M CYCLE: 1 T STATES: 4

Condition Bits Affected: none

Example:

When the CPU executes the instruction

DI

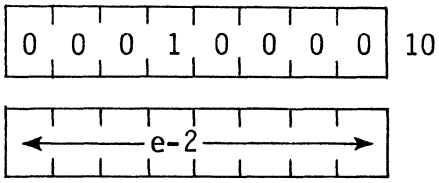
the maskable interrupt is disabled. The CPU will not respond to an interrupt request (INT) signal.

DJNZ, e

Operation: —

Format:

Opcode	Operand
DJNZ	e



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 opcode 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 to be executed is taken from the location following this instruction

If B≠0:

M CYCLES: 3 T STATES: 13(5,3,5)

IF B=0:

M CYCLES: 2 T STATES: 8(5,3)

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 move the bytes until it finds a CR, or until it has moved 80 bytes, whichever occurs first.

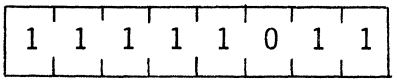
```
        LD              B,80            ;Set up counter
        LD              HL,Inbuf        ;Set up pointers
        LD              DE,Outbuf

LOOP:   LD              A,(HL)          ;Get next byte from
                                        ;input buffer
        LD              (DE),A          ;Store in output buffer
        CP              0DH             ;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:
```

EI

Operation: IFF ← 1

Format:

Opcode
EI



FB

Description:

EI enables the maskable interrupt by setting the interrupt enable flip-flops(IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M CYCLES: 1 T STATES: 4

Condition Bits Affected: None

Example:

When the CPU executes instruction

EI

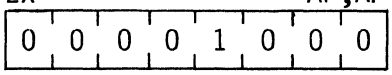
the maskable interrupt is enabled. The CPU will now respond to an Interrupt Request (INT) signal.

EX AF, AF'

Operation: AF ⟷ AF'

Format:

Opcode	Operands
EX	AF,AF



08

Description:

The two-byte contents of the register pairs AF and AF' are exchanged. (Note: register pair AF consists of registers A' and F'.

M CYCLES: 1 T STATES: 4

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, after the instruction

```
EX AF, AF'
```

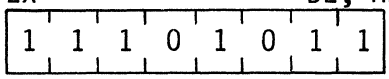
the contents of AF will be 5944H, and the contents of AF' will be 9900H.

EX DE, HL

Operation: DE ↔ HL

Format:

Opcode	Operands
EX	DE, HL



EB

Description:

The two-byte contents of register pairs DE and HL are exchanged.

M CYCLES: 1 T STATES: 4

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, after the instruction

```
EX DE,HL
```

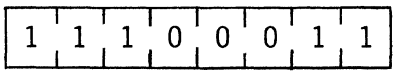
the content of register pair DE will be 499AH and the content of register pair HL 2822H.

EX (SP), HL

Operation: H ⟷ (SP+1), L ⟷ (SP)

Format:

Opcode	Operands
EX	(SP), HL



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: 5 T STATES: 19(4,3,4,3,5)

Condition Bits Affected: None

Example:

If the HL register pair contains 7012H, the SP register pair contains 8856H, the memory location 8856H contains the byte 11H, and the memory location 8857H contains the byte 22H, then the instruction

EX (SP),HL

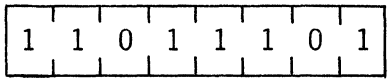
will result in the HL register pair containing number 2211H, memory location 8856H containing the byte 12H, the memory location 8857H containing the byte 70H and the Stack Pointer containing 8856H.

EX (SP), IX

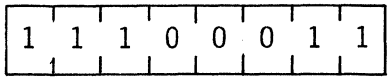
Operation: $IX_H \leftrightarrow (SP+1)$, $IX_L \leftrightarrow (SP)$

Format:

Opcode	Operands
EX	(SP),IX



DD



E3

Description:

The low order byte in Index Register IX is exchanged with 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: 6 T STATES: 23(4,4,3,4,3,5)

Condition Bits Affected: None

Example:

If the Index Register IX contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 80H, and the memory location 0101H contains byte 48H, then the instruction

```
EX (SP),IX
```

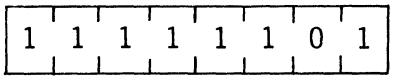
will result in the IX register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H and the Stack Pointer containing 0100H.

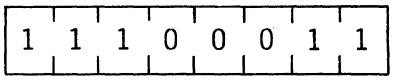
EX (SP), IY

Operation: $IY_H \leftrightarrow (SP+1)$, $IY_L \leftrightarrow (SP)$

Format:

Opcode	Operands
EX	(SP),IY





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: 6 T STATES: 23(4,4,3,4,3,5)

Condition Bits Affected: None

Example:

If the Index Register IY contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

```
EX (SP),IY
```

will result in the IY register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H, and the Stack Pointer containing 0100H.

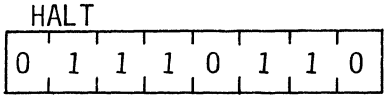
HALT

Operation: CPU halted.

Format:

Opcode

HALT



76

Description:

HALT stops execution of instructions in the CPU. Interrupts can be accepted when the CPU is in the halt state.

M CYCLES: 1 T STATES: 4

Condition Bits Affected: none

Example:

When the CPU executes the instruction

HALT

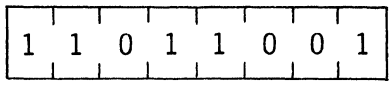
no further execution of instructions will occur unless an interrupt occurs and is accepted.

EXX

Operation: BC) ↔ (BC),(DE)↔(DE),(HL) ↔(HL)

Format:

Opcode	Operands
EXX	



D9

Description:

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

M CYCLES: 1 T STATES: 4

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, after the instruction

EXX

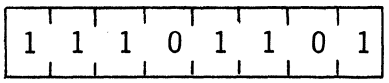
the contents of the register pairs will be as follows: BC: 0988H; DE: 9300H; HL: 00E7H; BC': 445AH; DE': 3DA2H; and HL': 8859H.

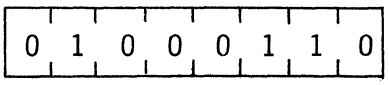
IM 0

Operation: ———

Format:

Opcode	Operands
IM	0





Description:

The IM 0 instruction sets interrupt mode 0. In this mode the interrupting device can insert any instruction on the data bus and allow the CPU to execute it.

M CYCLES: 2 T STATES: 8(4,4)

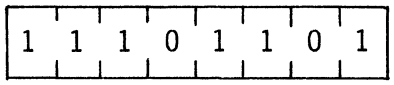
Condition Bits Affected: None

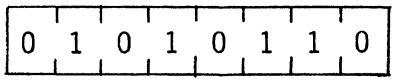
IM 1

)peration: ——

Format:

OPCODE	Operands
IM	1





)escription:

The IM1 instruction sets interrupt mode 1. In this mode the processor will respond to an interrupt by executing a restart to location 0038H.

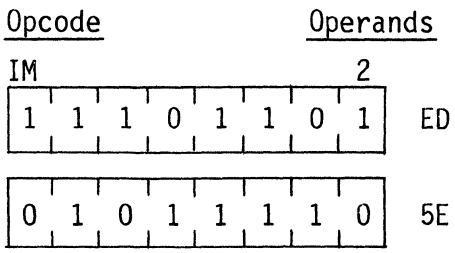
M CYCLES: 2 T STATES: 8(4,4)

Condition Bits Affected: None

IM 2

Operation: ———

Format:



Description:

The IM 2 instruction sets interrupt mode 2. This mode allows an indirect call to any location in memory. With this mode the CPU forms a 16-bit memory address. The upper eight bits are the contents of the Interrupt Vector Register I and the lower eight bits are supplied by the interrupting device.

M CYCLES: 2 T STATES: 8(4,4)

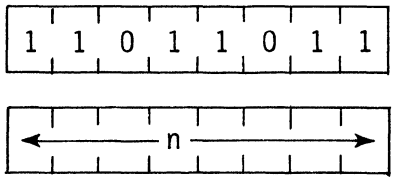
Condition Bits Affected: None

IN A, (n)

Operation: A ←(n)

Format:

Opcode	Operands
IN	A,(n)



Description:

The operand n is placed on the botton 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 into the Accumulator (register A) in the CPU.

M CYCLES: 3 T STATES: 11(4,3,4)

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H and the byte 7BH is available at the peripheral device mapped to I/O port address 01H, then after the execution of

```
IN A,(01H)
```

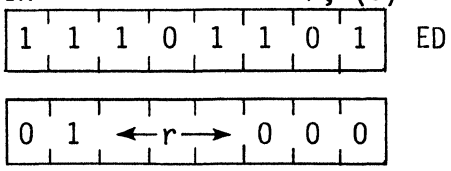
the Accumulator will contain 7BH.

IN r, (C)

Operation: r ← (C)

Format:

Opcode	Operands
IN	r, (C)



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 into register r in the CPU. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each. The flags will be affected, checking the input data.

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

M CYCLES: 3 T STATES: 12(4,4,4)

Condition Bits Affected:

S: Set if input data is negative;
reset otherwise

Z: Set if input data is zero;
reset otherwise

H: Reset

P/V: Set if parity is even;
reset otherwise

N: Reset

C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then after the execution of

```
IN D,(C)
```

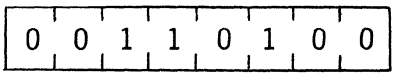
register D will contain 7BH, and register B will contain 10H.

INC (HL)

Operation: (HL) ⟵ (HL)+1

Format:

Opcode	Operands
INC	(HL)



34

Description:

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

M CYCLES: 3 T STATES: 11(4,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if (HL) was 7FH before operation; reset otherwise
N: Reset
C: Not Affected

Example:

If the contents of the HL register pair are 3434H, and the contents of address 3434 are 82H, after the execution of

INC (HL)

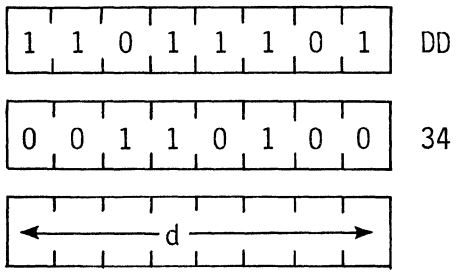
memory location 3434H will contain 83H.

INC (IX+d)

Operation: (IX+d) ⟵ (IX+d)+1

Format:

Opcode	Operands
INC	(IX+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: 6 T STATES: 23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Set if carry from Bit 3; reset otherwise

P/V: Set if (IX+d) was 7FH before operation; reset otherwise

N: Reset

C: Not affected

Example:

If the contents of the Index Register pair IX are 2020H, and the memory location 2030H contains byte 34H, after the execution of

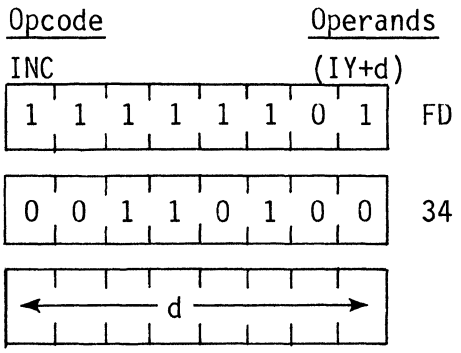
```
INC (IX+10H)
```

the contents of memory location 2030H will be 35H.

INC (IY+d)

Operation: (IY+d) ←(IY+d)+1

Format:



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: 6 T STATES: 23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Set if carry from Bit 3; reset otherwise

P/V: Set if (IY+d) was 7FH before operation; reset otherwise

N: Reset

C: Not Affected

Example:

If the contents of the Index Register pair IY are 2020H, and the memory location 2030H contain byte 34H, after the execution of

```
INC (IY+10H)
```

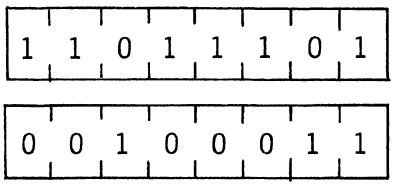
the contents of memory location 2030H will be 35H.

INC IX

Operation: IX ←—IX +1

Format:

Opcode	Operands
INC	IX



Description:

The contents of the Index Register IX are incremented.

M CYCLES: 2 T STATES: 10(4,6)

Condition Bits Affected: None

Example:

If the Index Register IX contains the integer 3300H, after the execution of

```
INC IX
```

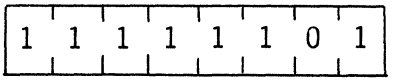
the contents of Index Register IX will be 3301H.

INC IY

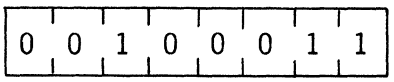
Operation: IY ← IY + 1

Format:

Opcode	Operands
INC	IY



FD



23

Description:

The contents of the Index Register IY are incremented.

M CYCLES: 2 T STATES: 10(4,6)

Condition Bits Affected: None

Example:

If the contents of the Index Register are 2977H, after the execution of

```
INC IY
```

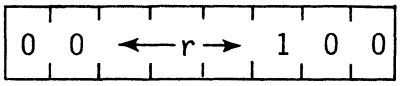
the contents of Index Register IY will be 2978H.

INC r

Operation: r ←r + 1

Format:

Opcode	Operands
INC	r



Description:
Register r is incremented. 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: 1 T STATES: 4

Conditions Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if carry from Bit 3; reset otherwise
P/V: Set if r was 7FH before operation; reset other wise
N: Reset
C: Not affected

Example:

If the contents of register D are 28H, after the execution of

```
INC D
```

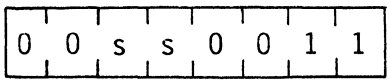
the contents of register D will be 29H.

INC ss

Operation: ss ⟵ ss + 1

Format:

Opcodes	Operands
INC	ss



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: 1 T STATES: 6

Condition Bits Affected: None

Example:

If the register pair contains 1000H, after the execution of

```
INC HL
```

HL will contain 1001H.

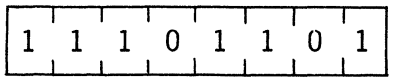
IND

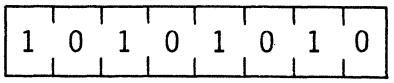
Operation: (HL) ← (C), B ← B-1, HL ← HL-1

Format:

Opcode

IND





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 written into the corresponding location of memory. Finally the byte counter and register pair HL are decremented.

M CYCLES: 4 T STATES 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set if B-1=0;
reset other wise
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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 byte 7BH is available at the

peripheral device mapped to I/O port address 07H, then after the execution of

IND

memory location 1000H will contain 7BH, the HL register pair will contain 0FFFH, and register B will contain 0FH.

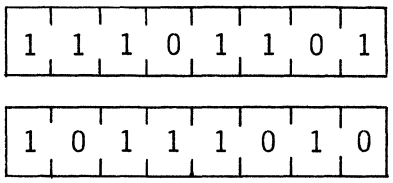
INDR

Operation: (HL) ← (C), B ← B-1, HL ← HL-1

Format:

Opcode

INDR



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 though 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 content of the HL register pair are placed on the address bus and the input byte is written into 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. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also interrupts will be recognized and two refresh cycles will be executed after each data transfer.

If B≠0:

M CYCLES: 5 T STATES: 21(4,5,3,4,5)

If B=0:

M CYCLES: 4 T STATES: 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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 after the execution of

INDR

the HL register pair will contain 0FFDH, register B will contain zero, and memory locations will have contents as follows:

Location	Contents
0FFEH	03H
0FFFH	A9H
1000H	51H

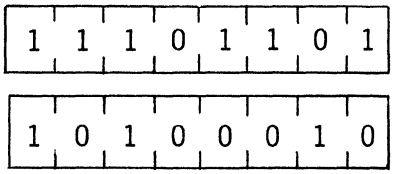
INI

Operation: (HL) ← (C), B ← B-1, HL ← HL + 1

Format:

Opcode

INI



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 into the corresponding location of memory. Finally the byte counter is decremented and register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set if B-1=0;
reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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 byte 7BH is available at the peripheral device

mapped to I/O port address 07H, then after the execution of

```
INI
```

memory location 1000H will contain 7BH, the HL register pair will contain 1001H, and register B will contain 0FH.

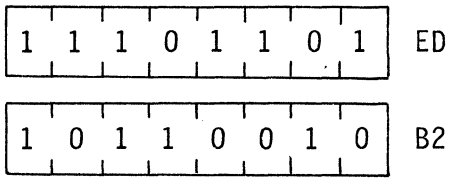
INIR

Operation: (HL) ← (C), B ← B-1, HL ← HL + 1

Format:

Opcode

INIR



Description:

The contents of register C are placed on the bottom half (A0 through A7) of t address bus to select the I/O device at one of 256 possible ports. Register B used as a byte counter, and its contents are placed on the top half (A8 through A1 of the address bus at this time. Then one byte from the selected port is plac on the data bus and written to the CPU. The contents of the HL register pair a placed on the address bus and the input byte is written in the corresponding locati of memory. Then register pair HL is incremented, the byte counter is decremente If decrementing causes B to go to zero, the instruction is terminated. If B is n zero, the PC is decremented by two and the instruction repeated. Note that if is set to zero prior to instruction execution, 256 bytes of data will be inpu Also interrupts will be recognized and two refresh cycles will be executed aft each data transfer.

If B≠0:

M CYCLES: 5 T STATES: 21(4,5,4,3,5)

If B=0:

M CYCLES: 4 T STATES: 16(4,5,4,3)

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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 after the execution of

INIR

the HL register pair will contain 1003H, register B will contain zero, and memory locations will have contents as follows;

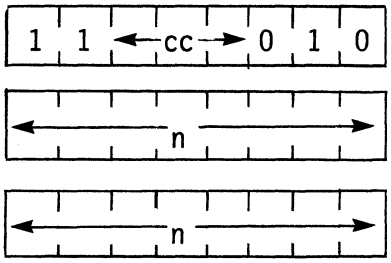
Location	Contents
1000H	51H
1001H	A9H
1002H	03H

JP cc, nn

Operation: IF cc TRUE, PC ← nn

Format:

Opcode	Operands
JP	cc, nn



Note: 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 into 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 bits which corresponds to condition bits in the Flag Register (register F). These eight status bits 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 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: 3 T STATES: 10(4,3,3)

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, after the execution of

```
JP C,1520H
```

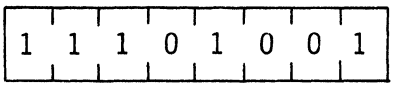
the Program Counter will contain 1520H, and on the next machine cycle the CPU will fetch from address 1520H the byte 03H.

JP (HL)

Operation: PC ← HL

Format:

Opcode	Operands
JP	(HL)



Description:

The Program Counter (register pair PC) is loaded with the contents of the HL registe pair. The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 1 T STATES: 4

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the HL registe pair are 4800H, after the execution of

JP (HL)

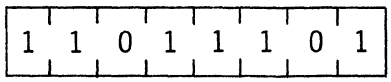
the contents of the Progam Counter will be 4800H.

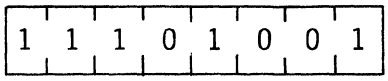
JP (IX)

Operation: PC ← IX

Format:

Opcode	Operands
JP	(IX)





Description:

The Program Counter (register pair PC) is loaded with the contents of the IX Register Pair (Index Register IX). The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 2 T STATES: 8(4,4)

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H, and the contents of the IX Register Pair are 4800H, after the execution of

```
JP (IX)
```

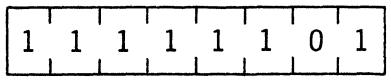
the contents of the Program Counter will be 4800H.

JP (IY)

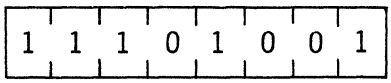
Operation: PC ← IY

Format:

Opcode	Operands
JP	(IY)



FD



E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the IY registe pair (Index Register IY). The next instruction is fetched from the location designate by the new contents of the PC.

M CYCLES: 2 T STATES: 8(4,4)

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the IY Registe Pair are 4800H, after the execution of

JP (IY)

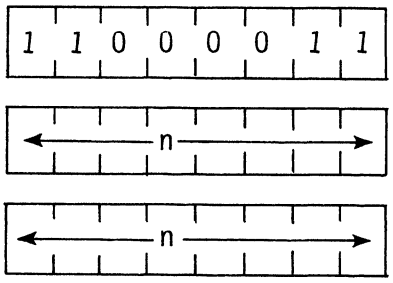
the contents of the Program Counter will be 4800H.

JP nn

Operation: PC ← nn

Format:

Opcode	Operands
JP	nn



Note: The first operand in this assembled object code is the low order byte of a 2-byte address.

Description:

Operand nn is loaded into register pair PC (Program Counter) and points to the address of the next program instruction to be executed.

M CYCLES: 3 T STATES: 10(4,3,3)

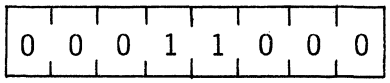
Condition Bits Affected: None

JR e

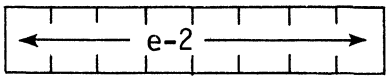
Operation: PC ←PC + e

Format:

Opcode	Operand
JR	e



18



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 opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

M CYCLES: 3 T STATES: 12(4,3,5)

Condition Bits Affected: None

Example:

To jump forward 5 locations from address 480, the following assembly language statement is used:

```
JR +5
```

The resulting object code and final PC value is shown below:

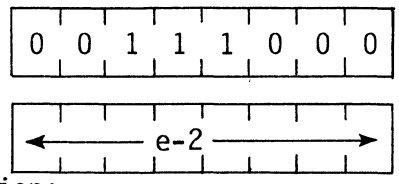
Location	Instruction
480	18
481	03
482	--
483	--
484	--
485	← PC after jump

JR C, e

Operation: IF C=0, continue
IF C=1, PC ← PC + e

Format:

Opcode	Operands
JR	C,e



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 opcode 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 to be executed is taken from the location following this instruction.

If condition is met:

M CYCLES: 3 T STATES: 12(4,3,5)

If condition is not met;

M CYLCES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

The Carry Flag is set and it is required to jump back 4 locations from 480. Th assembly language statement is:

JR C,-4

The resulting object code and final PC value is shown below:

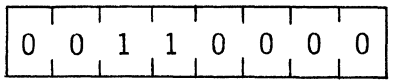
Location	Instruction
47C	←PC after jump
47D	—
47E	—
47F	—
480	38
481	FA (2's complement-6)

JR NC, e

Operation: If C=1, continue
If C=0, PC ← PC + e

Format:

Opcode	Operands
JR	NC,e



30

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 opcode 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 to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5)

If the condition is not met:

M CYCLES: 7 T STATES: 7(4,3)

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 shown below:

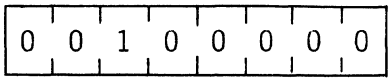
Location	Instruction
480	30 ← PC after jump
481	00

JR NZ, e

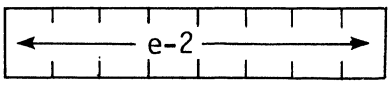
Operation: If Z = 1, continue
If Z = 0, PC ← PC + e

Format:

Opcode	Operands
JR	NZ,e



20



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 opcode 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 to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12,(4,3,5)

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

The Zero Flag is reset and it is required to jump back 4 locations from 480. The assembly language statement is :

JR NZ,-4

The resulting object code and final PC value is shown below:

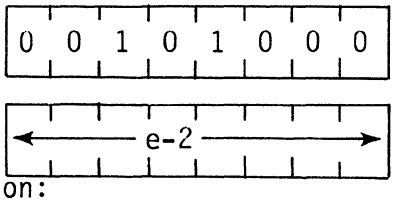
Location	Instruction
47C	← PC after jump
47D	—
47E	—
47F	—
480	20
481	FA (2' complement-6)

JR Z, e

Operation: If Z=0, continue
If Z=1, PC ← PC ← PC + e

Format:

Opcode	Operands
JR	Z,e



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 opcode 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 to be executed is taken from the location following this instruction.

If the conditon is met:

M CYCLES: 3 T STATES: 12(4,3,5)

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

The Zero Flag is set and it is required to jump forward 5 locations from address 300. The following assembly language statement is used:

JR Z, +5

The resulting object code and final PC value is shown below:

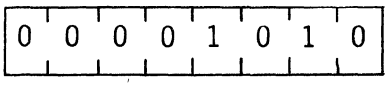
Location	Instruction
300	28
301	03
302	—
303	—
304	—
305	← PC after jump

LD A, (BC)

Operation: A ← (BC)

Format:

Opcode	Operands
LD	A, (BC)



Description:

The contents of the memory location specified by the contents of the BC register pair are loaded into the Accumulator.

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

If the BC register pair contains the number 4747H, and memory address 4747H contains the byte 12H, then the instruction

LD A, (BC)

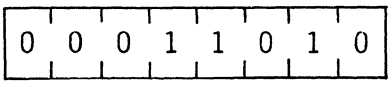
will result in byte 12H in register A.

LD A, (DE)

Operation: A ← (DE)

Format:

Opcode	Operands
LD	A, (DE)



1A

Description:

The contents of the memory location specified by the register pair DE are loaded into the Accumulator.

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

If the DE register pair contains the number 30A2H and memory address 30A2H contains the byte 22H, then the instruction

```
LD A, (DE)
```

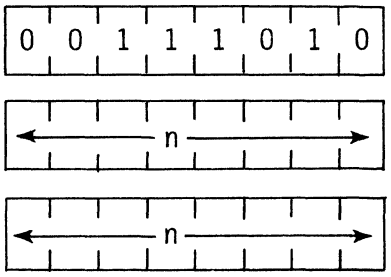
Will result in byte 22H in register A.

LD A, (nn)

Operation: A ⟵(nn)

Format:

Opcode	Operands
LD	A,(nn)



Description:

The contents of the memory location specified by the operands nn are loaded into the Accumulator. The first n operand is the low order byte of a two-byte memory address.

M CYCLES: 4 T STATES: 13(4,3,3,3)

Condition Bits Affected: None

Example:

If the contents of nn is number 8832H, and the contents of memory address 8832H is byte 04H, after the instruction

LD A, (nn)

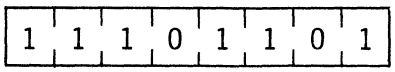
byte 04H will be in the Accumulator.

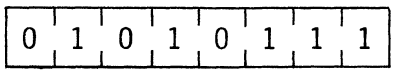
LD A, I

Operation: A ← I

Format:

Opcode	Operands
LD	A, I





Description:

The contents of the Interrupt Vector Register I are loaded into the Accumulator.

M CYCLES: 2 T states: 9(4,5)

Condition Bits Affected:

S: Set if I-Reg. is negative; reset otherwise
Z: Set if I-Reg. is zero; reset otherwise
H: Reset
P/V: Contains contents of IFF2
N: Reset
C: Not affected

Example:

If the Interrupt Vector Register contains the byte 4AH, after the execution of

```
LD A, I
```

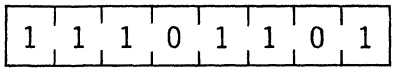
the accumulator will also contain 4AH.

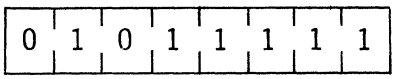
LD A, R

Operation: A ← R

Format:

Opcode	Operands
LD	A,R





Description:

The contents of Memory Refresh Register R are loaded into the Accumulator.

M CYCLES: 2 T STATES: 9(4,5)

Condition Bits Affected:

S: Set if R-Reg is negative; reset otherwise
Z: Set if R-Reg. is zero; reset otherwise
H: Reset
P/V: contains contents of IFF2
N: Reset
C: Not affected

Example:

If the Memory Refresh Register contains the byte 4AH, after the execution of

```
LD A,R
```

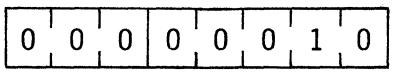
the Accumulator will also contain 4AH.

LD (BC), A

Operation: (BC) ← A

Format:

Opcode	Operands
LD	(BC),A



02

Description:

The contents of the Accumulator are loaded into the memory location specified by the contents of the register pair BC.

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example: If the Accumulator contains 7AH and the BC register pair contains 1212Hm the instruction

```
LD (BC),A
```

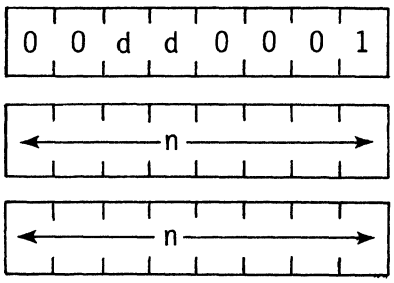
will result in 7AH being in memory location 1212H.

LD dd, nn

Operation: dd ← nn

Format:

Opcode	Operands
LD	dd, nn



Description:

The two-byte integer nn is loaded into the 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 in the assembled object code is the low order byte.

M CYCLES: 3 T STATES: 10(4,3,3)

Condition Bits Affected: None

Example:

After the execution of

LD HL, 5000H

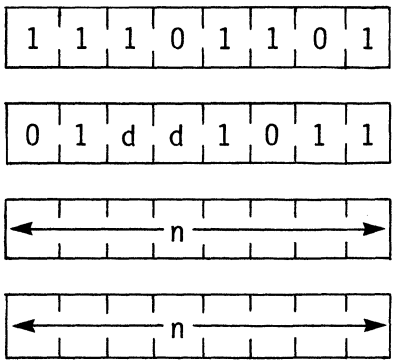
the contents of the HL register pair will be 5000H.

LD dd, (nn)

Operation: $dd_H \leftarrow (nn+1)$, $dd_L \leftarrow (nn)$

Format:

Opcode	Operands
LD	dd,(nn)



Description:

The contents of address nn are loaded into the low order portion of register pair dd, and the contents of the next highest memory address nn+1 are loaded into 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 in the assembled object code above is the low order byte of (nn).

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3,)

Condition Bits Affected: None

Example:

If Address 2130H contains 65H and address 2131H contains 78H after the instruction

```
LD BC, (2130H)
```

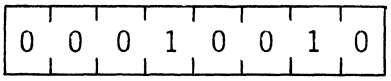
the BC register pair will contain 7865H.

LD (DE), A

Operation: (DE) ←A

Format:

Opcode	Operands
LD	(DE),A



12

Description:

The contents of the Accumulator are loaded into the memory location specified by the DE register pair.

M CYCLES: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

If the contents of register pair DE are 1128H, and the Accumulator contains byte AOH, the instruction

LD (DE),A

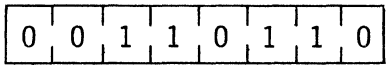
will result in AOH being in memory location 1128H.

LD (HL),n

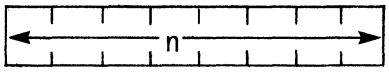
Operation: (HL) ← n

Format:

Opcode	Operand
LD	(HL),n



36



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

M CYCLES: 3 T STATES: 10(4,3,3)

Condition Bits Affected: None

Example:

If the HL register pair contains 4444H, the instruction

```
LD (HL), 28H
```

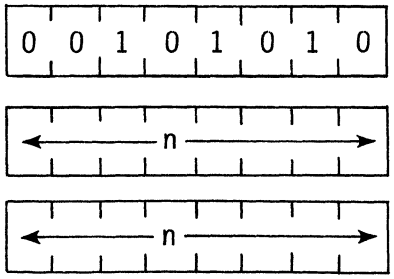
will result in the memory location 4444H containing the byte 28H.

LD HL,(nn)

Operation: H ← (nn+1), L ← (nn)

Format:

Opcode	Operands
LD	HL,(nn)



Description:

The contents of memory address nn are loaded into the low order portion of regist pair HL (register L), and the contents of the next highest memory address nn are loaded into the high order portion of HL (register H). The first n operand the assembled object code above is the low order byte of nn.

M CYCLES: 5 T STATES: 16(4,3,3,3,3,)

Condition Bits Affected: None

Example:

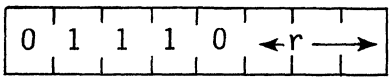
If address 4545H contains 37H and address 4546H contains A1H after the instructio
LD HL, (4545H)
the HL register pair will contain A137H.

LD (HL),r

Operation: (HL) ←r

Format:

Opcode	Operands
LD	(HL),r



Description:

The contents of register r are loaded into 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: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

If the contents of register pair HL specifies memory location 2146H, and the B register contains the byte 29H, after the execution of

```
LD (HL), B
```

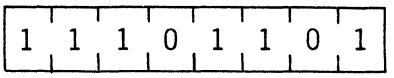
memory address 2146H will also contain 29H.

LD I, A

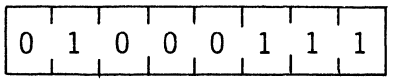
Operation: I ←—A

Format:

Opcode	Operands
LD	I,A



ED



47

Description:

The contents of the Accumulator are loaded into the Interrupt Control Vector Register, I.

M CYCLES: 2 T STATES: 9(4,5)

Condition Bits Affect: None

Example:

If the Accumulator contains the number 81H, after the instruction

```
LD I,A
```

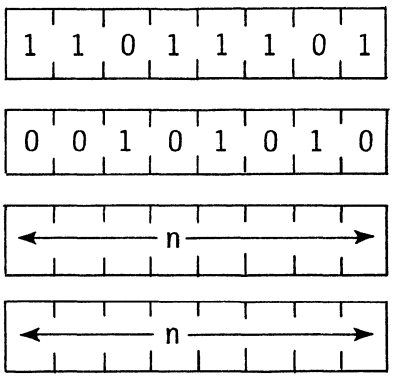
the Interrupt Vector Register will also contain 81H.

LD IX, (nn)

Operation: $IX_H \leftarrow (nn+1)$, $IX_L \leftarrow (nn)$

Format:

Opcode	Operands
LD	IX, (nn)



Description:

The contents of the address nn are loaded into the low order portion of Index Register IX, and the contents of the next highest memory address nn+1 are loaded into the high order portion of IX. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3,)

Condition Bits Affected: None

Example:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

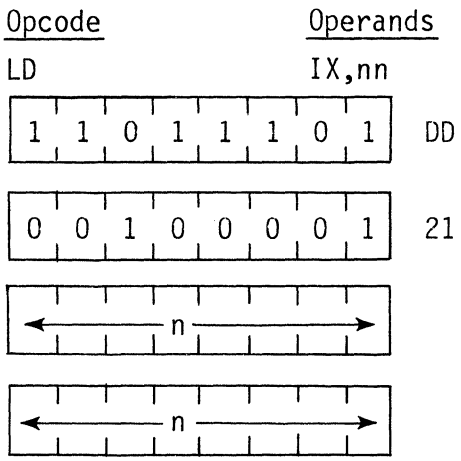
```
LD IX,(6666H)
```

the Index Register IX will contain DA92H.

LD IX,nn

Operation: IX ← nn

Format:



Description:

Integer nn is loaded into the Index Register IX. The first n operand in the assembled object code above is the low order byte.

M CYCLES: 4 T STATES: 14(4,4,3,3)

Condition Bits Affected: None

Example:

After the instruction

```
LD IX,45A2H
```

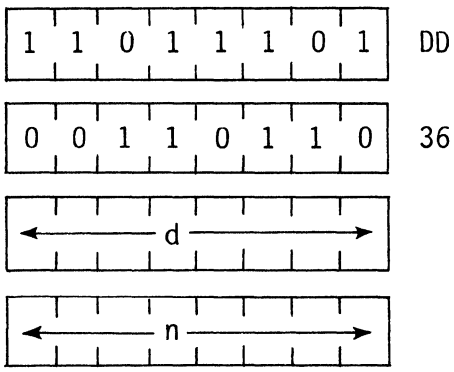
the Index Register will contain integer 45A2H.

LD(IX+d), n

Operation: (IX+d) ←n

Format:

Opcode	Operands
LD	(IX+d), n



Description:

The n operand is loaded into the memory address specified by the sum of the contents of the Index Register IX and the two's complement displacement operand d.

M CYLCLES: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected: None

Example:

If the Index Register IX contains the number 219AH the instruction

LD (IX+5H), 5AH

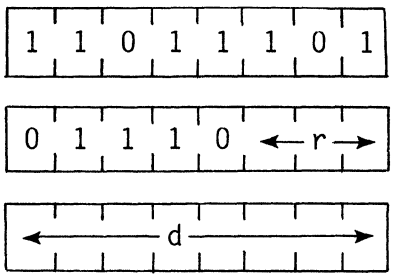
would result in the byte 5AH in the memory address 219FH.

LD (IX+d), r

Operation: (IX+d) ←r

Format:

Opcode	Operands
LD	(IX+d), r



Description:

The contents of register r are loaded into the memory address specified by the contents of Index Register IX summed with d, a two's complement displacement interger. 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: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected: None

Example:

If the C register contains the byte 1CH, and the Index Register IX contains 3100H, then the instruction

```
LD (IX+6H), C
```

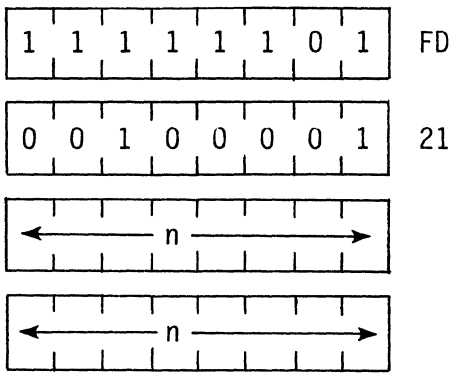
will perform the sum 3100H + 6H and will load 1CH into memory location 3106H.

LD IY, (nn)

Operation: IY ←nn

Format:

Opcode	Operands
LD	IY,nn



Description:

Integer nn is loaded into Index Register IY. The first n operand in the assembled object code above is the low order byte.

M CYCLES: 4 T STATES: 14(4,4,3,3)

Condition Bits Affected: None

Example:

After the instruction:

```
LD IY, 7733H
```

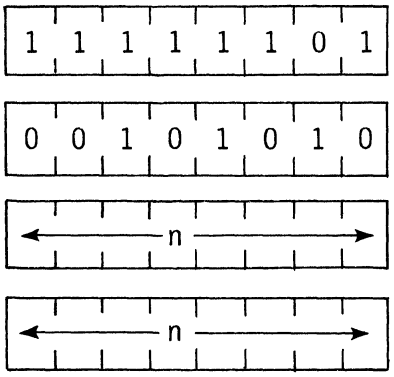
the Index Register IY will contain the integer 7733H.

LD IY, nn

Operation: IY ←(nn+1), IY_L ←(nn)

Format:

Opcode	Operands
LD	IY,(nn)



Description:

The contents of address nn are loaded into the low order portion of Index Register IY, and the contents of the next highest memory address nn+1 are loaded into the high order portion of IY. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3)

Condition Bits Affected: None

Example:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

LD IY, (6666H)

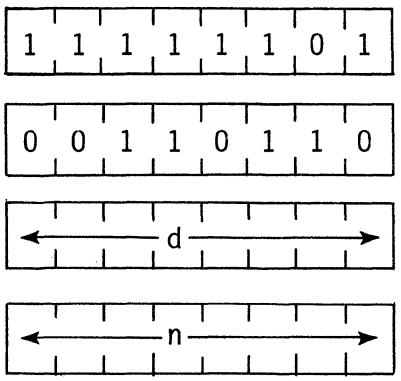
the Index Register IY will contain DA92H.

LD (IY+d), n

Operation: (IY+d) ←n

Format:

Opcode	Operands
LD	(IY+D), n



Description:

Integer n is loaded into memory location specified by the contents of the Index Register summed with a displacement interger d.

M CYCLES: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected: None

Example:

If the Index Register IY contains the number A940H, the instruction

LD (IY+10H), 97H

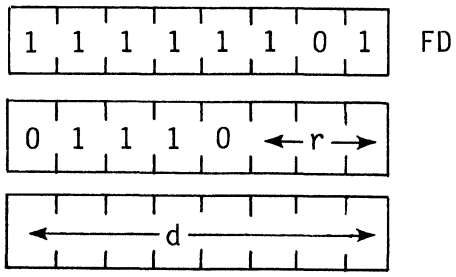
would result in byte 97H in memory location A950H.

LD (IY+d),r

Operation: (IY+d) ←r

Format:

Opcode	Operands
LD	(IY+d), r



Description:

The contents of register r are loaded into 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: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected: None

Example:

If the C register contains the byte 48H, and the Index Register IY contains 2A11H, then the instruction

```
LD (IY+4H), C
```

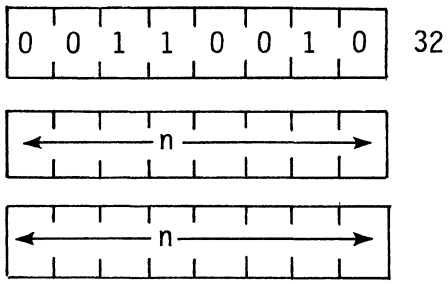
will perform the sum 2A11H + 4H, and will load 48H into memory location 2A15H.

LD (nn), A

Operation: (nn) ←A

Format:

Opcode	Operands
LD	(nn),A



Description:

The contents of the Accumulator are loaded into the memory address specified by the operands nn. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 4 T STATES: 13(4,3,3,3)

Condition Bits Affected: None

Example:

If the contents of the Accumulator are byte D7H, after the execution of

```
LD (3141H),A
```

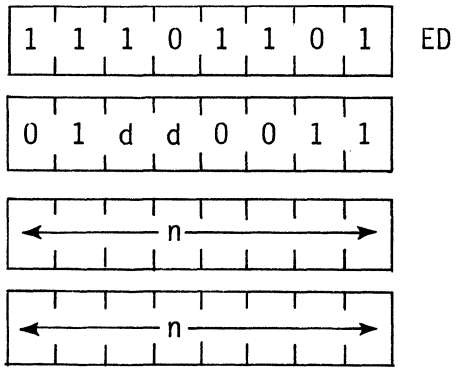
D7H will be in memory location 3141H.

LD (nn), dd

Operation: (nn+1) ←dd_H, (nn) ←dd_L

Format:

Opcode	Operands
LD	(nn),dd



Description:

The low order byte of register pair dd is loaded into memory address nn ; the upper byte is loaded into 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 in the assembled object code is the low order byte of a two byte memory address.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3)

Condition Bits Affected: None

Example:

If register pair BC contains the number 4644H, the instruction

```
LD (1000H),BC
```

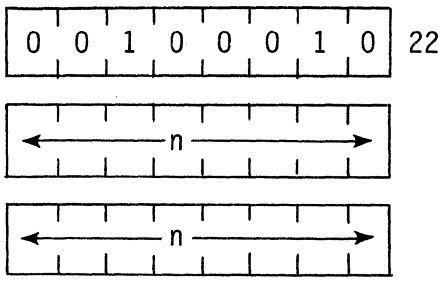
will result in 44H in memory location 1000H, and 46H in memory location 1001H.

LD (nn), HL

Operation: (nn+1) ←H, (nn) ←L

Format:

Opcode	Operands
LD	(nn),HL



Description.

The contents of the low order portion of register pair HL (register L) are loaded into memory address nn , and the contents of the high order portion of HL (register H) are loaded into the next highest memory address nn+1. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 5 T STATES: 17(4,3,3,3,3)

Condition Bits Affected: None

Example:

If the content of register pair HL is 483AH, after the instruction

```
LD  (B229H), HL
```

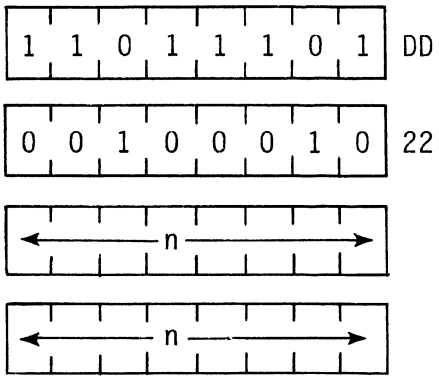
address B229H will contain 3AH, and address B22AH will contain 48H.

LD (nn), IX

Operation: $(nn+1) \leftarrow IX_H$ $(nn) \leftarrow IX_L$

Format:

Opcode	Operands
LD	(nn),IX



Description:

The low order byte in Index Register IX Is loaded into memory address nn ; the upper order byte is loaded into the next highest address nn+1. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3)

Condition Bits Affected: None

Example:

If the Index Register IX contains 5A30H, after the instruction

LD (4392H),IX

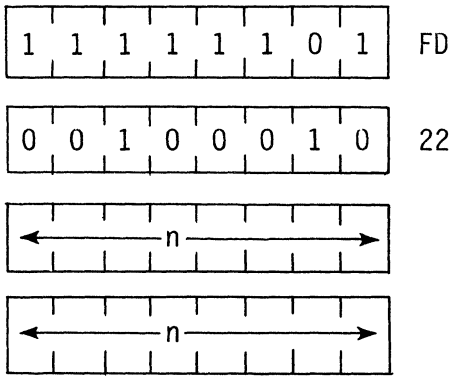
memory location 4392H will contain number 30H and location 4393H will contain 5AH.

LD (nn), IY

Operation: $(nn+1) \leftarrow IY_H$, $(nn) \leftarrow IY_L$

Format:

Opcode	Operands
LD	(nn),IY



Description:

The low order byte in Index Register IY is loaded into memory address nn ; the upper order byte is loaded into memory location nn+1. The first n operand in the assembled object code above is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3)

Condition Bits Affected: None

Example:

If the Index Register IY contains 4174H after the instruction

LD 8838H,IY

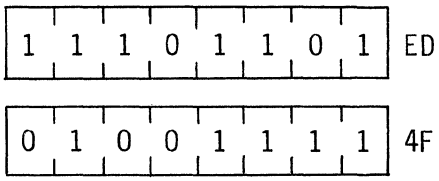
memory location 8838H will contain number 74H and memory location 8839H will contain 41H.

LD R,A

Operation: R ←—A

Format:

Opcode	Operands
LD	R,A



Description:

The contents of the Accumulator are loaded into Memory Refresh register R.

M CYCLES: 2 T STATES: 9(4,5)

Condition Bits Affected: None

Example:

If the Accumulator contains the number B4H, after the instruction

```
LD R,A
```

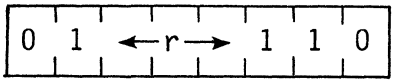
the Memory Refresh Register will also contain B4H.

LD r, (HL)

Operation: r ←—(HL)

Format:

Opcode	Operands
LD	r,(HL)



Description:
The eight-bit contents of memory location (HC) are loaded into register r, where r identifies register A,B,C,D,E,G 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: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

If register pair HL contains the number 75A1H, and memory address 75A1H contains the byte 58H, the execution of

```
LD C,(HL)
```

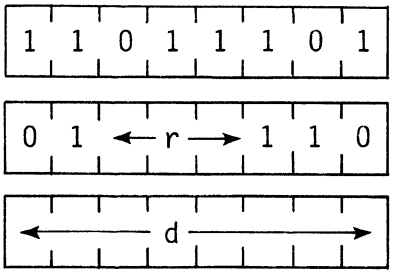
will result in 58H in register C.

LD r, (IX+d)

Operation: r ←(IX+d)

Format:

Opcode	Operands
LD	r, (IX+d)



Description:

The operand (IX+d) (the contents of the Index Register IX summed with a displacement integer d) is loaded into 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: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected: None

Example:

If the Index Register IX contains the number 25AFH, the instruction

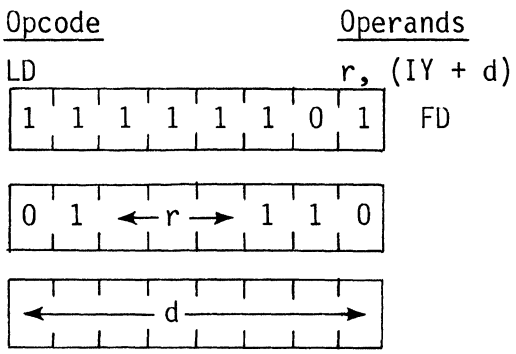
LD B, (IX+19H)

will cause the calculation of the sum 25AFH + 19H, which points to memory locati 25C8H. If this address contains byte 39H, the instruction will result in regist B also containing 39H.

LD r, (IY+d)

Operation: r ←—(IY + d)

Format:



Description:

The operand (IY + d) (the contents of the Index Register IY summed with a displacement integer d) is loaded into register r, where r identifies A, B, C, D, E, H or L, assembled as follows in the object code:

Register

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

M CYCLES: 5 T STATES: 19(4,4,3,5,3)

Condition Bits Affected: None

Example:

If the Index Register IY contains the number 25AFH, the instruction

LD B, (IY+19H)

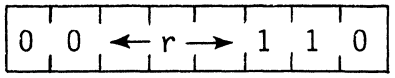
will cause the calculation of the sum 25AFH + 19H, which points to memory locati 25C8H. If this address contains byte 39H, the instruction will result in regist B also containing 39H.

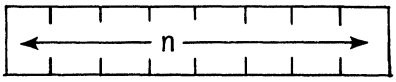
LD r, n

Operation: r ← n

Format:

Opcode	Operands
LD	r, n





Description:

The eight-bit integer n is loaded into 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: 2 T STATES: 7(4,3)

Condition Bits Affected: None

Example:

After the execution of

```
LD E, A5H
```

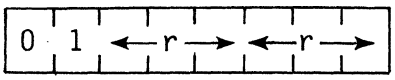
the contents of register E will be A5H.

LD r, r'

Operation: r ← r'

Format:

Opcode	Operands
LD	r,r'



Description:

The contents of any register r' are loaded into any other register r. Note: r,r' identifies any of the registers A, B, C, D, E, H, or L, assembled as follows in the object code:

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

M CYCLES: 1 T STATES: 4

Condition Bits Affected: None

Example:

If the H register contains the number 8AH, and the E register contains 10H, the instruction

```
LD H, E
```

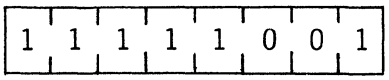
would result in both registers containing 10H.

LD SP, HL

Operation: SP ←HL

Format:

Opcode	Operands
LD	SP,HL



F9

Description:

The contents of the register pair HL are loaded into the Stack Pointer SP.

M CYCLES: 1 T STATES: 6

Condition Bits Affected: None

Example:

If the register pair HL contains 442EH, after the instruction

```
LD SP,HL
```

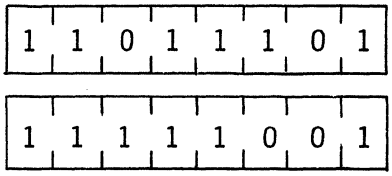
the Stack Pointer will also contain 442EH.

LD SP, IX

Operation: SP ←—IX

Format:

Opcode	Operands
LD	SP,IX



Description:

The two byte contents of Index Register IX are loaded into the Stack Pointer SP.

M CYCLES: 2 T STATES: 10(4,6)

Condition Bits Affected: None

Example:

If the contents of the Index Register IX are 98DAH, after instruction

LD SP,IX

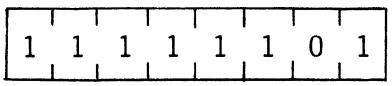
the contents of the Stack Pointer will also be 98DAH.

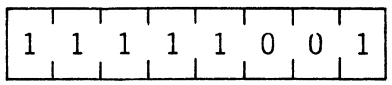
LD SP, IY

Operation: SP ← IY

Format:

Opcode	Operands
LD	SP,IY





Description:

The two byte contents of Index Register IY are loaded into the Stack Pointer SP.

M CYCLES: 2 T STATES: 10(4,6)

Condition Bits Affected: None

Example:

If Index Register IY contains the integer A227H, after the instruction

```
LD SP,IY
```

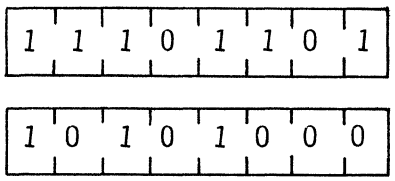
the Stack Pointer will also contain A227H.

LDD

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

Format:

Opcode	Operands
LDD	



Description:

This two 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: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Set if BC-1≠0;
reset otherwise
N: Reset
C: Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains the byte 88H, the DE register pair contains 2222H, memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

```
LDD
```

will result in the following contents in register pairs and memory addresses:

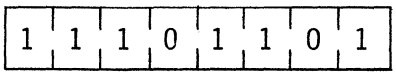
```
   HL   : 1110H
(1111H) :   88H
   DE   : 2221H
(2222H) :   88H
   BC   :    6H
```

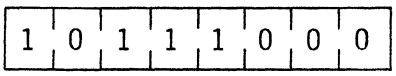
LDDR

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

Format:

Opcode	Operands
LDDR	





Description:

This two-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 as well as the BC (Byte Counter) are 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 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes. Also, interrupts will be recognized and two refresh cycles will be executed after each data transfer.

For BC≠0:

M CYCLES: 5 T STATES: 21(4,4,3,5,5)

For BC=0:

M CYCLES: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Reset
N: Reset
C: Not affected

Example:

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

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

then after the execution of

```
LDDR
```

the contents of register pairs and memory locations will be:

HL : 1111H
DE : 2222H
BC : 0000H

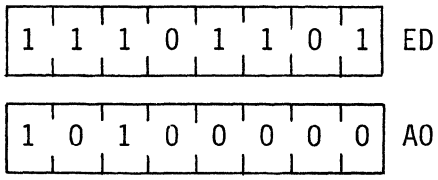
(1114H)	: A5H	(2225H)	: A5H
(1112H)	: 36H	(2224H)	: 36H
(1112H)	: 88H	(2223H)	: 88H

LDI

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

Format:

Opcode	Operands
LDI	



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: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Set if BC-1≠0;
reset other wise
N: Reset
C: Not affected

Example:

If the HL Register pair contains 1111H, memory location 1111H contains the 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
```

will result in the following contents in register pairs and memory addresses:

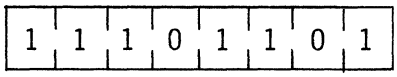
```
   HL   :   1112H
(1111H) :   88H
   DE   :  2223H
(2222H) :    88H
   BC   :     6H
```

LDIR

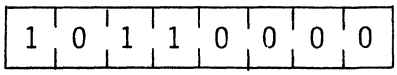
Operation: (DE) ←(HL), DE ← DE+1, HL ← HL+1, BC ← BC-1

Format:

Opcode	Operands
LDIR	



ED



B0

Description:

This two byte instruction transfers a byte of data from the memory location addres by the contents of the HL register pair to the memory location addressed by the l register pair. Then both these register pairs are incremented and the BC (By Counter) register pair is decremented. If decrementing causes the BC to go to zei the instruction is terminated. If BC is not zero the program counter is decrement by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruc tion execution, the instruction will loop through 64K bytes. Also, interrupts wi be recognized and two refresh cycles will be executed after each data transfei

For BC≠0:

M CYCLES: 5 T STATES: 21(4,4,3,5,5)

For BC=0:

M CYCLES: 4 T STATES: 16(4,4,3,5)

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Reset
N: Reset
C: Not affected

Example:

If the HL register pair contains 1111H, the DE register pair contains 2222H, the BC register pair contains 0003H, and memory locations have these contents:

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

then after the execution of

```
LDIR
```

the contents of register pairs and memory locations will be:

HL : 1114H
DE : 2225H
BC : 0000H

(1111H)	: 88H	(2222H)	: 88H
(1112H)	: 36H	(2223H)	: 36H
(1113H)	: A5H	(2224H)	: A5H

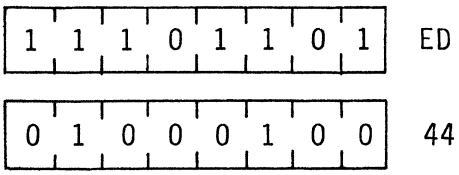
NEG

Operation: A ← 0 -A

Format:

Opcode

NEG



Description:

Contents of the Accumulator are negated (two's complement). This is the same subtracting the contents of the Accumulator from zero. Note that 80H is left unchange

M CYCLES: 2 T STATES: 8(4,4)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Set if there is a borrow from Bit 4; reset otherwise

P/V: Set if Acc. was 80H before operation; reset otherwise

N: Set

C: Set if Acc. was not 00H before operation; reset otherwise

Example:

If the contents of the Accumulator are

1 0 0 1 1 0 0 0

after the execution of

NEG

the Accumulator contents will be

0 1 1 0 1 0 0 0

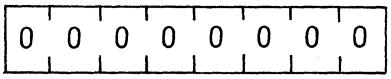
NOP

Operation: ——

Format:

Opcode

NOP



00

Description:

CPU performs no operation during this machine cycle.

M CYCLES: 1 T STATES: 4

Condition Bits Affected: None

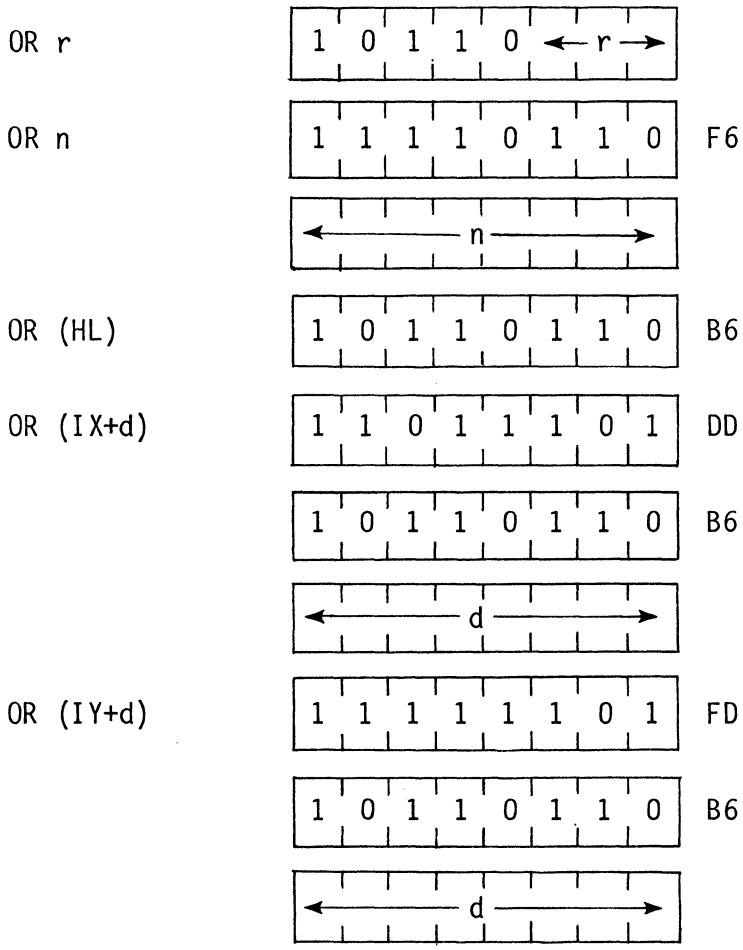
OR s

Operation: A ⟵A V s

Format:

Opcode	Operands
OR	s

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



*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:

A logical OR operation, bit by bit, it performed between the byte specified by the operand and the byte contained in the Accumulator; the result is stored in th Accumulator.

INSTRUCTION	M CYCLES	T STATES
OR r	1	4
OR n	2	7(4,3)
OR (HL)	2	7(4,3)
OR (IX+d)	5	19(4,4,3,5,3)
OR (IY+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

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

Example:

If the H register contains 48H (01001000) and the Accumulator contains 12H (00010010 after the execution of

OR H

the Accumulator will contain 5AH (01011010).

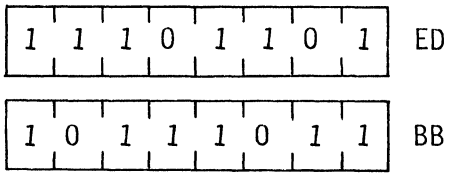
OTDR

Operation: (C) ← (HL), B ←B-1, HL ← HL-1

Format:

Opcode

OTDR



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 into 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 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data. Also, interrupts will be recognized and two refresh cycles will be executed after each data transfer.

If B≠0:

M CLCLES: 5 T STATES: 21(4,5,3,4,5)

If B=0;

M CYCLES: 4 T STATES: 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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:

Location	Contents
0FFEH	51H
0FFFH	A9H
1000H	03H

then after the execution of

OTDR

the HL register pair will contain 0FFDH, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

03H
A9H
51H

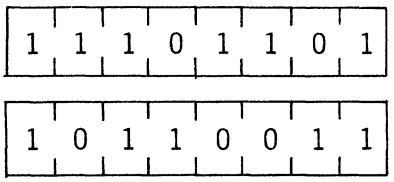
OTIR

Operation: (C) ←(HL), B ← B-1, HL ←HL + 1

Format:

Opcode

OTIR



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 into 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 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data. Also, interrupts will be recognized and two refresh cycles will be executed after each data transfer.

IF B≠0:

M CYCLES: 5 T STATES: 21(4,5,3,4,5)

If B=0:

M CYCLES: 4 T STATES: 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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:

Location	Contents
1000H	51H
1001H	A9H
1002H	03H

then after the execution of

```
OTIR
```

the HL register pair will contain 1003H, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

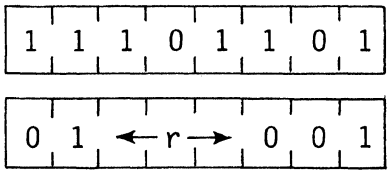
```
51H
A9H
03H
```

OUT (C), r

Operation: (C) ←r

Format:

Opcode	Operands
OUT	(C),r



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 into the selected peripheral device. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each which appears in the assembled object code:

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

M CYCLES: 3 T STATES: 12(4,4,4)

Condition Bits Affected: None

Example:

If the contents of register C are 01H and the contents of register D are 5AH, after the execution of

```
OUT (C),D
```

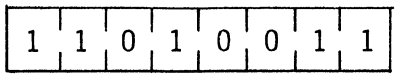
the byte 5AH will have been written to the peripheral device mapped to I/O port address 01H.

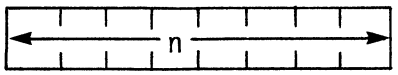
OUT (n), A

Operation: (n) ←A

Format:

Opcode	Operands
OUT	(n),A





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 into the selected peripheral device.

M CYCLES: 3 T STATES: 11(4,3,4)

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H, then after the execution of

```
OUT 01H,A
```

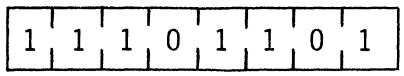
the byte 23H will have been written to the peripheral device mapped to I/O port address 01H.

OUTD

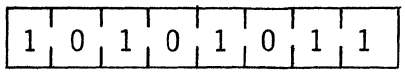
Operation: (C) ← (HL), B ←B-1, HL ←HL-1

Format:

Opcode



ED



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 156 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 into the selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set if B-1=0;
reset other wise
H: Unknown
P/V: Unknown
N: Set
C: Unknown

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, after the execution of

```
OUTD
```

register B will contain 0FH, the HL register pair will contain 0FFFH, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

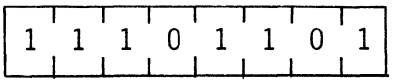
OUTI

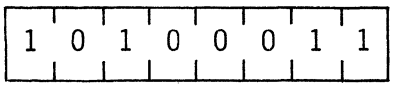
Operation: (C) ← (HL), B ←B-1, HL ←HL + 1

Format:

Opcode

OUTI





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 into selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4)

Condition Bits Affected:

S: Unknown
Z: Set if B-1=0;
reset otherwise
P/V: Unknown
N: Set
C: Unknown

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 address 1000H

are 59H, then after the execution of

OUTI

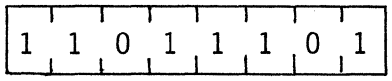
register B will contain 0FH, the HL register pair will contain 1001H, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

POP IX

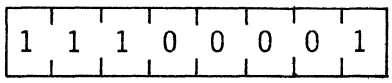
Operation: $IX_H \leftarrow (SP+1)$, $IX_L \leftarrow (SP)$

Format:

Opcode	Operands
POP	IX



DD



E1

Description:

The top two bytes of the external memory LIFO (last-in, first -out) Stack are popped into 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 into 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 into the high order portion IX. The SP is now incremented again.

M CYCLES: 4 T STATES: 14(4,4,3,3)

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
```

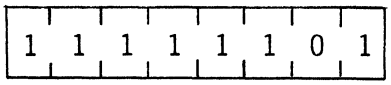
will result in Index Register IX containing 3355H, and the Stack Pointer containing 1002H.

POP IY

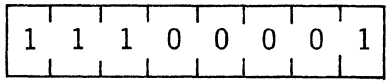
Operation: $IY_H \leftarrow (SP+1)$, $IY_L \leftarrow (SP)$

Format:

Opcode	Operands
POP	IY



FD



E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into 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 into 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 into the high order portion of IY. The SP is now incremented again.

M CYCLES: 4 T STATES: 14(4,4,3,3)

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
```

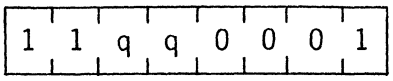
will result in Index Register IY containing 3355H, and the Stack Pointer containing 1002H.

POP qq

Operation: $qq_H \leftarrow (SP+1)$, $qq_L \leftarrow (SP)$

Format:

Opcode	Operands
POP	qq



Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into 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 into the low order portion of qq, 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 into the high order portion of qq and the SP is now incremented again. The operand qq defines 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: 3 T STATES: 10(4,3,3)

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
```

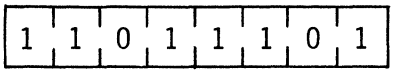
will result in register pair HL containing 3355H, and the Stack Pointer containing 1002H.

PUSH IX

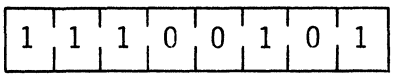
Operation: $(SP-2) \leftarrow IX_L$ $(SP-1) \leftarrow IX_H$

Format:

Opcode	Operands
PUSH	IX



DD



E5

Description:

The contents of the Index Register IX are pushed into 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 IX into the memory address now specified by the SP, then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M CYCLES: 3 T STATES: 15(4,5,3,3)

Condition Bits Affected: None

Example:

If the Index Register IX contains 2233H and the Stack Pointer contains 1007H, after the instruction

```
PUSH IX
```

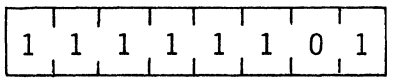
Memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

PUSH IY

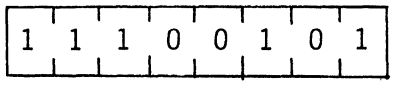
Operation: (SP-2) ← IY_L, (SP-1) ← IY_H

Format:

Opcode	Operands
PUSH	IY



FD



E5

Description:

The contents of the Index Register IY are pushed into 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 into the memory address now specified by the SP; then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M CYCLES: 4 T STATES: 15(4,5,3,3)

Condition Bits Affected: None

Example:

If the Index Register IY contains 2233H and the Stack Pointer contains 1007H, after the instruction

```
PUSH IY
```

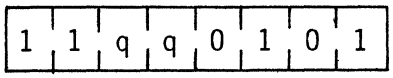
Memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

PUSH qq

Operation: (SP-2) ← qq_L , (SP-1) ← qq_H

Format:

Opcode	Operands
PUSH	qq



Description:

The contents of the register pair qq are pushed into the external memory LIFO (last-in first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address o the current "top" of the Stack. This instruction first decrements the SP and load the high order byte of register pair qq into the memory address now specified by the SP then decrements the SP again and loads the low order byte of qq into the memor location corresponding to this new address in the SP. The operand qq means registe 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: 3 T STATES: 11(5,3,3)

Conditon Bits Affected: None

Example:

If the AF register pair contains 2233H and the Stack Pointer contains 1007H, afte the instruction

PUSH AF

memory address 1006H will contain 22H, memory address 1005H will contain 33H, and th Stack Pointer will contain 1005H.

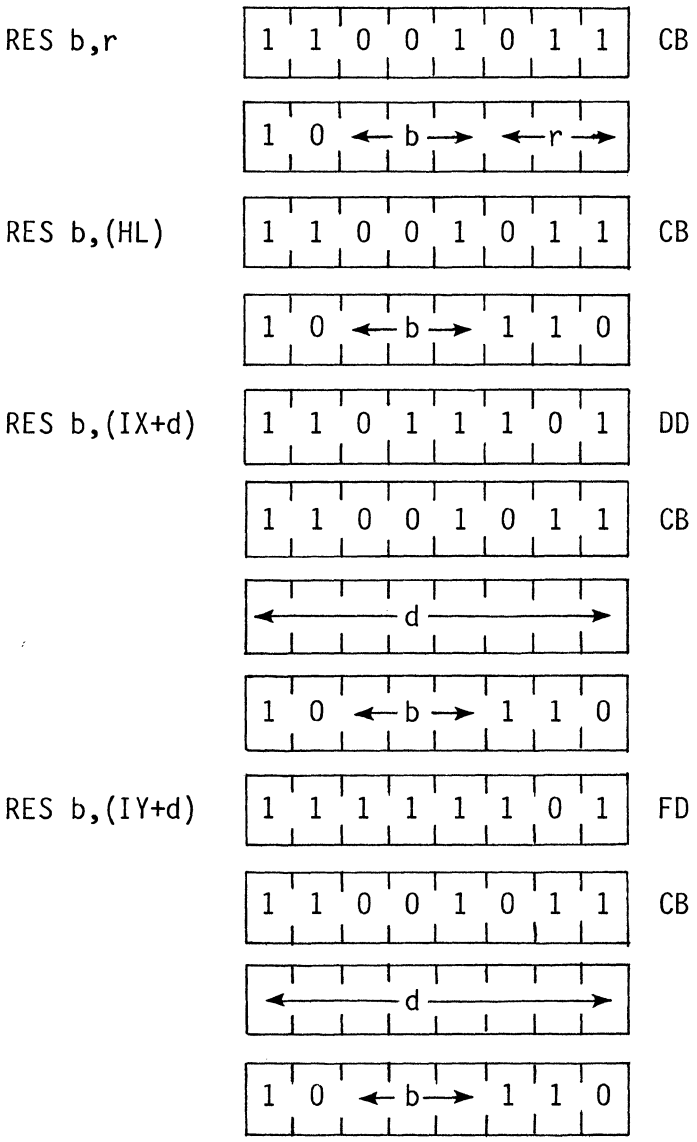
RES b, m

Operation: $s_b \leftarrow 0$

Format:

Opcode	Operands
RES	b,m

Operand b is any bit (7 through 0) of the contents of the m operand, (any of r, (HL), (IX+d) or (IY+d)) as defined for the analogous SET instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



Bit Reset	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
RES r	4	8(4,4)
RES (HL)	4	15(4,4,4,3)
RES (IX+d)	6	23(4,4,3,5,4,3)
RES (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected: None

Example:

After the execution of

```
RES 6,D
```

bit 6 in register D will be reset. (Bit 0 in register D is the least significant bit.)

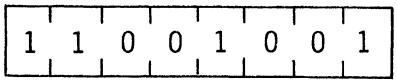
RET

Operation: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$

Format:

Opcode

RET



C9

Description:

Control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address pointed to by the Stack Pointer (SP), then incrementing the SP and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP. (The SP is now incremented a second time.) On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC.

M CYCLES: 3 T STATES: 10(4,3,3)

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 2001H are 18H, then after the execution of

RET

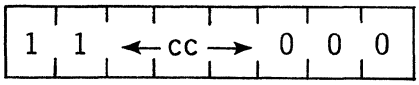
the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

RET cc

Operation: IF cc TRUE: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$

Format:

Opcode	Operand
RET	cc



Description:

If condition cc is true, control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address pointed to by the Stack Pointer (SP), then incrementing the SP, and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP (the SP is now incremented a second time). On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC. 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 which 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: 3 T STATES: 11(5,3,3)

If cc is false:

M CYCLES: 1 T STATES: 5

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, then after the execution of

```
RET M
```

the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

RETI

Operation: Return from interrupt

Format:

Opcode

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 an interrupt service routine to:

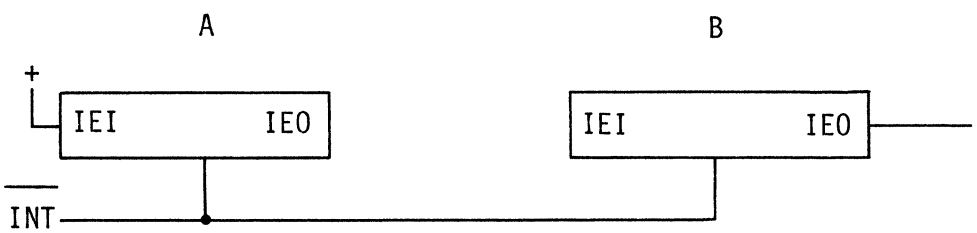
1. Restore the contents of the Program Counter (PC) (analogous to the RET instruction)
2. To signal an I/O device that the interrupt routine has been completed. The RETI instruction facilitates the nesting of interrupts allowing higher priority devices to suspend service of lower-priority service routines. This instruction has no effect on the IFF1 and IFF2 flip flops.

M CYCLES: 4 T STATES: 14(4,4,3,3)

Condition Bits Affected: None

Example:

Given: Two interrupting devices, A and B,connected in a daisy chain configuration with 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 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 IEO of B is reset (high) allowing lower priority devices interrupt access.

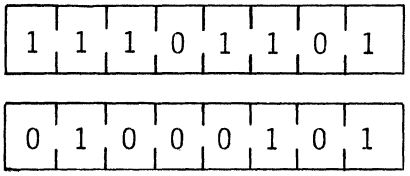
RETN

Operation: Return from non-maskable interrupt

Format:

Opcode

RETN



Description:

Used at the end of a service routine for a non-maskable interrupt, this instruction executes an unconditional return which functions identically to the RET instruction. That is, the previously stored contents of the Program Counter (PC) are popped off the top of the external memory stack; the low-order byte of PC is loaded with the contents of the memory location pointed to by the Stack Pointer (SP), SP is incremented, the high-order byte of PC is loaded with the contents of the memory location now pointed to by SP, and SP is incremented again. Control is now returned to the original program flow: on the following machine cycle the CPU will fetch the next opcode from the location in memory now pointed to by the PC. Also the state of IFF2 is copied back into IFF1 to the state it had prior to the acceptance of the NMI.

M CYCLES: 4 T STATES: 14(4,4,3,3)

Condition Bits Affected: None

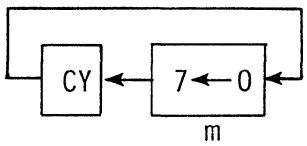
Example:

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 will ignore the next instruction and will instead restart to memory address 0066H. That is, the current Program Counter contents of 1A45H will be pushed onto the external stack address of OFFFH and OFFEH, high order-byte first, and 0066H will be loaded onto the

Program Counter. That address begins an interrupt service routine which ends with 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 opcode fetch to address 1A45H.

RL m

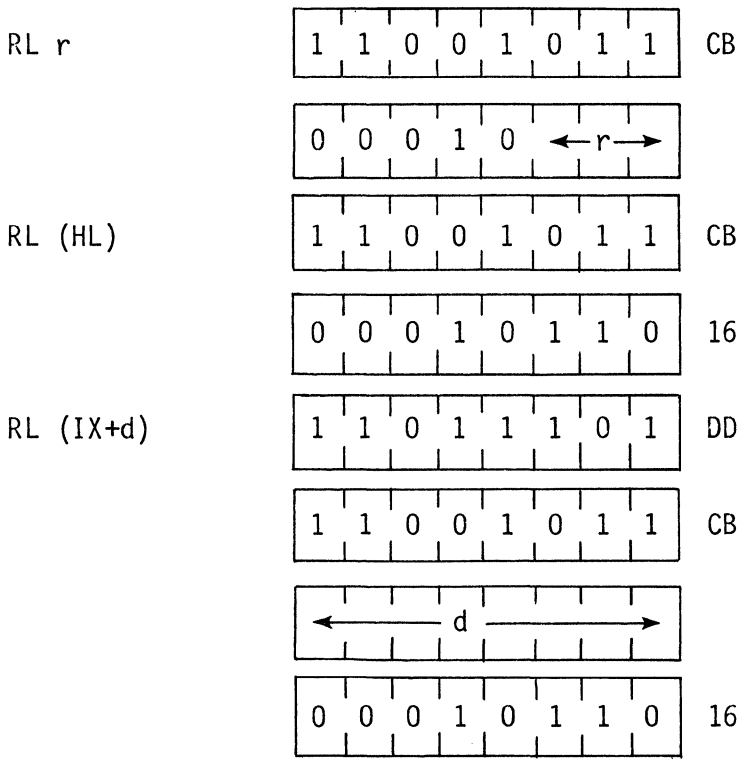
Operation:



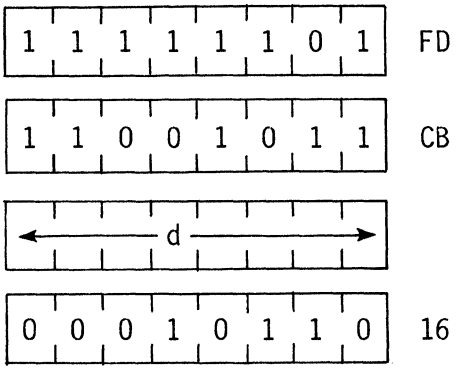
Format:

Opcode	Operands
RL	m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



RL (IY+d)



*r identifies register B,C,D,E,H,L or A specified as follows in the assembled object code 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: the content of bit 0 is copied into bit 1; the previous content of bit content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 0 (Bit 0 is the least significant bit.)

INSTRUCTION	M CYCLES	T STATES
RL r	2	8(4,4)
RL (HL)	4	15(4,4,4,3)
RL (IX+d)	6	23(4,4,3,5,4,3)
RL (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if result is zero;
reset otherwise

H: Reset

P/V: Set if parity even;
reset otherwise

N: Reset

C: 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

after the execution of

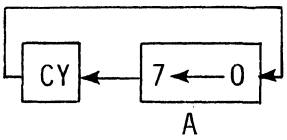
RL D

the contents of register D and the Carry Flag will be

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

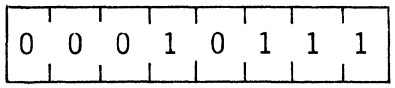
RLA

Operation:



Format:

Opcode	Operands
RLA	



17

Description:

The contents of the Accumulator (register A) are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 0. Bit 0 is the least significant bit.

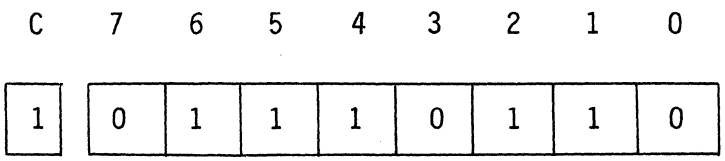
M CYCLES: 1 T STATES: 4

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 7 or Acc.

Example:

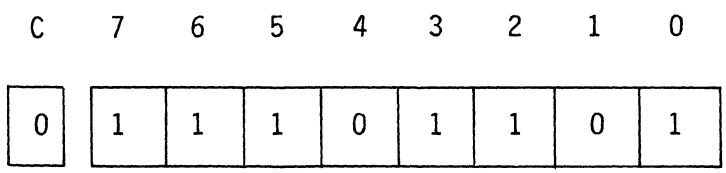
If the contents of the Accumulator and the Carry Flag are



after the execution of

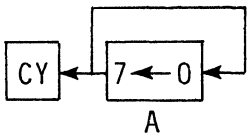
RLA

the contents of the Accumulator and the Carry Flag will be



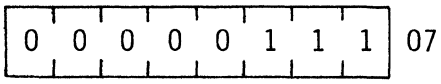
RLCA

Operation:



Format:

Opcode	Operands
RLCA	



Description:

The contents of the Accumulator (register A) are rotated left: the content of bit 0 is moved to bit 1; the previous content of bit 1 is moved to bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. (Bit 0 is the least significant bit.)

M CYCLES: 1 T STATES: 4

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 7 of Acc.

Example:

If the contents of the Accumulator are

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

after the execution of

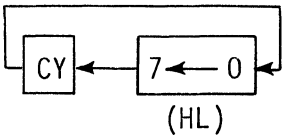
RLCA

the contents of the Accumulator and Carry Flag will be

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

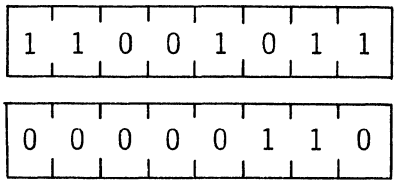
RLC (HL)

Operation:



Format:

Opcode	Operands
RLC	(HL)



Description:

The contents of the memory address specified by the contents of register pair HL are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 4 T STATES: 15(4,4,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Reset

P/V: Set if parity even; reset otherwise

N: Reset

C: 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

after the execution of

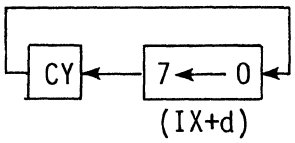
RLC (HL)

the contents of memory location 2828H and the Carry Flag will be

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

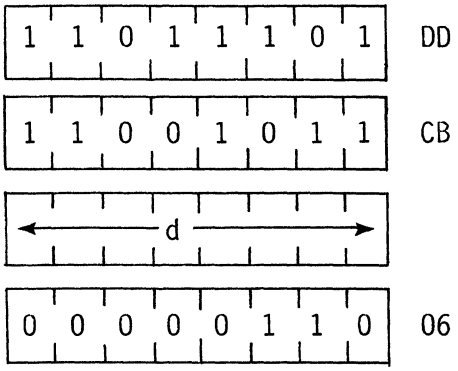
RLC(IX+d)

Operation:



Format:

Opcode	Operands
RLC	(IX+d)



Description:

The contents of he memory address specified by the sum of the contents of the Index Register IX and a two's complement displacement interger d, are rotated left: the contents of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if result is zero;
reset other wise

H: Reset

P/V: Set if parity even;
reset otherwise

N: Reset

C: Data from Bit 7 of
source register

Example:

If the contents of the Index Register IX 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

after the execution of

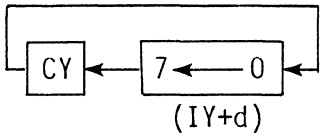
RLC (IX+2H)

the contents of memory location 1002H and the Carry Flag will be

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

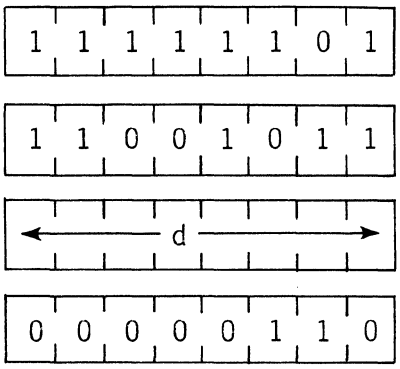
RLC (IY+d)

Operation:



Format:

Opcode	Operands
RLC	(IY+d)



Description:

The contents of the memory address specified by the sum of the contents of the Index Register IY and a two's compliment displacement interger d are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit 2; this process is continued throughout the byte. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

P/V: Set if parity even; reset otherwise

N: Reset

C: 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

after the execution of

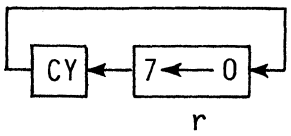
RLC (IY+2H)

the contents of memory location 1002H and the Carry Flag will be

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

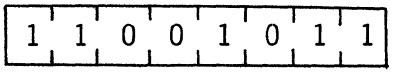
RLC r

Operation:

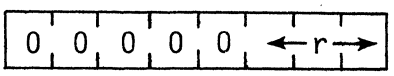


Format:

Opcode	Operands
RLC	r



CB



Description:

The eight-bit contents of register r are rotated left: the content of bit 0 is copied into bit 1; the previous content of bit 1 is copied into bit2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into 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

Note: Bit 0 is the least significant bit.

CYCLES: 2 T STATES: 8(4,4)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if result is zero;
reset otherwise

H: Reset

P/V: Set if parity even;
reset otherwise

N: Reset

C: 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

after the execution of

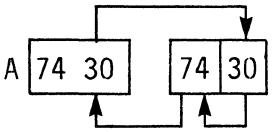
RLC r

the contents of register r and the Carry Flag will be

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

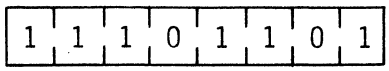
RLD

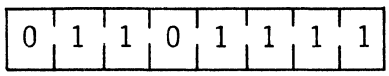
Operation:



Format:

Opcode	Operands
RLD	





Description:

The contents of the low-order four bits (Bits 3,2,1 and 0) of the memory location (HL) are copied into 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 into 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 into 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: 5 T STATES: 18(4,4,3,4,3)

Condition Bits Affected:

S: Set if Acc. is negative after operation; reset otherwise
Z: Set if Acc. is zero after operation; reset otherwise
H: Reset
P/V: Set if parity of Acc. is even after operation; reset otherwise
N: Reset
C: 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)

after the execution of

RLD

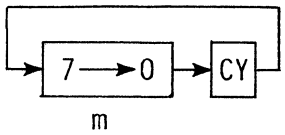
the contents of the Accumulator and memory location 5000H will be

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)

RR m

Operation:

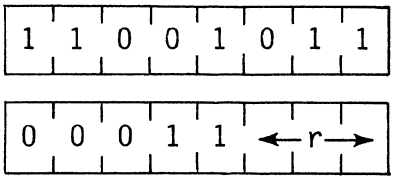


Format:

Opcode	Operands
RR	m

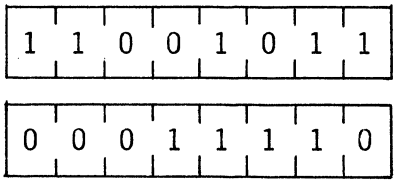
The m operand is any of r, (HL), (IX+d), or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

RR r



CB

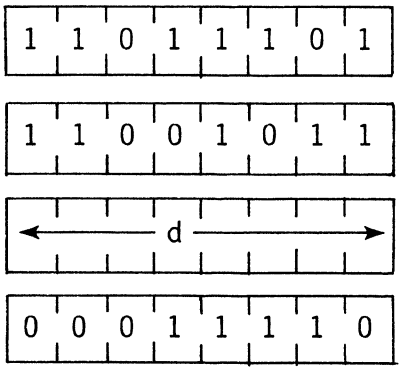
RR (HL)



CB

1E

RR (IX+d)

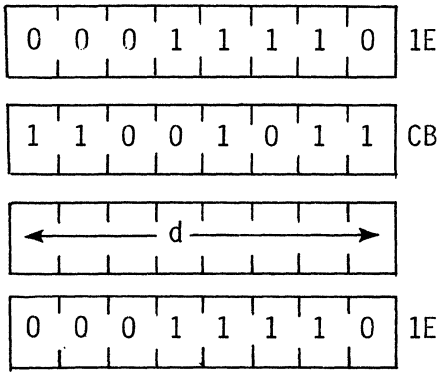


DD

CB

1E

RR (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code 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: the contents of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 7. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES
RR r	2	8(4,4)
RR (HL)	4	15(4,4,4,3)
RR (IX+d)	6	23(4,4,3,5,4,3)
RR (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

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

after the execution of

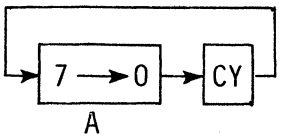
RR (HL)

the contents of location 4343H and the Carry Flag will be

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

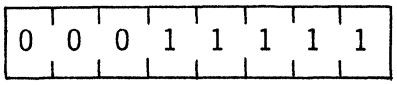
RRA

Operation:



Format:

Opcode	Operands
RRA	



Description:

The contents of the Accumulator (register A) are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the register. The content of bit 0 is copied into the Carry Flag (C flag in register F) and the previous content of the Carry Flag is copied into bit 7. Bit 0 is the least significant bit.

M CYCLES: 1 T STATES: 4

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 0 of Acc.

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

after the execution of

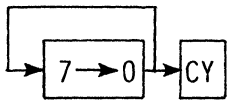
RRA

the contents of the Accumulator and the Carry Flag will be

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

RRCA

Operation:



Format:

Opcode	Operands
RRCA	

0	0	0	0	1	1	1	1	OF

Description:

The contents of the Accumulator (register A) are rotated right: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the register. The content of bit 0 is copied into bit 7 and also into the Carry Flag (C flag in register F.) Bit 0 is the least significant bit.

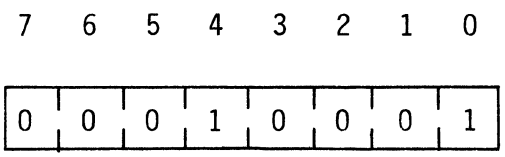
M CYCLES: 1 T STATES: 4

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 0 of Acc.

Example:

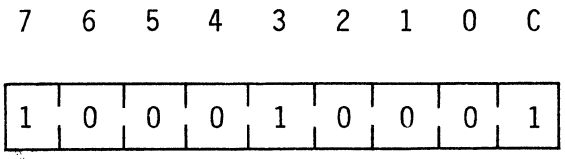
If the contents of the Accumulator are



After the execution of

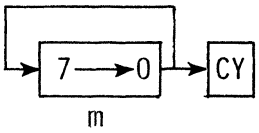
RRCA

the contents of the Accumulator and the Carry Flag will be



RRC m

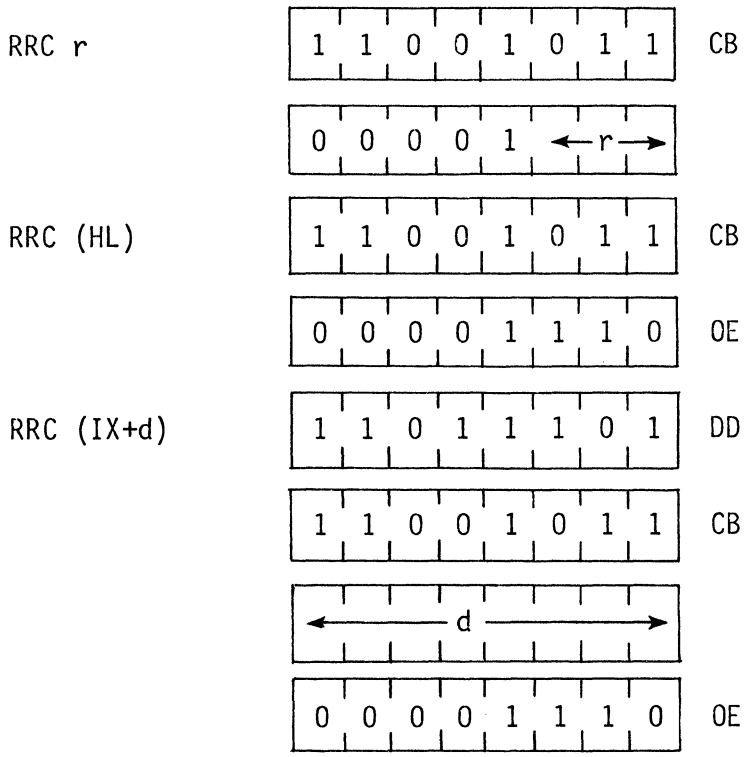
Operation:



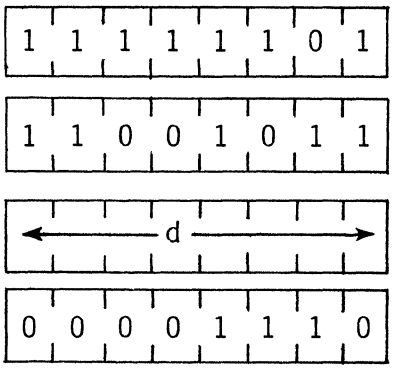
Format:

Opcode	Operands
RRC	m

The m operand is any of r,(HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



RRC (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code 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: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in the F register) and also into bit 7. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES
RRC r	2	8(4,4)
RRC (HL)	4	15(4,4,4,3)
RRC (IX+d)	6	23(4,4,3,5,4,3)
RRC (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

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

after the execution of

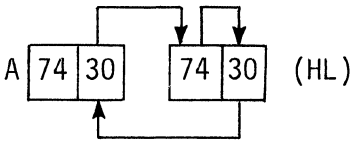
RRC A

the contents of register A and the Carry Flag will be

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

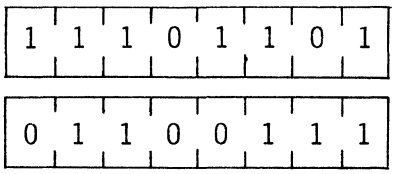
RRD

Operation:



Format:

Opcode	Operands
RRD	



Description:

The contents of the low-order four bits (bits 3,2,1 and 0) of memory location (HL) are copied into the low-order four bits of the Accumulator (register A); the previous contents of the low-order four bits of the Accumulator are copied into 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 into the low-order four bits of (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: 5 T STATES: 18(4,4,3,4,3)

Condition Bits Affected:

S: Set if Acc. is negative after operation; reset otherwise
Z: Set if Acc. is zero after operation; reset other wise
H: Reset
P/V: Set if parity of Acc. is even after operation; reset otherwise
N: Reset
C: 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)

after the execution of

RRD

the contents of the Accumulator and memory location 5000H will be

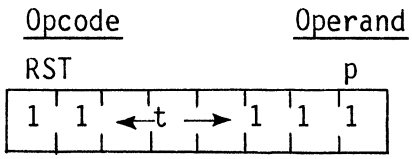
7	6	5	4	3	2	1	0	
1	0	0	0	0	0	0	0	Accumulator

7	6	5	4	3	2	1	0	
0	1	0	0	0	0	1	0	(5000H)

RST p

Operation: $(SP-1) \leftarrow PC_H$, $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$, $PC_L \leftarrow P$

Format:



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 into the PC. Program execution then begins with the opcode 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 into the memory address now pointed to by SP, decrementing SP again, and loading the low-order byte of PC into the address now pointed to by SP. The ResTart instruction allows for a jump to one of eight addresses as shown in the table below. The operand p is assembled into the object code using the corresponding T state. Note: Since 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 into 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: 3 T STATES: 11(5,3,3)

Example:

If the contents of the Program Counter are 15B3H, after the execution of

RST 18H (Object code 1101111)

the PC will contain 0018H, as the address of the next opcode to be fetched.

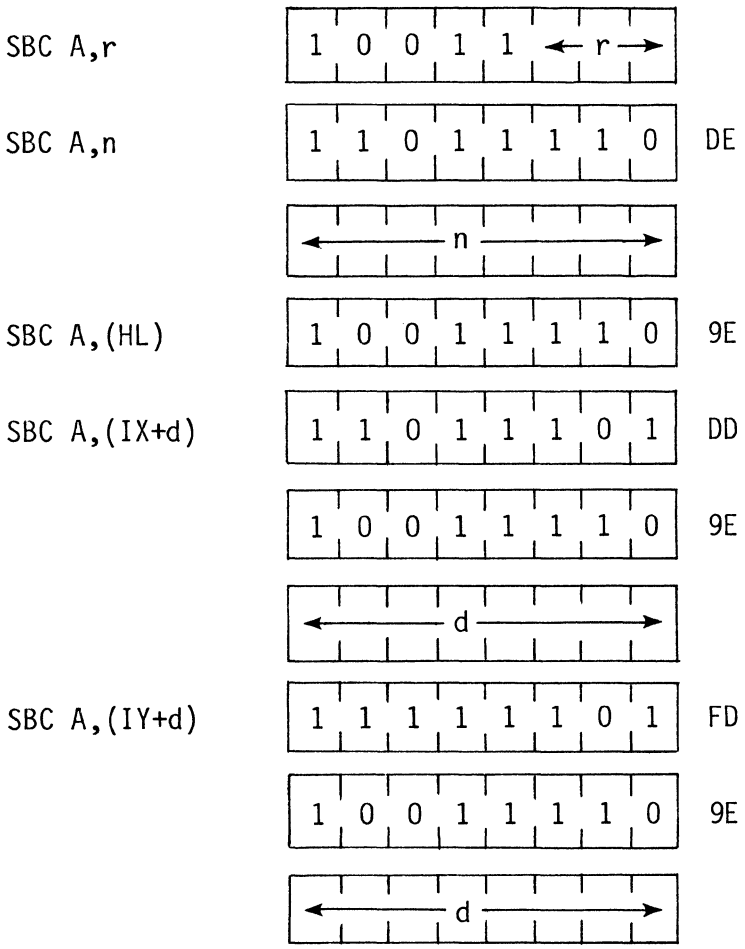
SBC A,s

Operation: A ← A - s - CY

Format:

Opcode	Operands
SBC	A,s

The s operand is any of r,n,(HL),(IX+D) or (IY+d) as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies register 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
SBC A,r	1	4
SBC A,n	2	7(4,3)
SBC A,(HL)	2	7(4,3)
SBC A,(IX+d)	5	19(4,4,3,5,3)
SBC A,(IX+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero, reset otherwise
H: Set if there is a borrow from Bit 4; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if there is a 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, after the execution of

SBC A,(HL)

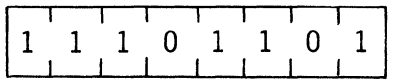
the Accumulator will contain 10H.

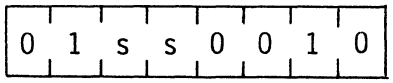
SBC HL,ss

Operation: HL ←HL-ss-CY

Format:

Opcode	Operands
SBC	HL,ss





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: 4 T STATES: 15(4,4,4,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise

Z: Set if result is zero; reset otherwise

H: Set if there is a borrow from Bit 12; reset other wise

P/V: Set if overflow; reset otherwise

N: Set

C: Set it there is a borrow; reset otherwise

Example:

If the contents of the HL register pair are 9999H, the contents of register pai DE are 1111H, and the Carry Flag is set, after the execution of

```
SBC HL,DE
```

the contents of HL will be 8887H.

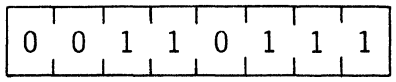
SCF

Operation: CY ←1

Format:

Opcode

SCF



37

Description:

The C flag in the F register is set.

M CYCLES: 1 T STATES: 4

Condition Bits Affected:

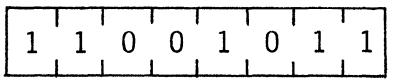
S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Set

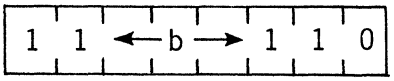
SET b, (HL)

Operation: $(HL)_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,(HL)





Description:

Bit b (any bit, 7 through 0) 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: 4 T STATES: 15(4,4,4,3)

Condition Bits Affected: None

Example:

If the contents of the HL register pair are 3000H, after the execution of

```
SET 4,(HL)
```

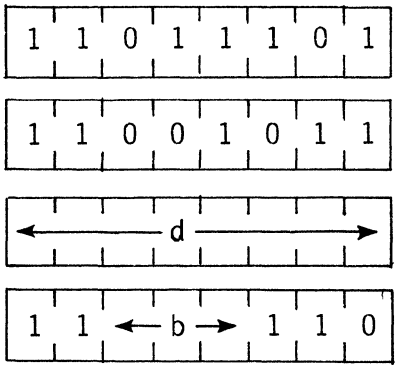
bit 4 in memory location 3000H will be 1. (Bit 0 in memory location 3000H is the least significant bit.)

SET b,(IX+d)

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

Format:

Opcode	Operands
SET	b,(IX+d)



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the sum of the contents of the IX register pair (Index Register IX) 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: 6 T STATES: 23(4,4,3,5,4,3)

Condition Bits Affected: None

Example:

If the contents of Index Register are 2000H, after the execution of

```
SET 0,(IX+3H)
```

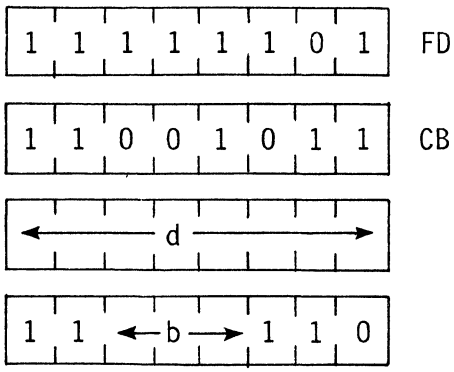
bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

SET b,(IY+d)

Operation: $(IY+d)_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,(IY+d)



Description:

Bit b (any bit, 7 through 0) in the memory location addressed by the sum of the contents of the IY register pair (Index Register IY) 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: 6 T STATES: 23(4,4,3,5,4,3)

Condition Bits Affected: None

Example:

If the contents of Index Register IY are 2000H, after the execution of

```
SET 0,(IY+3H)
```

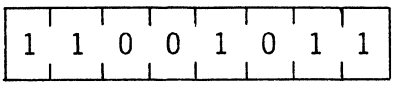
bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

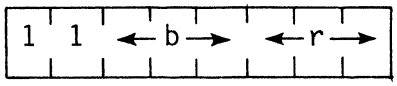
SET b, r

Operation: $r_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,r





Description:

Bit b (any bit, 7 through 0) 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: 2 T STATES: 8(4,4)

Condition Bits Affected: None

Example:

After the execution of

```
SET 4,A
```

bit 4 in register A will be set. (Bit 0 is the least significant bit.)

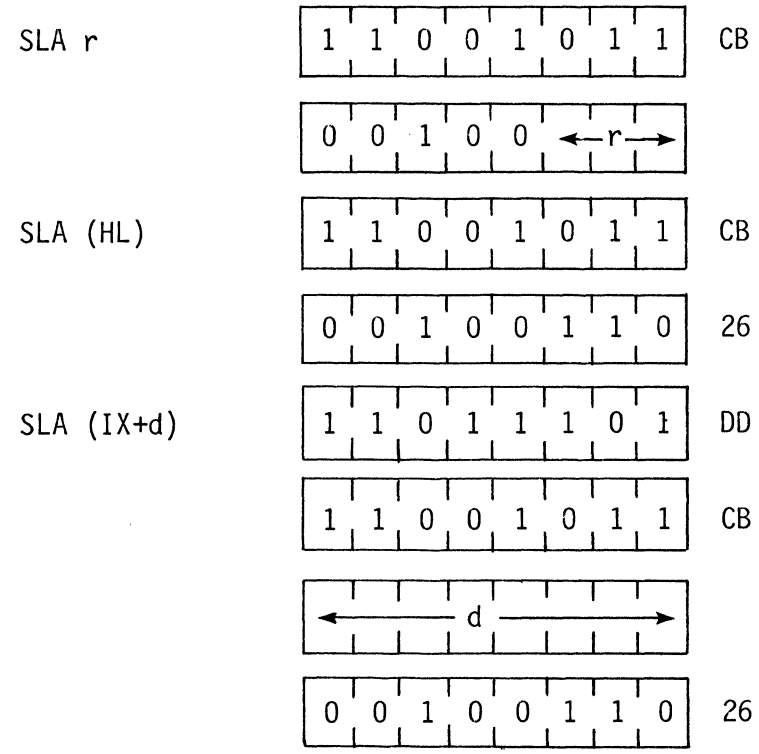
SLA m

Operation: n CY ← 7 ← 0 ← 0

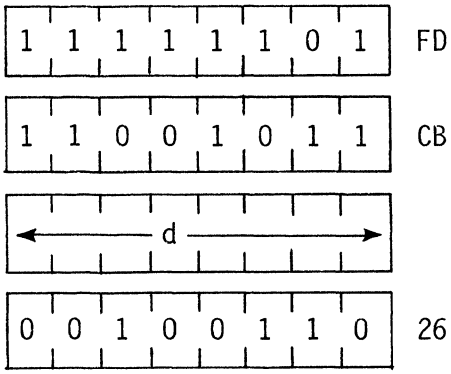
Format:

Opcode	Operands
SLA	m

The operand m is any of r,(HL),(IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



SLA (IY+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:

An arithmetic shift left is performed on the contents of operand m: bit 0 is reset, the previous content of bit 0 is copied into bit 1, the previous content of bit 1 is copied into bit 2; this pattern is continued throughout; the content of bit 7 is copied into the Carry Flag (C flag in register F). Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES
SLA r	2	8(4,4)
SLA (HL)	4	15(4,4,4,3)
SLA (IX+d)	6	23(4,4,3,5,4,3)
SLA (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if result is zero;
reset otherwise

H: Reset

P/V: Set if parity is even;
reset otherwise

N: Reset

C: 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

after the execution of

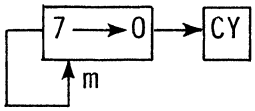
SLA L

the contents of register L and the Carry Flag will be

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

SRA m

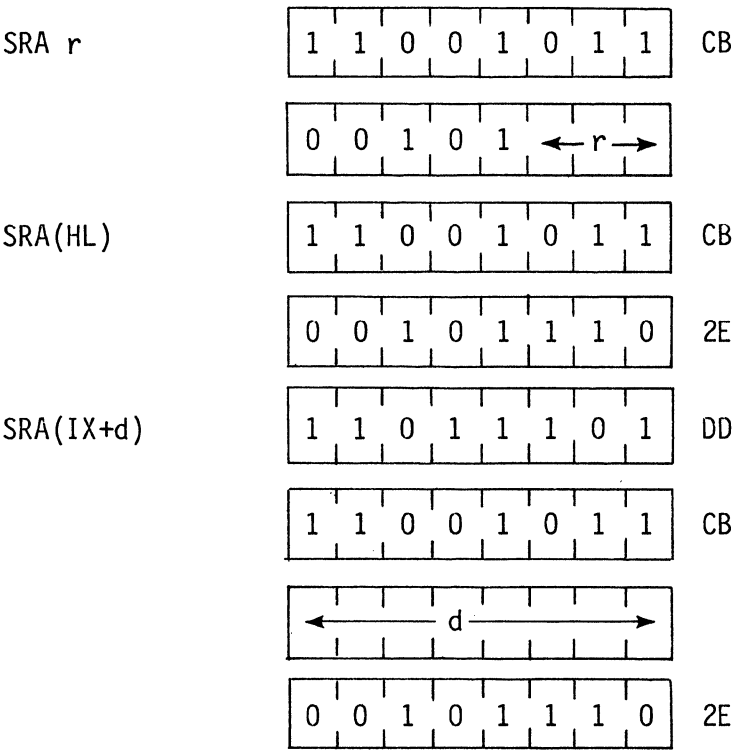
Operation:



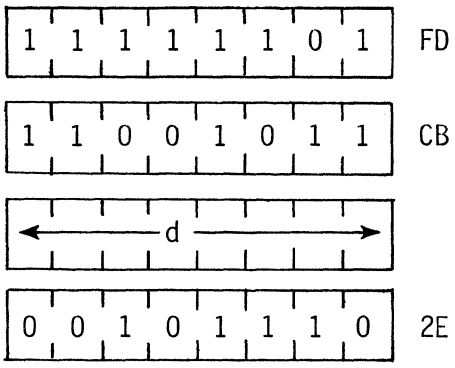
Format:

Opcode	Operands
SRA	m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



SRA(IY+d)



*r means 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

An arithmetic shift right is performed on the contents of operand m: the content of bit 7 is copied into bit 6; the previous content of bit 6 is copied into bit 5; this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag (C flag in register F), and the previous content of bit 7 is unchanged. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES
SRA r	2	8(4, 4)
SRA (HL)	4	15(4,4,4,3)
SRA (IX+d)	6	23(4,4,3,5,4,3
SBR (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if result is zero;
reset otherwise

H: Reset

P/V: Set if parity is even;
reset otherwise

N: Reset

C: 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

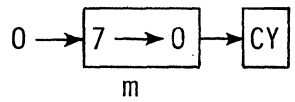
after the execution of

SRA (IX+3H)

the contents of memory location 1003H and the Carry Flag will be

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

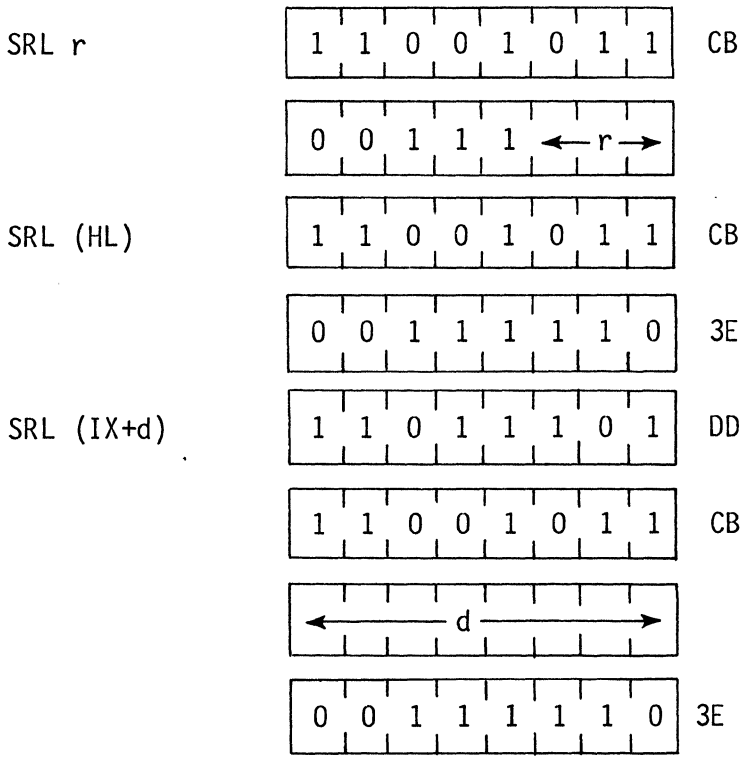
SRL m

Operation: 

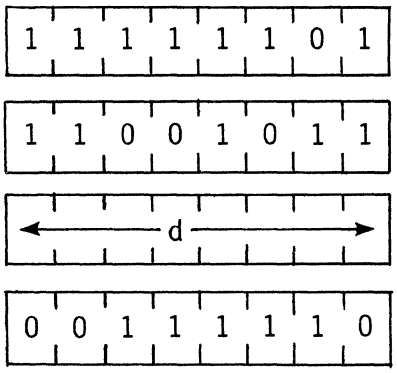
Format:

Opcode	Operands
SRL	m

The operand m is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



SRL (IY+d)



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

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

Description:

The contents of operand m are shifted right: the content of bit 7 is copied into bit 6; the content of bit 6 is copied into bit 5: this pattern is continued throughout the byte. The content of bit 0 is copied into the Carry Flag, and bit 7 is reset. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES
SRL r	2	8(4,4)
SRL (HL)	4	15(4,4,4,3)
SRL (IX+d)	6	23(4,4,3,5,4,3)
SRL (IY+d)	6	23(4,4,3,5,4,3)

Condition Bits Affected:

S: Set if result is negative;
reset otherwise

Z: Set if result is zero;
reset otherwise

H: Reset

P/V: Set if parity is even;
reset otherwise

N: Reset

C: 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

after the execution of

SRL B

the contents of register B and the Carry Flag will be

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

SUB s

Operation: A ←A - s

Format:

Opcode	Operands
SUB	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-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 CYCLES	T STATES
SUB r	1	4
SUB n	2	7(4,3)
SUB (HL)	2	7(4,3)
SUB (IX+d)	5	19(4,4,3,5,3)
SUB (IY+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if there is a borrow from Bit 4; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if there is a borrow; reset otherwise

Example:

If the Accumulator contains 29H and register D contains 11H, after the execution of

```
SUB D
```

the Accumulator will contain 18H.

XOR s

Operation: A ←— A ⊕ s

Format:

Opcode	Operands
XOR	s

The S operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

XOR r	1 0 1 0 1 ←r→	
XOR n	1 1 1 0 1 1 1 0	EE
	←— n —→	
XOR (HL)	1 0 1 0 1 1 1 0	AE
XOR (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 1 0 1 1 1 0	AE
	←— d —→	
XOR (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 1 0 1 1 1 0	AE
	←— 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:

A logical exclusive-OR operation, bit by bit, 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
XOR r	1	4
XOR n	2	7(4,3)
XOR (HL)	2	7(4,3)
XOR (IX+d)	5	19(4,4,3,5,3)
XOR (IY+d)	5	19(4,4,3,5,3)

Condition Bits Affected:

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

Example:

If the Accumulator contains 96H (10010110), after the execution of

XOR 5DH (Note: 5DH = 01011101)

the Accumulator will contain CBH (11001011).

APPENDIX A

ALPHABETICAL LISTING OF Z80 OPCODES

```
  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

                     1          TITLE Z80 OPCODE LISTING
                       ; PSEUDO OPS
                       ;
                     4          NAME OPCODES
0                    5          ORG  0
                     6          PSECT REL
                       ;
0 AA                 8          DEFB 0AAH
     =0001'          9 L2       DEFL $
     =55AA          10 L2       DEFL 55AAH
1 41424344          11          DEFM 'ABCD'
5'                  12 NN       DEFS 2
7 BBAA              13          DEFW 0AABBH
     =AABB          14 L1       EQU  0AABBH
     =0005          15 IND      EQU  5
     =0020          16 N        EQU  20H
     =0030          17 DIS      EQU  30H
                    18          GLOBAL NN
     =0000          19          IF   0
                       ; SHOULD NOT BE ASSEMBLED
                    21          LD   A,B
                    22          ENDIF
     =0001          23          IF   1
                       ; SHOULD BE ASSEMBLED
9 78                25          LD   A,B
                    26          ENDIF
                       ; TURN LISTING OFF
                    31          LIST
                       ; LISTING SHOULD BE ON
                       ;
                       ;
                       ;
                       ; Z80 OPCODES
                       ;
B 8E                38          ADC  A,(HL)
C DD8E05            39          ADC  A,(IX+IND)
F FD8E05            40          ADC  A,(IY+IND)
2 8F                41          ADC  A,A
3 88                42          ADC  A,B
4 89                43          ADC  A,C
5 8A                44          ADC  A,D
6 8B                45          ADC  A,E
7 8C                46          ADC  A,H
8 8D                47          ADC  A,L
9 CE20              48          ADC  A,N
B ED4A              49          ADC  HL,BC
D ED5A              50          ADC  HL,DE
F ED6A              51          ADC  HL,HL
1 ED7A              52          ADC  HL,SP
                       ;
3 86                54          ADD  A,(HL)
4 DD8605            55          ADD  A,(IX+IND)
7 FD8605            56          ADD  A,(IY+IND)
A 87                57          ADD  A,A
B 80                58          ADD  A,B
C 81                59          ADD  A,C
D 82                60          ADD  A,D
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

002E 83             61          ADD  A,E
002F 84             62          ADD  A,H
0030 85             63          ADD  A,L
0031 C620           64          ADD  A,N
0033 09             65          ADD  HL,BC
0034 19             66          ADD  HL,DE
0035 29             67          ADD  HL,HL
0036 39             68          ADD  HL,SP
0037 DD09           69          ADD  IX,BC
0039 DD19           70          ADD  IX,DE
003B DD29           71          ADD  IX,IX
003D DD39           72          ADD  IX,SP
003F FD09           73          ADD  IY,BC
0041 FD19           74          ADD  IY,DE
0043 FD29           75          ADD  IY,IY
0045 FD39           76          ADD  IY,SP
                    ;
0047 A6             78          AND  (HL)
0048 DDA605         79          AND  (IX+IND)
004B FDA605         80          AND  (IY+IND)
004E A7             81          AND  A
004F A0             82          AND  B
0050 A1             83          AND  C
0051 A2             84          AND  D
0052 A3             85          AND  E
0053 A4             86          AND  H
0054 A5             87          AND  L
0055 E620           88          AND  N
                    ;
0057 CB46           90          BIT  0,(HL)
0059 DDCB0546       91          BIT  0,(IX+IND)
005D FDCB0546       92          BIT  0,(IY+IND)
0061 CB47           93          BIT  0,A
0063 CB40           94          BIT  0,B
0065 CB41           95          BIT  0,C
0067 CB42           96          BIT  0,D
0069 CB43           97          BIT  0,E
006B CB44           98          BIT  0,H
006D CB45           99          BIT  0,L
                    ;
006F CB4E          101          BIT  1,(HL)
0071 DDCB054E      102          BIT  1,(IX+IND)
0075 FDCB054E      103          BIT  1,(IY+IND)
0079 CB4F          104          BIT  1,A
007B CB48          105          BIT  1,B
007D CB49          106          BIT  1,C
007F CB4A          107          BIT  1,D
0081 CB4B          108          BIT  1,E
0083 CB4C          109          BIT  1,H
0085 CB4D          110          BIT  1,L
                    ;
0087 CB56          112          BIT  2,(HL)
0089 DDCB0556      113          BIT  2,(IX+IND)
008D FDCB0556      114          BIT  2,(IY+IND)
0091 CB57          115          BIT  2,A
0093 CB50          116          BIT  2,B
0095 CB51          117          BIT  2,C
```

```
 OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

7 CB52             118          BIT  2,D
9 CB53             119          BIT  2,E
B CB54             120          BIT  2,H
D CB55             121          BIT  2,L
                       ;
F CB5E             123          BIT  3,(HL)
1 DDCB055E         124          BIT  3,(IX+IND)
5 FDCB055E         125          BIT  3,(IY+IND)
9 CB5F             126          BIT  3,A
B CB58             127          BIT  3,B
D CB59             128          BIT  3,C
F CB5A             129          BIT  3,D
1 CB5B             130          BIT  3,E
3 CB5C             131          BIT  3,H
5 CB5D             132          BIT  3,L
                       ;
7 CB66             134          BIT  4,(HL)
9 DDCB0566         135          BIT  4,(IX+IND)
D FDCB0566         136          BIT  4,(IY+IND)
1 CB67             137          BIT  4,A
3 CB60             138          BIT  4,B
5 CB61             139          BIT  4,C
7 CB62             140          BIT  4,D
9 CB63             141          BIT  4,E
B CB64             142          BIT  4,H
D CB65             143          BIT  4,L
                       ;
F CB6E             145          BIT  5,(HL)
1 DDCB056E         146          BIT  5,(IX+IND)
5 FDCB056E         147          BIT  5,(IY+IND)
9 CB6F             148          BIT  5,A
B CB68             149          BIT  5,B
D CB69             150          BIT  5,C
F CB6A             151          BIT  5,D
1 CB6B             152          BIT  5,E
3 CB6C             153          BIT  5,H
5 CB6D             154          BIT  5,L
                       ;
7 CB76             156          BIT  6,(HL)
9 DDCB0576         157          BIT  6,(IX+IND)
D FDCB0576         158          BIT  6,(IY+IND)
1 CB77             159          BIT  6,A
3 CB70             160          BIT  6,B
5 CB71             161          BIT  6,C
7 CB72             162          BIT  6,D
9 CB73             163          BIT  6,E
B CB74             164          BIT  6,H
D CB75             165          BIT  6,L
                       ;
F CB7E             167          BIT  7,(HL)
1 DDCB057E         168          BIT  7,(IX+IND)
5 FDCB057E         169          BIT  7,(IY+IND)
9 CB7F             170          BIT  7,A
B CB78             171          BIT  7,B
D CB79             172          BIT  7,C
F CB7A             173          BIT  7,D
1 CB7B             174          BIT  7,E
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

0113 CB7C              175          BIT  7,H
0115 CB7D              176          BIT  7,L
                           ;
0117 DC0500'           178          CALL C,NN
011A FC0500'           179          CALL M,NN
011D D40500'           180          CALL NC,NN
0120 CD0500'           181          CALL NN
0123 C40500'           182          CALL NZ,NN
0126 F40500'           183          CALL P,NN
0129 EC0500'           184          CALL PE,NN
012C E40500'           185          CALL PO,NN
012F CC0500'           186          CALL Z,NN
                           ;
0132 3F                188          CCF
                           ;
0133 BE                190          CP   (HL)
0134 DDBE05            191          CP   (IX+IND)
0137 FDBE05            192          CP   (IY+IND)
013A BF                193          CP   A
013B B8                194          CP   B
013C B9                195          CP   C
013D BA                196          CP   D
013E BB                197          CP   E
013F BC                198          CP   H
0140 BD                199          CP   L
0141 FE20              200          CP   N
                           ;
0143 EDA9              202          CPD
0145 EDB9              203          CPDR
0147 EDA1              204          CPI
0149 EDB1              205          CPIR
                           ;
014B 2F                207          CPL
                           ;
014C 27                209          DAA
                           ;
014D 35                211          DEC  (HL)
014E DD3505            212          DEC  (IX+IND)
0151 FD3505            213          DEC  (IY+IND)
0154 3D                214          DEC  A
0155 05                215          DEC  B
0156 0B                216          DEC  BC
0157 0D                217          DEC  C
0158 15                218          DEC  D
0159 1B                219          DEC  DE
015A 1D                220          DEC  E
015B 25                221          DEC  H
015C 2B                222          DEC  HL
015D DD2B              223          DEC  IX
015F FD2B              224          DEC  IY
0161 2D                225          DEC  L
0162 3B                226          DEC  SP
                           ;
0163 F3                228          DI
                           ;
0164 102E              230          DJNZ DIS
                           ;
```

```
 OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

6 FB                232         EI
                    ;
7 E3                234         EX    (SP),HL
8 DDE3              235         EX    (SP),IX
A FDE3              236         EX    (SP),IY
C 08                237         EX    AF,AF'
D EB                238         EX    DE,HL
E D9                239         EXX
                    ;
F 76                241         HALT
                    ;
0 ED46              243         IM    0
2 ED56              244         IM    1
4 ED5E              245         IM    2
                    ;
6 ED78              247         IN    A,(C)
8 DB20              248         IN    A,(N)
A ED40              249         IN    B,(C)
C ED48              250         IN    C,(C)
E ED50              251         IN    D,(C)
0 ED58              252         IN    E,(C)
2 ED70              253         IN    F,(C)
4 ED60              254         IN    H,(C)
6 ED68              255         IN    L,(C)
                    ;
8 34                257         INC   (HL)
9 FD3405            258         INC   (IY+IND)
C DD3405            259         INC   (IX+IND)
F 3C                260         INC   A
0 04                261         INC   B
1 03                262         INC   BC
2 0C                263         INC   C
3 14                264         INC   D
4 13                265         INC   DE
5 1C                266         INC   E
6 24                267         INC   H
7 23                268         INC   HL
8 DD23              269         INC   IX
A FD23              270         INC   IY
C 2C                271         INC   L
D 33                272         INC   SP
                    ;
E EDAA              274         IND
0 EDBA              275         INDR
2 EDA2              276         INI
4 EDB2              277         INIR
                    ;
6 E9                279         JP    (HL)
7 DDE9              280         JP    (IX)
9 FDE9              281         JP    (IY)
B DA0500'           282         JP    C,NN
E FA0500'           283         JP    M,NN
1 D20500'           284         JP    NC,NN
4 C30500'           285         JP    NN
7 C20500'           286         JP    NZ,NN
A F20500'           287         JP    P,NN
D EA0500'           288         JP    PE,NN
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

01C0 E20500'          289         JP   PO,NN
01C3 CA0500'          290         JP   Z,NN
                          ;
01C6 382E             292         JR   C,DIS
01C8 182E             293         JR   DIS
01CA 302E             294         JR   NC,DIS
01CC 202E             295         JR   NZ,DIS
01CE 282E             296         JR   Z,DIS
                          ;
01D0 02               298         LD   (BC),A
01D1 12               299         LD   (DE),A
01D2 77               300         LD   (HL),A
01D3 70               301         LD   (HL),B
01D4 71               302         LD   (HL),C
01D5 72               303         LD   (HL),D
01D6 73               304         LD   (HL),E
01D7 74               305         LD   (HL),H
01D8 75               306         LD   (HL),L
01D9 3620             307         LD   (HL),N
                          ;
01DB DD7705           309         LD   (IX+IND),A
01DE DD7005           310         LD   (IX+IND),B
01E1 DD7105           311         LD   (IX+IND),C
01E4 DD7205           312         LD   (IX+IND),D
01E7 DD7305           313         LD   (IX+IND),E
01EA DD7405           314         LD   (IX+IND),H
01ED DD7505           315         LD   (IX+IND),L
01F0 DD360520         316         LD   (IX+IND),N
                          ;
01F4 FD7705           318         LD   (IY+IND),A
01F7 FD7005           319         LD   (IY+IND),B
01FA FD7105           320         LD   (IY+IND),C
01FD FD7205           321         LD   (IY+IND),D
0200 FD7305           322         LD   (IY+IND),E
0203 FD7405           323         LD   (IY+IND),H
0206 FD7505           324         LD   (IY+IND),L
0209 FD360520         325         LD   (IY+IND),N
                          ;
020D 320500'          327         LD   (NN),A
0210 ED430500'        328         LD   (NN),BC
0214 ED530500'        329         LD   (NN),DE
0218 220500'          330         LD   (NN),HL
021B DD220500'        331         LD   (NN),IX
021F FD220500'        332         LD   (NN),IY
0223 ED730500'        333         LD   (NN),SP
                          ;
0227 0A               335         LD   A,(BC)
0228 1A               336         LD   A,(DE)
0229 7E               337         LD   A,(HL)
022A DD7E05           338         LD   A,(IX+IND)
022D FD7E05           339         LD   A,(IY+IND)
0230 3A0500'          340         LD   A,(NN)
0233 7F               341         LD   A,A
0234 78               342         LD   A,B
0235 79               343         LD   A,C
0236 7A               344         LD   A,D
0237 7B               345         LD   A,E
```

```
 OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

38 7C                346          LD   A,H
39 ED57              347          LD   A,I
3B 7D                348          LD   A,L
3C 3E20              349          LD   A,N
3E ED5F              350          LD   A,R
                               ;
40 46                352          LD   B,(HL)
41 DD4605            353          LD   B,(IX+IND)
44 FD4605            354          LD   B,(IY+IND)
47 47                355          LD   B,A
48 40                356          LD   B,B
49 41                357          LD   B,C
4A 42                358          LD   B,D
4B 43                359          LD   B,E
4C 44                360          LD   B,H
4D 45                361          LD   B,L
4E 0620              362          LD   B,N
                               ;
50 ED4B0500'         364          LD   BC,(NN)
54 010500'           365          LD   BC,NN
                               ;
57 4E                367          LD   C,(HL)
58 DD4E05            368          LD   C,(IX+IND)
5B FD4E05            369          LD   C,(IY+IND)
5E 4F                370          LD   C,A
5F 48                371          LD   C,B
60 49                372          LD   C,C
61 4A                373          LD   C,D
62 4B                374          LD   C,E
63 4C                375          LD   C,H
64 4D                376          LD   C,L
65 0E20              377          LD   C,N
                               ;
67 56                379          LD   D,(HL)
68 DD5605            380          LD   D,(IX+IND)
6B FD5605            381          LD   D,(IY+IND)
6E 57                382          LD   D,A
6F 50                383          LD   D,B
70 51                384          LD   D,C
71 52                385          LD   D,D
72 53                386          LD   D,E
73 54                387          LD   D,H
74 55                388          LD   D,L
75 1620              389          LD   D,N
                               ;
77 ED5B0500'         391          LD   DE,(NN)
7B 110500'           392          LD   DE,NN
                               ;
7E 5E                394          LD   E,(HL)
7F DD5E05            395          LD   E,(IX+IND)
82 FD5E05            396          LD   E,(IY+IND)
85 5F                397          LD   E,A
86 58                398          LD   E,B
87 59                399          LD   E,C
88 5A                400          LD   E,D
89 5B                401          LD   E,E
8A 5C                402          LD   E,H
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

028B 5D              403          LD   E,L
028C 1E20            404          LD   E,N
                              ;
028E 66              406          LD   H,(HL)
028F DD6605          407          LD   H,(IX+IND)
0292 FD6605          408          LD   H,(IY+IND)
0295 67              409          LD   H,A
0296 60              410          LD   H,B
0297 61              411          LD   H,C
0298 62              412          LD   H,D
0299 63              413          LD   H,E
029A 64              414          LD   H,H
029B 65              415          LD   H,L
029C 2620            416          LD   H,N
                              ;
029E 2A0500'         418          LD   HL,(NN)
02A1 210500'         419          LD   HL,NN
                              ;
02A4 ED47            421          LD   I,A
                              ;
02A6 DD2A0500'       423          LD   IX,(NN)
02AA DD210500'       424          LD   IX,NN
                              ;
02AE FD2A0500'       426          LD   IY,(NN)
02B2 FD210500'       427          LD   IY,NN
                              ;
02B6 6E              429          LD   L,(HL)
02B7 DD6E05          430          LD   L,(IX+IND)
02BA FD6E05          431          LD   L,(IY+IND)
02BD 6F              432          LD   L,A
02BE 68              433          LD   L,B
02BF 69              434          LD   L,C
02C0 6A              435          LD   L,D
02C1 6B              436          LD   L,E
02C2 6C              437          LD   L,H
02C3 6D              438          LD   L,L
02C4 2E20            439          LD   L,N
                              ;
02C6 ED4F            441          LD   R,A
                              ;
02C8 ED7B0500'       443          LD   SP,(NN)
02CC F9              444          LD   SP,HL
02CD DDF9            445          LD   SP,IX
02CF FDF9            446          LD   SP,IY
02D1 310500'         447          LD   SP,NN
                              ;
02D4 EDA8            449          LDD
02D6 EDB8            450          LDDR
02D8 EDA0            451          LDI
02DA EDB0            452          LDIR
                              ;
02DC ED44            454          NEG
                              ;
02DE 00              456          NOP
                              ;
02DF B6              458          OR   (HL)
02E0 DDB605          459          OR   (IX+IND)
```

```
C  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

E3 FDB605            460          OR   (IY+IND)
E6 B7                461          OR   A
E7 B0                462          OR   B
E8 B1                463          OR   C
E9 B2                464          OR   D
EA B3                465          OR   E
EB B4                466          OR   H
EC B5                467          OR   L
ED F620              468          OR   N
                         ;
EF EDBB              470          OTDR
F1 EDB3              471          OTIR
                         ;
F3 ED79              473          OUT  (C),A
F5 ED41              474          OUT  (C),B
F7 ED49              475          OUT  (C),C
F9 ED51              476          OUT  (C),D
FB ED59              477          OUT  (C),E
FD ED61              478          OUT  (C),H
FF ED69              479          OUT  (C),L
01 D320              480          OUT  (N),A
                         ;
03 EDAB              482          OUTD
05 EDA3              483          OUTI
                         ;
07 F1                485          POP  AF
08 C1                486          POP  BC
09 D1                487          POP  DE
0A E1                488          POP  HL
0B DDE1              489          POP  IX
0D FDE1              490          POP  IY
0F F5                491          PUSH AF
10 C5                492          PUSH BC
11 D5                493          PUSH DE
12 E5                494          PUSH HL
13 DDE5              495          PUSH IX
15 FDE5              496          PUSH IY
                         ;
17 CB86              498          RES  0,(HL)
19 DDCB0586          499          RES  0,(IX+IND)
1D FDCB0586          500          RES  0,(IY+IND)
21 CB87              501          RES  0,A
23 CB80              502          RES  0,B
25 CB81              503          RES  0,C
27 CB82              504          RES  0,D
29 CB83              505          RES  0,E
2B CB84              506          RES  0,H
2D CB85              507          RES  0,L
                         ;
2F CB8E              509          RES  1,(HL)
31 DDCB058E          510          RES  1,(IX+IND)
35 FDCB058E          511          RES  1,(IY+IND)
39 CB8F              512          RES  1,A
3B CB88              513          RES  1,B
3D CB89              514          RES  1,C
3F CB8A              515          RES  1,D
41 CB8B              516          RES  1,E
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

0343 CB8C              517          RES  1,H
0345 CB8D              518          RES  1,L
                           ;
0347 CB96              520          RES  2,(HL)
0349 DDCB0596          521          RES  2,(IX+IND)
034D FDCB0596          522          RES  2,(IY+IND)
0351 CB97              523          RES  2,A
0353 CB90              524          RES  2,B
0355 CB91              525          RES  2,C
0357 CB92              526          RES  2,D
0359 CB93              527          RES  2,E
035B CB94              528          RES  2,H
035D CB95              529          RES  2,L
                           ;
035F CB9E              531          RES  3,(HL)
0361 DDCB059E          532          RES  3,(IX+IND)
0365 FDCB059E          533          RES  3,(IY+IND)
0369 CB9F              534          RES  3,A
036B CB98              535          RES  3,B
036D CB99              536          RES  3,C
036F CB9A              537          RES  3,D
0371 CB9B              538          RES  3,E
0373 CB9C              539          RES  3,H
0375 CB9D              540          RES  3,L
                           ;
0377 CBA6              542          RES  4,(HL)
0379 DDCB05A6          543          RES  4,(IX+IND)
037D FDCB05A6          544          RES  4,(IY+IND)
0381 CBA7              545          RES  4,A
0383 CBA0              546          RES  4,B
0385 CBA1              547          RES  4,C
0387 CBA2              548          RES  4,D
0389 CBA3              549          RES  4,E
038B CBA4              550          RES  4,H
038D CBA5              551          RES  4,L
                           ;
038F CBAE              553          RES  5,(HL)
0391 DDCB05AE          554          RES  5,(IX+IND)
0395 FDCB05AE          555          RES  5,(IY+IND)
0399 CBAF              556          RES  5,A
039B CBA8              557          RES  5,B
039D CBA9              558          RES  5,C
039F CBAA              559          RES  5,D
03A1 CBAB              560          RES  5,E
03A3 CBAC              561          RES  5,H
03A5 CBAD              562          RES  5,L
                           ;
03A7 CBB6              564          RES  6,(HL)
03A9 DDCB05B6          565          RES  6,(IX+IND)
03AD FDCB05B6          566          RES  6,(IY+IND)
03B1 CBB7              567          RES  6,A
03B3 CBB0              568          RES  6,B
03B5 CBB1              569          RES  6,C
03B7 CBB2              570          RES  6,D
03B9 CBB3              571          RES  6,E
03BB CBB4              572          RES  6,H
03BD CBB5              573          RES  6,L
```

```
C  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

                        ;
BF CBBE             575         RES  7,(HL)
C1 DDCB05BE         576         RES  7,(IX+IND)
C5 FDCB05BE         577         RES  7,(IY+IND)
C9 CBBF             578         RES  7,A
CB CBB8             579         RES  7,B
CD CBB9             580         RES  7,C
CF CBBA             581         RES  7,D
D1 CBBB             582         RES  7,E
D3 CBBC             583         RES  7,H
D5 CBBD             584         RES  7,L
                        ;
D7 C9               586         RET
D8 D8               587         RET  C
D9 F8               588         RET  M
DA D0               589         RET  NC
DB C0               590         RET  NZ
DC F0               591         RET  P
DD E8               592         RET  PE
DE E0               593         RET  PO
DF C8               594         RET  Z
                        ;
E0 ED4D             596         RETI
E2 ED45             597         RETN
                        ;
E4 CB16             599         RL   (HL)
E6 DDCB0516         600         RL   (IX+IND)
EA FDCB0516         601         RL   (IY+IND)
EE CB17             602         RL   A
F0 CB10             603         RL   B
F2 CB11             604         RL   C
F4 CB12             605         RL   D
F6 CB13             606         RL   E
F8 CB14             607         RL   H
FA CB15             608         RL   L
                        ;
FC 17               610         RLA
                        ;
FD CB06             612         RLC  (HL)
FF DDCB0506         613         RLC  (IX+IND)
03 FDCB0506         614         RLC  (IY+IND)
07 CB07             615         RLC  A
09 CB00             616         RLC  B
0B CB01             617         RLC  C
0D CB02             618         RLC  D
0F CB03             619         RLC  E
11 CB04             620         RLC  H
13 CB05             621         RLC  L
                        ;
15 07               623         RLCA
                        ;
16 ED6F             625         RLD
                        ;
18 CB1E             627         RR   (HL)
1A DDCB051E         628         RR   (IX+IND)
1E FDCB051E         629         RR   (IY+IND)
22 CB1F             630         RR   A
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

0424 CB18              631          RR   B
0426 CB19              632          RR   C
0428 CB1A              633          RR   D
042A CB1B              634          RR   E
042C CB1C              635          RR   H
042E CB1D              636          RR   L
                           ;
0430 1F                638          RRA
                           ;
0431 CB0E              640          RRC  (HL)
0433 DDCB050E          641          RRC  (IX+IND)
0437 FDCB050E          642          RRC  (IY+IND)
043B CB0F              643          RRC  A
043D CB08              644          RRC  B
043F CB09              645          RRC  C
0441 CB0A              646          RRC  D
0443 CB0B              647          RRC  E
0445 CB0C              648          RRC  H
0447 CB0D              649          RRC  L
                           ;
0449 0F                651          RRCA
                           ;
044A ED67              653          RRD
                           ;
044C C7                655          RST  0
044D CF                656          RST  08H
044E D7                657          RST  10H
044F DF                658          RST  18H
0450 E7                659          RST  20H
0451 EF                660          RST  28H
0452 F7                661          RST  30H
0453 FF                662          RST  38H
                           ;
0454 9E                664          SBC  A,(HL)
0455 DD9E05            665          SBC  A,(IX+IND)
0458 FD9E05            666          SBC  A,(IY+IND)
045B 9F                667          SBC  A,A
045C 98                668          SBC  A,B
045D 99                669          SBC  A,C
045E 9A                670          SBC  A,D
045F 9B                671          SBC  A,E
0460 9C                672          SBC  A,H
0461 9D                673          SBC  A,L
0462 DE20              674          SBC  A,N
                           ;
0464 ED42              676          SBC  HL,BC
0466 ED52              677          SBC  HL,DE
0468 ED62              678          SBC  HL,HL
046A ED72              679          SBC  HL,SP
                           ;
046C 37                681          SCF
                           ;
046D CBC6              683          SET  0,(HL)
046F DDCB05C6          684          SET  0,(IX+IND)
0473 FDCB05C6          685          SET  0,(IY+IND)
0477 CBC7              686          SET  0,A
0479 CBC0              687          SET  0,B
```

```
C  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

7B CBC1             688         SET  0,C
7D CBC2             689         SET  0,D
7F CBC3             690         SET  0,E
81 CBC4             691         SET  0,H
83 CBC5             692         SET  0,L
                        ;
85 CBCE             694         SET  1,(HL)
87 DDCB05CE         695         SET  1,(IX+IND)
8B FDCB05CE         696         SET  1,(IY+IND)
8F CBCF             697         SET  1,A
91 CBC8             698         SET  1,B
93 CBC9             699         SET  1,C
95 CBCA             700         SET  1,D
97 CBCB             701         SET  1,E
99 CBCC             702         SET  1,H
9B CBCD             703         SET  1,L
                        ;
9D CBD6             705         SET  2,(HL)
9F DDCB05D6         706         SET  2,(IX+IND)
A3 FDCB05D6         707         SET  2,(IY+IND)
A7 CBD7             708         SET  2,A
A9 CBD0             709         SET  2,B
AB CBD1             710         SET  2,C
AD CBD2             711         SET  2,D
AF CBD3             712         SET  2,E
B1 CBD4             713         SET  2,H
B3 CBD5             714         SET  2,L
                        ;
B5 CBDE             716         SET  3,(HL)
B7 DDCB05DE         717         SET  3,(IX+IND)
BB FDCB05DE         718         SET  3,(IY+IND)
BF CBDF             719         SET  3,A
C1 CBD8             720         SET  3,B
C3 CBD9             721         SET  3,C
C5 CBDA             722         SET  3,D
C7 CBDB             723         SET  3,E
C9 CBDC             724         SET  3,H
CB CBDD             725         SET  3,L
                        ;
CD CBE6             727         SET  4,(HL)
CF DDCB05E6         728         SET  4,(IX+IND)
D3 FDCB05E6         729         SET  4,(IY+IND)
D7 CBE7             730         SET  4,A
D9 CBE0             731         SET  4,B
DB CBE1             732         SET  4,C
DD CBE2             733         SET  4,D
DF CBE3             734         SET  4,E
E1 CBE4             735         SET  4,H
E3 CBE5             736         SET  4,L
                        ;
E5 CBEE             738         SET  5,(HL)
E7 DDCB05EE         739         SET  5,(IX+IND)
EB FDCB05EE         740         SET  5,(IY+IND)
EF CBEF             741         SET  5,A
F1 CBE8             742         SET  5,B
F3 CBE9             743         SET  5,C
F5 CBEA             744         SET  5,D
```

```
LOC  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

04F7 CBEB             745         SET  5,E
04F9 CBEC             746         SET  5,H
04FB CBED             747         SET  5,L
                          ;
04FD CBF6             749         SET  6,(HL)
04FF DDCB05F6         750         SET  6,(IX+IND)
0503 FDCB05F6         751         SET  6,(IY+IND)
0507 CBF7             752         SET  6,A
0509 CBF0             753         SET  6,B
050B CBF1             754         SET  6,C
050D CBF2             755         SET  6,D
050F CBF3             756         SET  6,E
0511 CBF4             757         SET  6,H
0513 CBF5             758         SET  6,L
                          ;
0515 CBFE             760         SET  7,(HL)
0517 DDCB05FE         761         SET  7,(IX+IND)
051B FDCB05FE         762         SET  7,(IY+IND)
051F CBFF             763         SET  7,A
0521 CBF8             764         SET  7,B
0523 CBF9             765         SET  7,C
0525 CBFA             766         SET  7,D
0527 CBFB             767         SET  7,E
0529 CBFC             768         SET  7,H
052B CBFD             769         SET  7,L
                          ;
052D CB26             771         SLA  (HL)
052F DDCB0526         772         SLA  (IX+IND)
0533 FDCB0526         773         SLA  (IY+IND)
0537 CB27             774         SLA  A
0539 CB20             775         SLA  B
053B CB21             776         SLA  C
053D CB22             777         SLA  D
053F CB23             778         SLA  E
0541 CB24             779         SLA  H
0543 CB25             780         SLA  L
                          ;
0545 CB2E             782         SRA  (HL)
0547 DDCB052E         783         SRA  (IX+IND)
054B FDCB052E         784         SRA  (IY+IND)
054F CB2F             785         SRA  A
0551 CB28             786         SRA  B
0553 CB29             787         SRA  C
0555 CB2A             788         SRA  D
0557 CB2B             789         SRA  E
0559 CB2C             790         SRA  H
055B CB2D             791         SRA  L
                          ;
055D CB3E             793         SRL  (HL)
055F DDCB053E         794         SRL  (IX+IND)
0563 FDCB053E         795         SRL  (IY+IND)
0567 CB3F             796         SRL  A
0569 CB38             797         SRL  B
056B CB39             798         SRL  C
056D CB3A             799         SRL  D
056F CB3B             800         SRL  E
0571 CB3C             801         SRL  H
```

```
C  OBJ.CODE     STMT-NR SOURCE-STMT PASS2 OPCODE OPCODE OPCODE REL

73 CB3D             802         SRL  L
                        ;
75 96               804         SUB  (HL)
76 DD9605           805         SUB  (IX+IND)
79 FD9605           806         SUB  (IY+IND)
7C 97               807         SUB  A
7D 90               808         SUB  B
7E 91               809         SUB  C
7F 92               810         SUB  D
80 93               811         SUB  E
81 94               812         SUB  H
82 95               813         SUB  L
83 D620             814         SUB  N
                        ;
85 AE               816         XOR  (HL)
86 DDAE05           817         XOR  (IX+IND)
89 FDAE05           818         XOR  (IY+IND)
8C AF               819         XOR  A
8D A8               820         XOR  B
8E A9               821         XOR  C
8F AA               822         XOR  D
90 AB               823         XOR  E
91 AC               824         XOR  H
92 AD               825         XOR  L
93 EE20             826         XOR  N
                        ;
95                  828         END
```

APPENDIX B

MOSTEK ASSEMBLER STANDARD PSEUDO-OPS

APPENDIX B

MOSTEK ASSEMBLER STANDARD PSEUDO-OPS

B-1. INTRODUCTION.

B-2. The following pseudo-ops are standard for Z80 assemblers from MOSTEK. Note that other pseudo-ops may be allowed depending on the features of a particular assembler. For example, additional pseudo-ops may be required to handle conditional assembly, global symbols, and macros.

DEFB n

Define byte of memory.

Operation: (PC) <- m (static)

Format

Opcode	Operands	Machine Code
DEFB	n	← n →

(no execution time)

Description: This pseudo-op reserves and defines one byte of memory to contain the value n.

Example:

```
DEFB   OAH
```

causes the current memory location to be defined with the value OAH.

label DEFL nn

Define 'label' to have the value nn.

Operation: label ← nn

Format:

	Opcode	Operand
label	DEFL	nn

(no execution time , no machine code)

Description: This pseudo-op assigns the value nn to the label which appears in the label field. The same label can be defined any number of times in a program using this pseudo-op.

Example:

```
LAB4:    DEFL    050AH
```

The label 'LAB4' is defined to have the value 050AH.

DEFM 's'

Define message

Operation: (PC) $\leftarrow S_1$
(PC+1) $\leftarrow S_2$
(PC+2) $\leftarrow S_3$
.
.
.

where s_1 is the first ASCII character in string s, s_2 is the second ASCII character, etc.

Format:

Opcode	Operand	Machine Code
DEFM	's'	s_1
		s_2
		s_3
		.
		.
		.

(no execution time)

Description: This pseudo-op reserves and defines sequential bytes of memory to contain ASCII equivalents of the characters in the string s.

Example:

```
DEFM    'ABC'
```

will reserve 3 bytes of memory and cause them to be loaded with 41H, 42H,43H, respectively.

DEFS nn

Define storage

Operation: (PC) ⟵ (PC) + nn (static)

Format

Opcode	Operand	Machine code
DEFS	nn	

(no execution time)

Description: This pseudo-op causes nn bytes of memory to be defined as storage. In the object module, these bytes are not loaded. In a load (binary) module these bytes are loaded with meaningless data.

Example:

```
DEFS    40D
```

This causes 40 (decimal) memory locations to be defined as storage and skipped in the object module.

DEFW nn

Define word of memory

Operation: (PC) ← nn = 1
(PC+1) ← nn (static)

Format

Opcode	Operand	Machine code	
DEFW	nn	nn + 1	(Lower byte)
		nn	(Upper byte)

Description: This pseudo-op reserves and defines two bytes of memory. The fir byte is defined to contain the least significant byte of the operand n The next byte is defined to contain the most significant byte of t operand nn.

Example:

```
DEFW        0A00H
```

will define the current memory location to contain 00H and the ne memory location to contain 0AH

END s

End of assembly

Operation: terminates current assembler pass.

Format

Opcode	Operand
END	s

(no execution time, no machine code)

Description: This pseudo-op terminates the current assember pass. The operands s is optional and is an expression which defines the starting execution address of the program being assembled. The value of s is entered in the end-of-file record in the object output of the assembler.

Example:

```
END   OAAH
```

terminates the current assembler pass and causes OAAH to be defined as the starting address of the program.

label EQU nn

Equate 'label' to value nn.

Operation: label nn

Format:

	Opcode	Operand
label	EQU	nn

(no execution time, no machine code)

Description: This pseudo-op assigns the value nn to the label which appears in the label field. The label can only appear once in the label field in a program using this pseudo-op.

Example:

```
LAB4:     EQU    05H
```

The label 'LAB4' is defined to have the value 05H.

APPENDIX C

MOSTEK STANDARD Z80 OBJECT CODE FORMAT

APPENDIX C

MOSTEK STANDARD Z80 OBJECT OUTPUT DEFINITION

C-1. INTRODUCTION.

C-2. Each record of an object module begins with a delimiter (colon of dollar sign) and ends with carriage return and line feed. A colon (:) is used for data records and end-of-file record. A dollar sign ($) is used for records containing relocation information and linking information. An Intel loader will ignore such information and allow loading non-relocatable, non-linkable programs. All information is in ASCII.

C-3. Each record is identified as a type . The type appears in the 8th and 9th bytes of the record and can take the following values:

00 - data record
01 - end-of-file
02 - internal symbol
03 - external symbol
04 - relocation information
05 - module definition

C-4. DATA RECORD FORMAT (TYPE 00).

Byte 1 Colon (:) delimiter

2-3 Number of binary bytes of data in this record. The maximum is 32 binary bytes (64 ASCII bytes).

4-5 Most significant byte of the start address of data.

6-7 Least significant byte of start address of data.

8-9 ASCII zeros. This is the "record type" for data.

10- Data bytes.

Last two bytes - Checksum of all bytes except the delimiter, carriage return, and line feed. The checksum is the negative of the binary sum of all bytes in the record.

CRLF Carriage return, line feed.

C-5. **END-OF-FILE RECORD (TYPE 01).**

Byte 1 Colon (:) delimiter.

2-3 ASCII zeros.

4-5 Most significant byte of the transfer address of the program. This transfer address appears as an argument in the 'END' pseudo-op of a program. It represents the starting execution address of the program.

6-7 Least significant byte of the transfer address.

8-9 Record type 01.

10-11 Checksum.

CRLF Carriage return, line feed.

C-6. **INTERNAL SYMBOL RECORD (TYPE 02).**

Byte 1 Dollar sign ($) delimiter.

2-7 Up to 6 ASCII characters of the internal symbol name. The name is left-justified, blank filled.

8-9 Record type 02.

10-13 Address of the internal symbol, most significant byte first.

14-15 Binary checksum. Note that the ASCII letters of the symbol are converted to binary before the checksum is calculated. Binary conversion is done without regard to errors.

CRLF Carriage return, line feed.

C-7. EXTERNAL SYMBOL RECORD (TYPE 03).

Byte 1 Dollar sign ($) delimiter.

2-7 Up to 6 ASCII characters of the external symbol name. The name is left justified, blank filled.

8-9 Record type 03.

10-13 Last address which uses the external symbol. This is the start of a link list in the object data records which is described below. The most significant byte is first.

14-15 Binary checksum.

CRLF Carriage return, line feed.

C-8. The ASMB-80 Assembler outputs the external symbol name and the last address in the program where the symbol is used. The data records which follow contain a link list pointing to all occurrences of that symbol in the object code.

1. The external symbol record shows the symbol ('LAB') and the last location in the program which uses the symbol (212AH).

2. The object code at 212AH has a pointer which shows where the previous reference to the external symbol occurred (200FH).

3. This backward reference list continues until a terminator ends the list. This terminator is OFFFFH.

This method is easy to generate and decode. It has the advantage of reducing the number of bytes of object code needed to define all external references in a program.

C-9. RELOCATING INFORMATION RECORD (TYPE 04).

The addresses in the program which must be relocated are explicitly defined in these records. Up to 16 addresses (64 ASCII characters) may be defined in each record.

Byte 1 Dollar sign ($) delimiter.

2-3 Number of sets of 2 ASCII characters, where 2 sets define an address.

4-7 ASCII zeros.

8-9 Record type 04.

10- Addresses which must be relocated, most significant byte first.

Last two bytes - Binary checksum.

CRLF Carriage return, line feed.

C-10. MODULE DEFINITION RECORD (TYPE 05).

This record has the name of the module (defined by the 'NAME' pseudo-op) and a loading information flag byte. The flag byte is determined by the 'PSECT' pseudo-op.

Byte 1 Dollar sign ($) delimiter.

2-7 Name of the module, left-justified, blank filled.

8-9 Record type 05.

10-11 Flag byte. When converted to binary, the flag byte is defined as follows:

Bit 0 - 0 for absolute assemblies
1 for relocatable assemblies

12-13 Binary checksum.

CRLF Carriage return, line feed.

APPENDIX D

REFERENCE TABLES

TABLE D-1. Hexadecimal to Decimal Conversion Table

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,408	8	128	8	8
9	9,437,184	9	589,82r	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
0 1 2 3		4 5 6 7		0 1 2 3		4 5 6 7		0 1 2 3		4 5 6 7	
BYTE				BYTE				BYTE			

TABLE D-2. ASCII Character Set (7-Bit Code)

Table D-3. Powers of 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

TABLE D-4.
Powers of 2/Powers of 16 Conversion

2^0	$= 16^0$
2^4	$= 16^1$
2^8	$= 16^2$
2^{12}	$= 16^3$
2^{16}	$= 16^4$
2^{20}	$= 16^5$
2^{24}	$= 16^6$
2^{28}	$= 16^7$
2^{32}	$= 16^8$
2^{36}	$= 16^9$
2^{40}	$= 16^{10}$
2^{44}	$= 16^{11}$
2^{48}	$= 16^{12}$
2^{52}	$= 16^{13}$
2^{56}	$= 16^{14}$
2^{60}	$= 16^{15}$

TABLE D-5. Powers of 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

MOSTEK®
Z80·F8 Covering the full spectrum of
3870 microcomputer applications.

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214/242-0444
In Europe, Contact: MOSTEK Brussels
150 Chaussee de la Hulpe, B1170, Belgium;
Telephone: (32) 02/660-2568/4713

PRINTED IN USA October 1978
Publication No. MK78515