64000 LOGIC DEVELOPMENT SYSTEM

ASSEMBLER SUPPLEMENT Z80

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its options, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service. Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

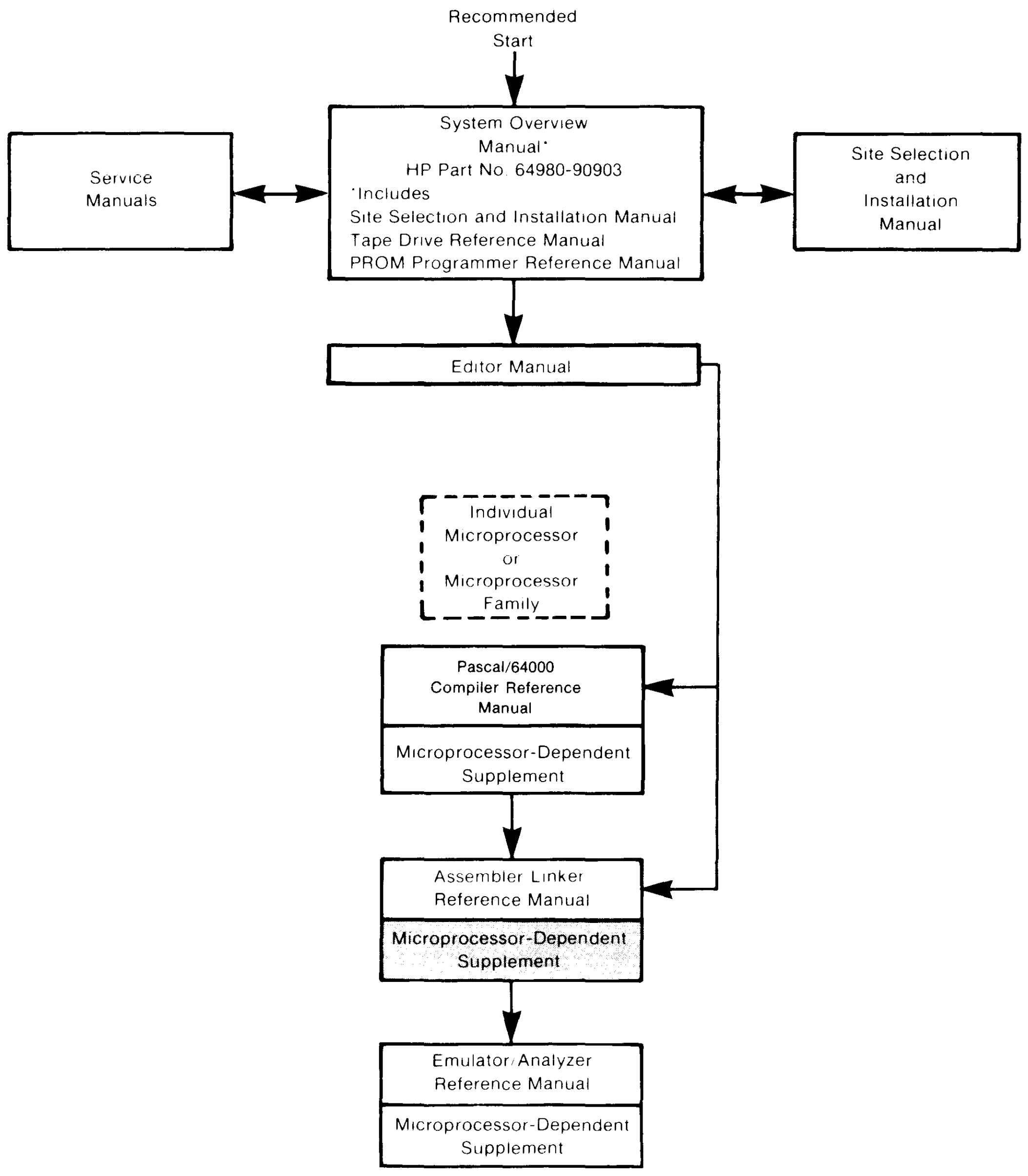
CW&A 9/79

Model 64000 Reference Manuals

The following block diagram shows the documentation scheme for the HP Model 64000 Logic Development System. The interconnecting arrows show the recommended progression through the manuals as a way of gaining familiarity with the system.

For a detailed map showing specific manuals and their part numbers, refer to the Manual Map in the System Overview Manual.

Manual Map



Recommended Start
System Overview Manual*
HP Part No. 64980-90903
*Includes
Site Selection and Installation Manual
Tape Drive Reference Manual
PROM Programmer Reference Manual
Service Manuals
Site Selection and Installation Manual
Editor Manual
Individual Microprocessor or Microprocessor Family
Pascal/64000 Compiler Reference Manual
Microprocessor-Dependent Supplement
Assembler Linker Reference Manual
Microprocessor-Dependent Supplement
Emulator/Analyzer Reference Manual
Microprocessor-Dependent Supplement

Printing History

Each new edition of this manual incorporates all material updated since the previous edition. Each new or revised page is indicated by a revision (rev) date. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The part number on the back cover changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions.

First Printing . . . March 1980
Second Printing . . . September 1980

Z80 Assembler Supplement

HP Model 64000 Logic Development System

Table of Contents

Chapter 1: General Information
Introduction 1-1
Programming Considerations 1-1
Z80 Architecture 1-1
Accumulator and Flag Registers 1-1
General Purpose Registers 1-1
Program Counter Register 1-2
Index Registers 1-2
Stack Pointer Register 1-2
Interrupt Page Address Register 1-2
Memory Refresh Register 1-2
Modes of Addressing 1-2
Immediate Addressing 1-3
Immediate Extended Addressing 1-3
Relative Addressing 1-3
Extended Addressing 1-3
Indexed Addressing 1-4
Register Addressing 1-4
Register Indirect Addressing 1-4
Implied Addressing 1-4
Bit Addressing 1-4
Modified Page Zero Addressing 1-4
Condition Flags 1-5
Zero (Z) Flag 1-5
Sign (S) Flag 1-5
Parity/Overflow (P/V) Flag 1-5
Carry (CY) Flag 1-5
Half Carry (HC) Flag 1-5
Add/Subtract (N) Flag 1-5

Chapter 2: Operand Rules and Conventions
Types of Information 2-1
Identifying Types of Information 2-2

Chapter 3: Special Pseudo Instructions
Introduction 3-1
DEFB Instruction 3-2
DEFS Instruction 3-3
DEFW Instruction 3-4

Table of Contents (Cont'd)

Chapter 4: Assembler Output Listing
General 4-1
Input/Output Files 4-1
  Source Input File 4-1
  Assembler Output Files 4-1
Output Listing 4-2

Chapter 5: Z80 Instruction Set Summary
General 5-1
Predefined Symbols 5-3

List of Tables

4-1. Source Program Format Example 4-3
4-2. Assembler Output Listing 4-4
4-3. Assembler Output with Errors 4-6

Chapter 1

General Information (Z80)

Introduction

This chapter contains general information about the Z80 microprocessor. It briefly discusses the microprocessor's architecture, addressing modes, and condition codes. For a detailed description of the microprocessor refer to the manufacturer's Z80 User's Manual.

NOTE

If you are unfamiliar with assembly language or assemblers, read Chapter 6 in the Assembler/Linker Reference Manual. That chapter briefly reviews assemblers, assembly language, and the numbering systems.

Programming Considerations

Z80 Architecture

There are eighteen 8-bit registers and four 16-bit registers available in the Z80 microprocessor for control of external memory and peripheral devices that may be associated with the target system. These registers are discussed briefly in the following paragraphs.

Accumulator and Flag Registers

The microprocessor contains two independent 8-bit registers (accumulators) with associated 8-bit flag registers. They are referred to as register A and register A′. The F flag register is associated with the A register and the F′ flag register is associated with the A′ register. The A or A′ register holds the result of 8-bit arithmetic or logical operations while the flag register (F or F′) indicates the specific effect of those operations.

General Purpose Registers

There are two sets of general purpose registers with each set containing six 8-bit registers that may be addressed individually or as 16-bit register pairs. One set contains register pairs BC, DE, and HL while the second set contains register pairs BC′, DE′, and HL′.

Program Counter Register

The program counter (PC) is a two-byte (16-bit) register that points to the current program address. After each instruction the program counter will be automatically incremented. When a program branch is required, the branch address will be placed into the program counter, overriding the incremental operation.

Index Registers

The two index registers (IX and IY) are special-purpose 16-bit registers that may be used as pointers to memory locations where data may be retrieved or stored. When using the indexed addressing mode, an additional byte may be included in the instructions to indicate an offset from the register content. The offset must be specified as a two's complement signed integer.

Stack Pointer Register

The stack pointer (SP) register is another special-purpose 16-bit register that allows the microprocessor to use a section of external memory as a last-in, first-out (LIFO) file. Data may be pushed onto the stack or pulled off the stack by using the PUSH and POP instructions.

Interrupt Page Address Register

The microprocessor may be operated whereby an indirect call to any memory location can be accomplished in response to an interrupt. The interrupt address (I) register may be used to store the high-order 8 bits of the indirect address while the interrupting device provides the low-order 8 bits.

Memory Refresh Register

The microprocessor has a memory refresh register that allows dynamic memories to be used in the same fashion as static memories. The data in the refresh counter will be sent out on the address bus along with a refresh control signal while the microprocessor is executing an instruction.

Modes of Addressing

A number of instructions, such as the "compare" instruction, require data to be operated on. The method of specifing or locating the data is generally referred to as the mode of addressing. The Z80 microprocessor uses ten addressing modes to reference the data stored in memory or in registers. The addressing modes are briefly explained in the following paragraphs.

Immediate Addressing

In this mode of addressing, the operand field of the instruction contains the value to be used in the operation or computation. The value is an 8-bit quantity that immediately follows the opcode.

Immediate Extended Addressing

This mode of addressing is the same as the immediate addressing mode except that the data is a 16-bit quantity (two bytes) that immediately follows the opcode.

Relative Addressing

Branch instructions are somewhat different from other instructions in that their selected addresses do not indicate the location of data. Instead, the addresses indicate the location of the next instruction. This location will be relative to the current setting of the program counter (plus 2).

For the relative addressing mode to be valid, the branch distance (range) between the instruction and the destination of the branch must fall in the value range of −128 to +127 bytes. This relationship between the relative address and the absolute address of the destination of the branch may be expressed by:

$$DA = (PC+2)+R$$

where:

DA = address of the destination of the branch instruction.
PC = content of the lowest 8 bits of the program counter.
R = the 8-bit, two's complement, binary number stored in the second byte of the instruction.

Extended Addressing

In the extended addressing mode, the instruction uses three bytes of memory with the first byte containing the opcode of the instruction, the second byte containing the low-order 8 bits of the absolute numerical address, and the third byte containing the high-order 8 bits.

The second and third bytes of an extended address instruction may be an address to which a program can jump or it may be an address where an operand is located. To indicate an operand location as opposed to a jump address, enclose the second and third bytes of the instruction in brackets. The use of brackets always means that the enclosed value is to be used as a pointer to a memory location.

NOTE

Do not use parentheses as enclosing symbols. The Model 64000 assembler decodes parentheses as enclosing expressions.

Indexed Addressing

The byte of data following the opcode contains an offset that is added to the designated index register to form a pointer to a memory location. The instruction will specify which index register (IX or IY) that is to be used. The content of the index register will not be altered by this operation.

Register Addressing

The instruction itself specifies the register that contains the data.

Register Indirect Addressing

The instruction specifies a 16-bit register that contains the memory address where the data is located.

Implied Addressing

In a number of instructions, the opcode specifies one or more registers that contain operands or where results are to be stored. For example, there are a number of arithmetic instructions where the accumulator is always implied to be the destination of the results of the operation.

Bit Addressing

The instruction set for the Z80 contains many set, reset, and test instructions. These instructions permit any memory location or register to be specified for a bit operation. Three bits in the opcode designates the specific bit to be manipulated or tested.

Modified Page Zero Addressing

The Z80 microprocessor has a special call instruction that may specify any one of eight locations in page zero of memory. These locations may be used to initialize restart subroutines.

Condition Flags

There are six condition flags associated with each flag register. The state of each flag will be established by the execution of various instructions. Unless the description of an instruction states otherwise (see Chapter 5), the condition flags associated with a specified register are affected as described in the following paragraphs.

Zero (Z) Flag

If the result of the instruction has the value of 'zero', the zero flag will be set to 'one'; otherwise, it will be reset. The zero flag is assigned to bit-position 6 in the flag register.

Sign (S) Flag

After an operation, if the most significant bit in the register has the value of 'one', the sign flag will be set to 'one'; otherwise, it will be reset. The sign flag is assigned to bit-position 7 in the flag register.

Parity/Overflow (P/V) Flag

For logical operations, if the result of the instruction indicates even parity, the parity/overflow flag will be set; otherwise, it will be reset. For signed two's complement arithmetic operation, the parity/overflow flag will be set if the two's complement number in the associated register is in error. The error indicates that the resulting number of the arithmetic operation exceeds the maximum possible number (-128 to +127) that can be represented by the two's complement notation. The parity/overflow flag is assigned to bit-position 2 in the flag register.

Carry (CY) Flag

If the result of the instruction is a carry (from addition) or a borrow (from subtraction) out of the high-order bit, the carry/borrow flag will be set to 'one'; otherwise, it will be reset. The carry flag is assigned to bit-position Ø in the flag register.

Half Carry (HC) Flag

If the instruction causes a carry out of bit 3 into bit 4 of the resulting value, the auxiliary carry flag will be set to 'one'; otherwise, it will be reset. The half carry flag is assigned to bit-position 4 in the flag register.

Add/Subtract (N) Flag

This flag is used to specify the type of instruction that was executed last so that a following decimal adjust operation will be executed correctly by the microprocessor. For all 'add' instructions, the N flag will be reset. For all 'subtract' instructions, the N flag will be set. The add/subtract flag is assigned to bit-position 1 in the flag register.

NOTE

The first four flags (Z, S, P/V, and C) may be checked and used as conditions for jump, call, or return instructions. Bit-positions 3 and 5 in the flag register are not used.

Chapter 2

Operand Rules and Conventions

Types of Information

There are five types of information that may be used as data in the operand field:

a. Register Information - operands may reference directly, data contained in the processor registers such as the stack, register A, or data memory registers B,C,D,E,H, and L.

Example:

```
LD          A,RØ        ;LOAD CONTENTS OF
                        ;REGISTER Ø INTO
                        ;REGISTER A
```

b. Register Pair Information - operands may reference directly data contained in register pairs such as the BC and DE registers.

Examples:

```
LD          SP,HL       ;LOAD REGISTER PAIR
                        ;HL INTO STACK
                        ;POINTER

LD          DE,5FFFH    ;LOAD IMMEDIATE DATA
                        ;INTO REGISTER PAIR
                        ;DE
```

c. Immediate Data - operands may contain immediate data. The required value is inserted directly into the operand field. The value may be in the form of numbers, an expression to be evaluated at assembly time, a symbol, or an ASCII constant enclosed in quotation marks.

Examples:

```
LD        D,ØFFH     ;LOAD "FF" HEX INTO
                     ;REGISTER D

LD        B,"A"      ;LOAD VALUE OF ASCII
                     ;CONSTANT A (Ø1ØØØØØ1)
                     ;INTO REGISTER B
```

d. 16-bit Memory Address - operands may reference a 16-bit absolute memory address within the range of Ø to 65,535 that contains the operand data.

Example:

```
LD     IX,[HL]
```

e. Dual Operands - when the operand field contains two operands, the first operand listed is the destination and the second operand is the source.

Example:

```
LD        A,H        ;LOAD REGISTER H
                     ;INTO REGISTER A
```

Identifying Types of Information

There are nine ways to define the types of information that can be presented in the operand field. These ways are discussed in the following paragraphs.

a. Binary Data - each binary number must be followed by the letter B.

Example:

```
SAM         LD      H,1ØØ1ØØ11B
```

b. Octal Data - each octal number must be followed by either the letter O or the letter Q.

Example:

```
SAM         LD      H,55O

       or
SAM         LD      H,55Q
```

c. Hexadecimal Data - each hexadecimal number must begin with a number and be followed by the letter H.

Example:

```
SAM         LD      H,ØF1H
```

d. Decimal Data - each decimal number may be followed by the letter D or it may stand alone. Any number not specifically identified as binary, octal, or hexadecimal is assumed to be decimal.

Example:

```
SAM         LD      H,55D

       or
SAM         LD      H,55
```

NOTE

Leading zeros are appended or truncated from constants to produce 8- or 16-bit values as required by the particular operand. Spaces are not permitted within a numeric constant.

e. ASCII Constants - one or more ASCII characters enclosed in quotation marks or apostrophes identify an ASCII constant.

Example:

```
            LD      H,'T'           ;LOADS H REG WITH
                                    ;8-BIT ASCII CODE
                                    ;FOR LETTER T

SAM         ASC     "FULTON'S FOLLY"
```

f. Location Counter - The dollar symbol ($) refers to the content of the program counter. The program counter contains the address of the current instruction or data statement being assembled.

Example:

```
JUMP        JP      $+3             ;JUMP TO ADDRESS
                                    ;3 BYTES BEYOND
                                    ;FIRST BYTE OF THIS
                                    ;INSTRUCTION
```

g. Label Assigned Values - The EQU directives can be used to assign values to labels.

Example:

```
SAM         EQU     6AH
```

h. Labels of Instruction or Data. The label assigned to an instruction or a data definition has as its value the address of the first byte of the instruction or data. Instructions elsewhere in the program can refer to this address by its symbolic name.

Example:

```
SAM       JP      FRED        ;JUMP TO INSTRUC-
                              ;TION AT FRED
          .       .
          .       .
          .       .
          .       .

FRED      LD      B,6AH
```

i. Expressions. The operand field may contain an expression consisting of one or more terms acted on by the expression operators. A term may be either a symbol, a string constant, a numeric constant, or an expression. The assembler reduces the entire expression to a single value.

Terms within expressions may be connected by the following arithmetic operators:

a. The plus operator (+) produces the arithmetic sum of its operands.

b. The minus operator (–) produces the arithmetic difference of its operands or the arithmetic negative of its operand when used alone.

c. The asterisk operator (*) produces the arithmetic product of the operands.

d. The slant operator (/) produces the quotient of its operands, discarding any remainder.

e. An instruction enclosed in parentheses is a legal expression in the operand field. For expressions in parentheses, the deepest expression in parentheses is evaluated first.

Care should be taken when using the arithmetic operators since their operational results may affect the condition flags in the condition code register.

Chapter 3

Special Pseudo Instructions

Introduction

This chapter supplements Chapter 3 in the HP Model 64000 Assembler/Linker Reference Manual. It lists and defines in detail those assembler instructions that are applicable to the Z80 microprocessor only.

Define Byte

SYNTAX:

Label	**Operation**	**Operand**
[Name]	DEFB	expression

The DEFB pseudo instruction will store data in consecutive memory locations starting with the current setting of the program counter. The operand field may contain an expression that evaluates to one-byte (8 bits) numbers in the range Ø through 255.

The label name is optional. If the label name is present, it is assigned the starting value of the program counter, and will reference the first byte stored by the DEFB instruction.

Example:

Label	**Operation**	**Operand**
SAM	DEFB	CHARLIE

Define Storage Block

SYNTAX:

Label	Operation	Operand
[Name]	DEFS	expression

The DEFS pseudo instruction may be used to define a block of memory space. The value of the expression in the operand field specifies the number of bytes to be reserved.

Any symbol appearing in the operand field must be predefined. If the value of the operand expression is zero, no memory will be reserved; however, if the optional label name is present, it will be assigned the current value of the program counter.

The DEFS instruction reserves space in memory by incrementing the program counter by the value in the operand expression.

Example:

Label	Operation	Operand	Comment
SAM	DEFS	15	;RESERVE 15 BYTES ;FOR SAM ROUTINE

Define Word

SYNTAX:

Label	Operation	Operand
[Name]	DEFW	expression

The DEFW pseudo instruction will store a 16-bit value from the expression as an address. The value is stored in memory starting at the current setting of the program counter.

Expressions evaluate to one-word (16 bits) numbers, typically addresses. If an expression evaluates to a single byte, it is assumed to be the low-order byte of a l6-bit word where the high-order byte is all zeros.

If the label name is present, it is assigned the starting address of the program counter, and thus will reference the first byte stored by the DEFW instruction.

Example:

Label	Operation	Operand
SAM	DEFW	ØB123H

Chapter **4**

Assembler Output Listing

General

The assembler processes source program modules and produces an output that consists of a source program listing, a relocatable object file, and a symbol cross-reference list. Errors detected by the assembler will be noted in the output listing as error messages. Refer to Appendix D in the Assembler/Linker Manual for a listing of all error codes and their definitions.

Input/Output Files

Source Input File

Input to the assembler is a source file that is created through the editor. It consists of the following:

Example	**Description**
"Z80"	- Assembler directive
Source Code	- Source statements consisting of:
.	
.	Assembler Pseudos - refer to Chapter 3 (Assembler/Linker Reference Manual)
.	
.	Assembler Instructions - refer to Chapter 5, this Supplement
.	
.	

Assembler Output Files

The assembler produces relocatable object modules that are stored under the same name as the source file but in a format that can be processed by the linker. If an object file does not exist at assembly time, the assembler will create one. If an object file does exist, the assembler will replace it.

List File - The list file is a formatted file that is output to a line printer. It can also be stored in another file or applied to the system CRT display. The list may include:

a. Source statements with object code.

b. Error messages.

c. Summary of errors with a description list.

d. Symbol cross-reference list.

Symbol Cross-reference List - All symbols are cross-referenced except local macro labels and parameters. A cross-reference listing contains:

a. Alphabetical list of program symbols.

b. Line numbers where symbols are defined.

c. All references (by line numbers) to symbols in the program.

Output Listing

An example of an assembler output listing is given in table 4-2, using the source program example listed in table 4-1. To illustrate an assembler output listing that contains error messages refer to table 4-3.

NOTE

The source program example was not written as a specific program. It merely lists a group of mnemonics to present a formatted example.

Table 4-1. Source Program Format Example

```
"Z80" LIST XREF
                EXT         DSPL8,KYBD8
                ORG         Ø8ØØH
EXEC8           LD          HL,ØCØØH
                LD          SP,HL
                LD          HL,Ø8Ø5H
                LD          A,Ø3H
LP1             LD          [HL],A
                DEC         HL
                LD          B,Ø6H
LP2             CALL        KYBD8
                JP          C,LIGHT
                PUSH        AF
                DJNZ        LP2
                LD          B,Ø1H
                LD          HL,Ø8Ø5H
                LD          DE,Ø8Ø4H
GO              LD          A,[DE]
                CPDR
                JP          Z,LP2
                DEC         L
                DEC         E
                JP          NZ,GO
                POP         AF
                CPD
                JP          Z,LIGHT
                LD          [HL],A
                DEC         L
LIGHT           CALL        DSPL8
                JP          LP1
                END
```

Table 4-2. Assembler Output Listing

```
FILE: EXEC8:SAVE                    HEWLETT-PACKARD: ZILOG Z80 ASSEMBLER


LINE  LOC    CODE   ADDR                       SOURCE STATEMENT


  1                         "Z80" LIST XREF
  2                                    EXT     DSPL8,KYBD8
  3   Ø8ØØ                             ORG     Ø8ØØH
  4   Ø8ØØ   21     ØCØØ    EXEC8      LD      HL,ØCØØH
  5   Ø8Ø3   F9                        LD      SP,HL
  6   Ø8Ø4   21     Ø8Ø5               LD      HL,Ø8Ø5H
  7   Ø8Ø7   3E     Ø3                 LD      A,Ø3H
  8   Ø8Ø9   77             LP1        LD      [HL],A
  9   Ø8ØA   2B                        DEC     HL
 1Ø   Ø8ØB   Ø6     Ø6                 LD      B,Ø6H
 11   Ø8ØD   CD     ØØØØ    LP2        CALL    KYBD8
 12   Ø81Ø   DA     Ø831               JP      C,LIGHT
 13   Ø813   F5                        PUSH    AF
 14   Ø814   1ØF7                      DJNZ    LP2
 15   Ø816   Ø6     Ø1                 LD      B,Ø1H
 16   Ø818   21     Ø8Ø5               LD      HL,Ø8Ø5H
 17   Ø81B   11     Ø8Ø4               LD      DE,Ø8Ø4H
 18   Ø81E   1A             GO         LD      A,[DE]
 19   Ø81F   EDB9                      CPDR
 2Ø   Ø821   CA     Ø8ØD               JP      Z,LP2
 21   Ø824   2D                        DEC     L
 22   Ø825   1D                        DEC     E
 23   Ø826   C2     Ø81E               JP      NZ,GO
 24   Ø829   F1                        POP     AF
 25   Ø82A   EDA9                      CPD
 26   Ø82C   CA     Ø831               JP      Z,LIGHT
 27   Ø82F   77                        LD      [HL],A
 28   Ø83Ø   2D                        DEC     L
 29   Ø831   CD     ØØØØ    LIGHT      CALL    DSPL8
 3Ø   Ø834   C3     Ø8Ø9               JP      LP1
 31                                    END

  Errors= Ø
```

Table 4-2. Assembler Output Listing (Cont'd)

```
        FILE: EXEC8:SAVE                          CROSS REFERENCE TABLE   PAGE 2


  LINE#      SYMBOL                    TYPE         REFERENCES

             A                         R            7,8,18,27
             B                         R            1Ø,15
             C                         R            12
             DE                        R            17,18
      2      DSPL8                     E            29
             E                         R            22
      4      EXEC8                     A
     18      GO                        A            23
             HL                        R            4,5,6,8,9,16,27
      2      KYBD8                     E            11
             L                         R            21,28
     29      LIGHT                     A            12,26
      8      LP1                       A            3Ø
     11      LP2                       A            14,2Ø
             NZ                        A            23
             SP                        R            5
             Z                         A            2Ø,26
```

NOTE: In the cross-reference table, the letter listed under the TYPE column has the following definition:

A	=	Absolute
C	=	Common (COMN)
D	=	Data (DATA)
E	=	External
M	=	Multiple Defined
P	=	Program (PROG)
R	=	Predefined Register
U	=	Undefined

Table 4-3. Assembler Output with Errors

```
     FILE: EXEC8:SAVE      HEWLETT-PACKARD: ZILOG Z80 ASSEMBLER

   LINE   LOC      CODE     ADDR                     SOURCE STATEMENT


   1                                 "Z80" LIST XREF
   2                                              EXT      DSPL8,KYBD8
   3      Ø8ØØ                                    ORG      Ø8ØØH
   4      Ø8ØØ     21       ØCØØ     EXEC8        LD       HL,ØCØØH
   5                                              LB       SP,HL
     ERROR-UO                                     ^
   6      Ø8Ø3     21       Ø8Ø5                  LD       HL,0805H
   7      Ø8Ø6     3E       Ø3                    LD       A,Ø3H
   8      Ø8Ø8     77                LP1          LD       [HL],A
   9      Ø8Ø9     2B                             DEC      HL
   1Ø     Ø8ØA     Ø6       FF                    LD       B,Ø6FFH
     ERROR-LR,see line       5                               ^
   11     Ø8ØC     CD       ØØØØ     LP2          CALL     KYBD8
   12     Ø8ØF     DA       Ø83Ø                  JP       C,LIGHT
   13     Ø812     F5                             PUSH     AF
   14     Ø813     1ØF7                           DJNZ     LP2
   15     Ø815     Ø6       Ø1                    LD       B,Ø1H
   16     Ø817     21       Ø8Ø5                  LD       HL,Ø8Ø5H
   17     Ø81A     11       Ø8Ø4                  LD       DE,Ø8Ø4H
   18     Ø81D     1A                GO           LD       A,[DE]
   19     Ø81E     EDB9                           CPDR
   2Ø     Ø82Ø     CA       ØØØØ                  JP       Z,LP3
   ERROR-US,see line         1Ø                              ^
   21     Ø823     2D                             DEC      L
   22     Ø824     1D                             DEC      E
   23     Ø825     C2       Ø81D                  JP       NZ,GO
   24     Ø828     F1                             POP      AF
   25     Ø829     EDA9                           CPD
   26     Ø82B     CA       Ø83Ø                  JP       Z,LIGHT
   27     Ø82E     77                             LD       [HL],A
   28     Ø82F     2D                             DEC      L
   29     Ø83Ø     CD       ØØØØ     LIGHT        CALL     DSPL8
   3Ø     Ø833     C3       Ø8Ø8                  JP       LP1
   31                                             END
   Errors = 3, previous error at line 20
```

Table 4-3. Assembler Output with Errors (Cont'd)

```
US - Undefined Symbol, The indicated symbol is not defined as a (wrap around: label or
     declared as an external)
LR - Legal Range, Address or displacement is out of range of the (wrap around:
     instruction's addressing capability)
UO - Unidentified Opcode, Opcode encountered is not defined for (wrap around: this
     micro-processor)
```

```
FILE: EXEC8:SAVE                    CROSS REFERENCE TABLE   PAGE 2

LINE    SYMBOL              TYPE        REFERENCES
        A                   R           7,8,18,27
        B                   R           1Ø,15
        C                   R           12
        DE                  R           17,18
   2    DSPL8               E           29
        E                   R           22
   4    EXEC8               A
  18    GO                  A           23
        HL                  R           4,6,8,9,16,27
   2    KYBD8               E           11
        L                   R           21,28
  29    LIGHT               A           12,26
   8    LP1                 A           30
  11    LP2                 A           14,20
 ***    LP3                 U           2Ø
        NZ                  A           23
        Z                   A           20,26
```

NOTE: Error messages are inserted immediately following the statement where the error occurs. All error messages (after the first error message) will contain a statement which points to the statement where the last error occurred. At the end of the source program listing, an error summary statement will be printed. The summary will contain a statement as to the total number of errors noted, along with a line reference to the previous error. It will also define all error codes listed in the source program listing.

The primary purpose of the error statement that points to the line number where the previous error occurred is to facilitate location of errors. Since some programs may be many pages in length, this feature helps the programmer locate errors quickly (as opposed to thumbing through each page of the program).

Chapter 5

Z80 Instruction Set Summary

General

This chapter describes the instruction set for the Z80 microprocessor. It briefly explains the individual instructions and gives formatted examples and object codes. For a detailed description of the instruction set, refer to the manufacturer's User's Manual.

Each Z80 instruction consists of a mnemonic code and up to two operands. The descriptive symbols used in this chapter to represent items in mnemonic definitions are as follows:

Symbol		**Description**
A	-	Register A (accumulator)
$\overline{A}$	-	Complement of Register A
B	-	Register B
b	-	Specifies one bit in the range Ø-7
[BC]	-	Memory location pointed to by register pair BC
C	-	Register C
cc	-	Specifies the state of the condition flags for certain conditional branch (jump) instructions
CY	-	Carry/borrow flag bit
$\overline{CY}$	-	Complement of carry/borrow flag bit
D	-	Register D
d	-	Specifies a one-byte expression in the range −128 to +127
[DE]	-	Memory location pointed to by register pair DE
E	-	Register E
e	-	Specifies a one-byte expression in the range −126 to +129
H	-	Register H
HC	-	Half carry flag bit
[HL]	-	Memory location pointed to by register pair HL
H (subscript)	-	High-order byte of a 16-bit word
I	-	Interrupt Vector Register
IFF	-	Content of interrupt enable flip-flop
IX	-	X Index Register

[IX+d]	-	Memory location pointed to by the X Index Register plus d displacement
IY	-	Y Index Register
[IY+d]	-	Memory location pointed to by the Y Index Register plus d displacement
L	-	Register L
L (subscript)	-	Low-order byte of a 16-bit word
m	-	Specifies any register r or memory location **[**HL**]**, **[**IX+d**]**, or **[**IY+d**]**
n	-	Specifies a one-byte expression in the range Ø to 255
nn	-	Specifies a two-byte expression in the range Ø to 65,535
PC	-	Program Counter Register
P/V	-	Conditional flag symbol indicating parity/overflow notation
p	-	Page zero memory location
pp	-	Any register pair BC, DE, IX, or SP
qq	-	Any register pair BC, DE, HL, or AF
R	-	Memory Refresh Register
r	-	Any of the following registers: A, B, C, D, E, H, or L
rr	-	Any register pair BC, DE, IY, or SP
s	-	Specifies any register r, value n, or memory location **[**HL**]**, **[**IX+d**]**, or **[**IY+d**]**
SP	-	Stack Pointer Register
ss	-	Any register pair BC, DE, HL, or SP
<---	-	Transfer into
<--->	-	Exchange register content
$\bullet$	-	Boolean AND
$\oplus$	-	Exclusive OR
$\odot$	-	Inclusive OR
$+$	-	Addition
$-$	-	Subtraction
Ø	-	Condition flag is reset
1	-	Condition flag is set

Predefined Symbols

The following symbols are reserved. They have special meaning to the assembler and cannot appear as user-defined symbols.

Symbol	**Definition**
A	Register A
AF	Register pair AF
B	Register B
BC	Register pair BC
C	Register C
D	Register D
DE	Register pair DE
E	Register E
H	Register H
HL	Register pair HL
I	Interrupt Vector Register
IX	X Register Index
IY	Y Register Index
L	Register L
PC	Program Counter Register
R	Memory Refresh Register
SP	Stack Pointer Register

Z80 Instruction Set Summary

ADC A,_ _

Add with carry to Register A

SYNTAX:

Label	Operation	Operand	Object Code(hex)
	ADC	A,A	8F
	ADC	A,B	88
	ADC	A,C	89
	ADC	A,D	8A
	ADC	A,E	8B
	ADC	A,H	8C
	ADC	A,L	8D
	ADC	A,n	CE(n)
	ADC	A,[HL]	8E
	ADC	A,[IX+d]	DD8E(d)
	ADC	A,[IY+d]	FD8E(d)

The ADC A,_ _ instruction will add the content of the designated register or memory location plus the carry flag bit (from register F) to the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: A <--- A+s+CY

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	ADC	A,A	

This instruction will double the content of register A, plus the carry flag bit.

Add (with carry) Register Pair to Register Pair HL

SYNTAX:

Label	Operation	Operand	Object Code(hex)
	ADC	HL,BC	ED4A
	ADC	HL,DE	ED5A
	ADC	HL,HL	ED6A
	ADC	HL,SP	ED7A

The ADC HL,_ _ instruction will add the content of the designated register pair plus the carry flag bit (from register F) to the content of register pair HL. The result of the operation will be stored in register pair HL.

Symbolic Operation: HL <--- HL+ss+CY

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	ADC	HL,DE	

This instruction will add the content of register pair DE to the content of register pair HL, plus the carry flag bit.

ADD A,_ _

Add to Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ADD	A,A	87
	ADD	A,B	8Ø
	ADD	A,C	81
	ADD	A,D	82
	ADD	A,E	83
	ADD	A,H	84
	ADD	A,L	85
	ADD	A,n	C6(n)
	ADD	A,[HL]	86
	ADD	A,[IX+d]	DD86(d)
	ADD	A,[IY+d]	FD86(d)

The ADD A,_ _ instruction will add the content of the designated register or memory location to the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: A <--- A+s

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	ADD	A,[HL]	

This instruction will add the content of memory location addressed by register pair HL to the content of register A.

Add Register Pair to Register Pair HL

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ADD	HL,BC	Ø9
	ADD	HL,DE	19
	ADD	HL,HL	29
	ADD	HL,SP	39

The ADD HL,_ _ instruction will add the content of the designated register pair to the content of register pair HL. The result of the operation will be stored in register pair HL.

Symbolic Operation: HL <--- HL+ss

Condition codes affected: CY, N, and HC.

Example:

Label	Operation	Operand	Comment
	ADD	HL,DE	

This instruction will add the content of register pair DE to the content of register pair HL.

ADD IX,_ _

Add Register Pair to Index Register IX

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ADD	IX,BC	DDØ9
	ADD	IX,DE	DD19
	ADD	IX,IX	DD29
	ADD	IX,SP	DD39

The ADD IX,_ _ instruction will add the content of the designated register pair to the content of index register IX. The result of the operation will be stored in index register IX.

Symbolic Operation: IX <--- IX+pp

Condition codes affected: CY, N, and HC.

Example:

Label	Operation	Operand	Comment
	ADD	IX,IX	

This instruction will double the value of the content of index register IX.

Add Register Pair to Index Register IY

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	ADD	IY,BC	FDØ9
	ADD	IY,DE	FD19
	ADD	IY,IY	FD29
	ADD	IY,SP	FD39

The ADD IY,_ _ instruction will add the content of the designated register pair to the content of index register IY. The result of the operation will be stored in index register IY.

Symbolic Operation: IY <--- IY+rr

Condition codes affected: CY, N, and HC.

Example:

Label	Operation	Operand	Comment
	ADD	IY,BC	

This instruction will add the content of register pair BC to the content of index register IY.

Logical 'AND' with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	AND	A	A7
	AND	B	AØ
	AND	C	A1
	AND	D	A2
	AND	E	A3
	AND	H	A4
	AND	L	A5
	AND	n	E6(n)
	AND	[HL]	A6
	AND	[IX+d]	DDA6(d)
	AND	[IY+d]	FDA6(d)

A logical 'AND' operation will be performed between the byte specified by the operand field and the byte contained in register A. The result of the operation will be stored in register A.

Symbolic Operation: A <--- A • s

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	AND	[HL]	

This instruction performs a logical 'AND' operation using the content of memory location addressed by register pair HL and the content of register A.

BIT b,_ _

Test Bit b

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	BIT	b,reg or **[addr]**	(see below)

bit	Ø,A=CB47	bit	3,A=CB5F	bit	6,A=CB77
	Ø,B=CB4Ø		3,B=CB58		6,B=CB7Ø
	Ø,C=CB41		3,C=CB59		6,C=CB71
	Ø,D=CB42		3,D=CB5A		6,D=CB72
	Ø,E=CB43		3,E=CB5B		6,E=CB73
	Ø,H=CB44		3,H=CB5C		6,H=CB74
	Ø,L=CB45		3,L=CB5D		6,L=CB75
	Ø,**[**HL**]**=CB46		3,**[**HL**]**=CB5E		6,**[**HL**]**=CB76
	Ø,**[**IX+d**]**=DDCB(d)46		3,**[**IY+d**]**=DDCB(d)5E		6,**[**IX+d**]**=DDCB(d)76
	Ø,**[**IY+d**]**=FDCB(d)46		3,**[**IY+d**]**=FDCB(d)5E		6,**[**IY+d**]**=FDCB(d)76
	1,A=CB4F		4,A=CB67		7,A=CB7F
	1,B=CB48		4,B=CB6Ø		7,B=CB78
	1,C=CB49		4,C=CB61		7,C=CB79
	1,D=CB4A		4,D=CB62		7,D=CB7A
	1,E=CB4B		4,E=CB63		7,E=CB7B
	1,E=CB4C		4,H=CB64		7,H=CB7C
	1,L=CB4D		4,L=CB65		7,L=CB7D
	1,**[**HL**]**=CB4E		4,**[**HL**]**=CB66		7,**[**HL**]**=CB7E
	1,**[**IX+d**]**=DDCB(d)4E		4,**[**IX+d**]**=DDCB(d)66		7,**[**IX+d**]**=DDCB(d)7E
	1,**[**IY+d**]**=FDCB(d)4E		4,**[**IY+d**]**=FDCB(d)66		7,**[**IY+d**]**=FDCB(d)7E

2,A=CB57	5,A=CB6F
2,B=CB5Ø	5,B=CB68
2,C=CB51	5,C=CB69
2,D=CB52	5,D=CB6A
2,E=CB53	5,E=CB6B
2,H=CB54	5,H=CB6C
2,L=CB55	5,L=CB6D
2,[HL]=CB56	5,[HL]=CB6E
2,[IX+d]=DDCB(d)56	5,[IX+d]=DDCB(d)6E
2,[IY+d]=FDCB(d)56	5,[IY+d:=FDCB(d)6E

NOTE

The expression (d) in the object codes listed above is the register offset assigned by the user. The hexadecimal value of the offset will be a one-byte expression in the range of −128 to +127.

The BIT b,_ _ instruction will test a specific bit (b) in the designated register or memory location and then adjust the zero (Z) flag accordingly.

Symbolic Operation: Z <--- $\overline{s}_b$

Condition Codes affected: N, HC, and Z.

Example:

Label	Operation	Operand	Comment
	BIT	4,[IX+3H]	

Assume the content of index register IX is Ø5FAH. The instruction will check bit 4 of memory location Ø5FDH and if the bit is set (1), the zero (Z) flag in register F will be reset (Z=Ø).

CALL cc,nn

Conditional Call of Subroutine

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CALL	C,nn	DC(nn)
	CALL	M,nn	FC(nn)
	CALL	NC,nn	D4(nn)
	CALL	NZ,nn	C4(nn)
	CALL	P,nn	F4(nn)
	CALL	PE,nn	EC(nn)
	CALL	PO,nn	E4(nn)
	CALL	Z,nn	CC(nn)

where:

C	=	carry
M	=	sign negative
NC	=	non carry
NZ	=	non zero
P	=	sign positive
PE	=	parity even
PO	=	parity odd
Z	=	zero

The CALL cc,nn instruction will cause the condition flags in register F to be tested for the condition designated by the cc section of the operand. If the condition is true, the microprocessor will push the current content of the program counter (PC) onto the stack. It will then load the nn section of the operand into the program counter to point to the address in memory that contains the first opcode of a subroutine.

If condition cc is false, program execution will continue with the next instruction.

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	CALL	C,Ø5FAH	

If the carry flag in register F is set (1), this instruction will call the subroutine located at memory address Ø5FAH.

CALL nn

Unconditional Call of Subroutine

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CALL	nn	CD(nn)

The CALL nn instruction will push the current content of the program counter (PC) onto the stack. It will then load operand nn into the program counter, pointing to the memory address that contains the first opcode of a subroutine.

Symbolic Operation: $[SP-1] \leftarrow PC_H$
$[SP-2] \leftarrow PC_L$
$PC \leftarrow nn$

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	CALL	Ø5FAH	

This instruction will perform an unconditional branch to memory location Ø5FAH by loading the operand address into the program counter.

Complement Carry Flag

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CCF		3F

The CCF instruction will invert the carry (CY) flag in register F.

Symbolic Operation: CY <--- $\overline{\text{CY}}$

Condition codes affected: CY and N.

Compare with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CP	A	BF
	CP	B	B8
	CP	C	B9
	CP	D	BA
	CP	E	BB
	CP	H	BC
	CP	L	BD
	CP	n	FE(n)
	CP	[HL]	BE
	CP	[IX+d]	DDBE(d)
	CP	[IY+d]	FDBE(d)

The CP _ _ instruction will compare the content of the designated register or memory location with the content of register A. If the comparison is true, the zero (Z) flag will be set. The content of register A will not be affected by this operation.

Symbolic Operation: A–s

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	CP	D	

This instruction will compare the content of register D with the content of register A. The condition flags are set or reset according to the result of the operation.

Compare Memory Location [– HL] with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPD		EDA9

The CPD instruction will compare the content of memory location addressed by the HL register pair with the content of register A. If the comparison is true, the zero (Z) condition flag will be set. The HL and BC register pairs will then be decremented.

The contents of memory location **[HL]** and register A will not be affected by this operation.

Symbolic Operation: A – **[HL]**; HL <---HL–1, BC <---BC–1

Condition codes affected: N, P/V, HC, Z, and S.

CPDR

Compare Memory Block (decremented) with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPDR		EDB9

The CPDR instruction will compare the content of memory location addressed by the HL register pair with the content of register A. If the comparison is true, the zero (Z) condition flag will be set. The HL and BC register pairs will then be decremented. If register A = **[**HL**]**, the instruction will be terminated. If the BC register pair is not zero and if register A ≠ **[**HL**]**, the program counter will be decremented by 2 and the instruction will repeat itself.

NOTE

If the BC register pair is zeroed prior to executing the CPDR instruction, the microprocessor will loop through 64K bytes if no true comparison is found.

Symbolic Operation: A − **[**HL**]**; Hl <--- HL−1, BC <--- BC−1;
Repeat until A=**[**HL**]** or BC=Ø

Condition codes affected: N, P/V, HC, Z, and S.

Compare Memory Location [+HL] with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPI		EDA1

The CPI instruction will compare the content of memory location addressed by the HL register pair with the content of register A. If the comparison is true, the zero (Z) condition flag will be set. The HL register pair is incremented and the BC register pair is decremented.

The contents of memory location **[HL]** and register A are not affected by this operation.

Symbolic Operation: A–[HL]; HL <--- HL+1, BC <--- BC–1

Condition codes affected: N, P/V, HC, Z, and S.

Compare Memory Block (incremented) with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	CPIR		EDB1

The CPIR instruction will compare the content of memory location addressed by register pair HL with the content of register A. If the comparison is true, the zero (Z) condition flag will be set. Register pair HL will be incremented and register pair BC will be decremented. If decrementing causes register pair BC to go to zero or if register A = [HL], the instruction will be terminated. If register pair BC is not zero and if register A ≠ [HL], the program counter will be decremented by 2 and the instruction will repeat itself.

NOTE

If register pair BC is zeroed prior to executing the CPIR instruction and no true comparison is found, the microprocessor will loop through 64K bytes.

Symbolic Operation: A-[HL]; HL <--- HL+1, BC <--- BC-1;
Repeat until A=[HL] or BC=Ø

Condition codes affected: N, P/V, HC, Z, and S.

CPL

Complement Register A

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	CPL		2F

The CPL instruction will invert the content of register A (1's complement).

Symbolic Operation: A <---$\overline{A}$

Condition codes affected: N and HC.

DAA

Decimal Adjust Register A

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	DAA		27

The DAA instruction will conditionally adjust register A for a BCD addition or subtraction operation.

Condition codes affected: CY, P/V, HC, Z, and S.

Decrement

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DEC	A	3D
	DEC	B	Ø5
	DEC	C	ØD
	DEC	D	15
	DEC	E	1D
	DEC	H	25
	DEC	L	2D
	DEC	[HL]	35
	DEC	[IX+d]	DD35(d)
	DEC	[IY+d]	FD35(d)

The DEC _ _ instruction will decrement the content of the register or memory location designated by the operand field.

Symbolic Operation: s <--- s–1

Condition codes affected: N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	DEC	[HL]	

This instruction will decrement the content of the memory location addressed by register pair HL.

DEC IX

Decrement Index Register IX

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DEC	IX	DD2B

The DEC IX instruction will decrement the content of index register IX.

Symbolic Operation: IX <--- IX−1

Condition codes affected: none.

DEC IY

Decrement Index Register IY

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DEC	IY	FD2B

The DEC IY instruction will decrement the content of index register IY.

Symbolic Operation: IY <--- IY−1

Condition codes affected: none.

DEC ss

Decrement Register Pair

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DEC	BC	ØB
	DEC	DE	1B
	DEC	HL	2B
	DEC	SP	3B

The DEC ss instruction will decrement the content of the designated register pair specified by the operand field.

Symbolic Operation: ss <--- ss−1

Example:

Label	Operation	Operand	Comment
	DEC	SP	

This instruction will decrement the content of the Stack Pointer (SP).

Disable Interrupts

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DI		F3

The DI instruction will disable the maskable interrupt by resetting the interrupt enable flip-flops. The interrupt will remain disabled until reactivated by an EI instruction.

Symbolic Operation: IFF <--- Ø

Condition codes affected: none.

Decrement Register B - Conditionally Jump

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	DJNZ	offset	1Ø(e–2)

The DJNZ _ _ instruction will use the content of register B to determine if a branch must take place. The content of register B will be decremented and if a non zero value remains, the value of the offset indicated in the operand field will be added to the program counter (PC). The next instruction will then be brought from the location designated by the new content of the program counter. The jump will be measured from the address of the DJNZ instruction and has a range of –126 to +129 bytes.

If the result of decrementing register B leaves it with a value of zero, program execution will continue with the next instruction.

Symbolic Operation: B <---B–1;
If B=Ø, continue
If B=1, then PC <--- PC+e

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	DJNZ	4FH	

This instruction will cause the program to jump to a location 4F–2 bytes from the current setting of the program counter if the content of register B is not zero.

EI

Enable Interrupts

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EI		FB

The EI instruction will set the interrupt enable flip-flops allowing recognition of any maskable interrupt.

Symbolic Operation: IFF <--- 1

Condition codes affected: none.

EX [SP],HL

Exchange Memory Location and HL Register Pair

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EX	[SP],HL	E3

The EX [SP],HL instruction will exchange the low-order byte in register pair HL with the content of the memory location specified by the stack pointer (SP). It will then exchange the high-order byte in register pair HL with the next highest memory address [SP+1].

Symbolic Operation: L <---> [SP]
H <---> [SP+1]

Condition codes affected: none.

EX [SP],IX

Exchange Memory Location and IX Index Register

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	EX	[SP],IX	DDE3

The EX [SP],IX instruction will exchange the low-order byte in index register IX with the content of the memory location specified by the stack pointer (SP). It will then exchange the high-order byte in index register IX with the next highest memory address [SP+1].

Symbolic Operation: IX_L <---> [SP]
IX_H <---> [SP+1]

Condition codes affected: none.

EX [SP],IY

Exchange Memory Location and IY Index Register

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	EX	[SP],IY	FDE3

The EX [SP],IY instruction will exchange the low-order byte in index register IY with the content of the memory location specified by the stack pointer (SP). It then exchanges the high-order byte in index register IY with the next highest memory address [SP+1].

Symbolic Operation: IY_L <---> [SP]
IY_H <---> [SP+1]

Condition codes affected: none.

EX AF,AF′

Exchange Register AF and Register AF′

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EX	AF,AF′	Ø8

The EX AF,AF′ instruction will exchange the two-byte content of register pairs AF and AF′.

Symbolic Operation: AF <---> AF′

Condition codes affected: none.

EX DE,HL

Exchange Register Pairs DE and HL

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EX	DE,HL	EB

The EX DE,HL instruction will exchange the two-byte content of register pairs DE and HL.

Symbolic Operation: DE <---> HL

Condition codes affected: none.

EXX

Exchange Register Pairs

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	EXX		D9

The EXX instruction will exchange each two-byte content of register pairs BC, DE, and HL with the two-byte content of register pairs BC′, DE′, and HL′ respectively.

Symbolic Operation: BC <---> BC′
DE <---> DE′
HL <---> HL′

Condition codes affected: none.

HALT

Halt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	HALT		76

The HALT instruction will stop microprocessor operation until a subsequent interrupt or reset is received.

Condition codes affected: none.

Interrupt Modes

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	IM	Ø	ED46
	IM	1	ED56
	IM	2	ED5E

The IM Ø instruction will institute interrupt mode Ø. In this mode, the interrupting device may insert any instruction on the data bus for execution by the microprocessor. The first byte of a multi-byte instruction will be read during the interrupt acknowledgement cycle. The following bytes will be read by a normal read sequence.

The IM 1 instruction will institute interrupt mode 1. In this mode, the microprocessor will respond to an interrupt by executing a restart to location ØØ38H.

The IM 2 instruction will institute the vectored interrupt mode 2. In this mode, an indirect call to any memory location by an 8-bit vector supplied by the peripheral device is allowed.

Condition codes affected: none.

IN A,_ _

Load Register A from I/O Port

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	IN	A,[n]	DB(n)

The IN A,_ _ instruction will load register A with one byte of data from the selected I/O port.

Symbolic Operation: A <--- [n]

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	IN	A,[28H]	

This instruction will load into register A one byte of data from the device connected to the I/O port designated by 28H.

Load Register from Designated Device

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	IN	A,[C]	ED78
	IN	B,[C]	ED4Ø
	IN	C,[C]	ED48
	IN	D,[C]	ED5Ø
	IN	E,[C]	ED58
	IN	H,[C]	ED6Ø
	IN	L,[C]	ED68

The IN r,[C] instruction will place the content of register C onto the bottom half of the address bus to select a specified I/O port. The content of register B will be placed on the top half of the address bus. One byte from the selected port will then be loaded into the designated register.

Symbolic Operation: r <--- [C]

Condition codes affected: N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	IN	A,[C]	

This instruction will load one byte of data into register A from I/O port selected by the content of register C.

INC _ _

Increment Register

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INC	A	3C
	INC	B	Ø4
	INC	C	ØC
	INC	D	14
	INC	E	1C
	INC	H	24
	INC	L	2C

The INC _ _ instruction will increment the content of the register designated by the operand field.

Symbolic Operation: r <--- r+1

Condition codes affected: N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	INC	A	

This instruction will increment the content of register A.

INC [HL]

Increment Memory Location [+HL]

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INC	[HL]	34

The INC [HL] instruction will increment the content of memory location addressed by register pair HL.

Symbolic Operation: [HL] <--- [HL]+1

Condition codes affected: N, P/V, HC, Z, and S.

INC I _ _

Increment Index Register

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INC	IX	DD23
	INC	IY	FD23

The INC IX or INC IY instruction will increment the content of the designated index register.

Symbolic Operation: IX <--- IX+1
or
IY <--- IY+1

Condition codes affected: none.

INC [I _ _+d]

Increment Memory Location Addressed by Index Register

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INC	[IX+d]	DD34(d)
	INC	[IY+d]	FD34(d)

The INC [IX+d] or INC [IY+d] instruction will increment the content of memory location addressed by operand [I _ _+d].

Symbolic Operation: [IX+d] <--- [IX+d]+1
or
[IY+d] <--- [IY+d]+1

Condition codes affected: N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	INC	[IX+Ø4H]	

This instruction will add the content of Index Register IX and the two's complement of integer Ø4H to point to a memory address which will then be incremented.

Increment Register Pair

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INC	BC	Ø3
	INC	DE	13
	INC	HL	23
	INC	SP	33

The INC ss instruction will increment the content of the register pair designated by the operand field.

Symbolic Operation: ss <--- ss+1

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	INC	BC	

This instruction will increment the content of register pair BC.

Load Memory Location from I/O Port

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	IND		EDAA

The IND instruction will place the content of register C onto the bottom half of the address bus to select a specified I/O port. The content of register B will be placed on the top half of the address bus. One byte of data from the selected port will then be loaded into the memory location addressed by register pair HL. After the data transfer, the content of register B and the content of register pair HL will be decremented.

Symbolic Operation: [HL] <--- [C]; B <--- B-1, HL <--- HL-1

Condition codes affected: N and Z.

Load Decreasing Memory Locations from I/O Port

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INDR		EDBA

The INDR instruction will place the content of register C onto the bottom half of the address bus to select a specified I/O port. The content of register B will be placed on the top half of the address bus. One byte of data from the selected port will then be loaded into memory location addressed by register pair HL. After the data transfer, the content of register B and the content of register pair HL will be decremented. If decrementing causes register B to go to zero, the instruction will be terminated. If register B is not zero, the instruction will be repeated.

NOTE

If register B is zeroed prior to execution of the INDR instruction, 256 bytes of data will be read in from the I/O port.

Symbolic Operation: [HL] <--- [C]; B <--- B-1, HL <--- HL-1;
Repeat until B=Ø

Condition codes affected: N and Z.

Load Memory Location [+HL] from I/O Port

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INI		EDA2

The INI instruction will place the content of register C onto the bottom half of the address bus to select a specified I/O port. The content of register B will be placed on the top half of the address bus. One byte of data from the selected port will then be loaded into the memory location addressed by register pair HL. After the data transfer, the content of register B will be decremented and the content of register pair HL will be incremented.

Symbolic Operation: [HL] <--- [C]; B <--- B-1, HL <--- HL+1

Condition codes affected: N and Z.

Load Memory Location [+HL] from I/O Port - Repeat

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	INIR		EDB2

The INIR instruction will place the content of register C onto the bottom half of the address bus to select a specified I/O port. The content of register B will be placed on the top half of the address bus. One byte of data from the selected port will then be loaded into the memory location addressed by register pair HL. After the data transfer, the content of register B will be decremented and the content of register pair HL will be incremented. If decrementing causes register B to go to zero, the instruction will be terminated. If register B is not zero, the instruction will be repeated.

NOTE

If register B is zeroed prior to execution of the INIR instruction, 256 bytes of data will be read in from the I/O port.

Symbolic Operation: [HL] <--- [C]; B <--- B-1, HL <--- HL+1;
Repeat until B=Ø

Condition codes affected: N and Z.

Unconditional Jump to Memory Location

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JP	[HL]	E9
	JP	[IX]	DDE9
	JP	[IY]	FDE9
	JP	nn	C3(nn)

The JP _ _ instruction will load the program counter (PC) with the content of the designated register pair or the value assigned by nn. The next instruction will then be brought from the memory location pointed to by the program counter.

Symbolic Operation: PC <--- [HL], or
PC <--- [IX], or
PC <--- [IY], or
PC <--- nn

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	JP	Ø5FAH	

This instruction will cause the value Ø5FAH to be loaded into the program counter. The next instruction will then be brought from memory location Ø5FAH.

Conditional Jump to Memory Location

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JP	C,nn	DA(nn)
	JP	M,nn	FA(nn)
	JP	NC,nn	D2(nn)
	JP	NZ,nn	C2(nn)
	JP	P,nn	F2(nn)
	JP	PE,nn	EA(nn)
	JP	PO,nn	E2(nn)
	JP	Z,nn	CA(nn)

where:

C	=	carry
M	=	sign negative
NC	=	non carry
NZ	=	non zero
P	=	sign positive
PE	=	parity even
PO	=	parity odd
Z	=	zero

The JP cc,nn instruction will cause the condition flags in register F to be tested for the condition designated by the cc section of the operand. If the condition is true, the instruction will load operand nn into the program counter (PC) and program execution will continue with the instruction located at address nn.

If condition cc is false, program execution will continue with the next instruction.

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	JP	C,Ø5FAH	

If the carry flag in register F is set (1), this instruction will cause the program to branch to memory location Ø5FAH. The branch will be accomplished by loading address Ø5FAH into the program counter.

Unconditional Jump to Location PC+e

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	JR	e	18(e–2)

The JR e instruction will unconditionally branch to other portions of the program. The value of the offset e will be added to the program counter (PC) and the next instruction will be brought from the memory location designated by the new content in the program counter. Displacement e may be either a positive or negative number. The jump will be measured from the current content of the program counter and has a range of −126 to +129 bytes.

Symbolic Operation: PC <--- PC+e

Condition codes affected: none.

Example:

Label	**Operation**	**Operand**	**Comment**
	JR	$+1Ø	

This instruction will unconditionally jump to 1Ø locations from the current value of the program counter. The jump will be in a positive direction.

Conditional Jump to Location PC+e

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	JR	C,e	38(e-2)
	JR	NC,e	3Ø(e-2)
	JR	NZ,e	2Ø(e-2)
	JR	Z,e	28(e-2)

where:	C	-	carry condition flag = 1
	NC	-	carry condition flag = Ø
	NZ	-	zero condition flag = Ø
	Z	-	zero condition flag = 1

The JR _ _,e instructions will provide conditional branching to other parts of the program depending on the results of a test on either the carry or zero condition flags. If the condition flag meets the instruction requirements, offset e will be added to the program counter (PC) causing the next instruction to be brought from the new location. The jump will be measured from the current program counter address and has a range of −126 to +129 bytes.

If the test of the carry or zero flag does not meet the instruction requirements, program execution will continue with the next instruction.

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	JR	C,$+5	

If the carry condition flag is set (1), the instruction will cause the program to branch ahead five bytes.

Load Register or Memory Location

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	LD	reg,reg	(see below)

[BC],A=Ø2	A,H=7C	E,[IY+d]=FD5E(d)
[DE],A=12	A,I=ED57	E,A=5F
[HL],A=77	A,L=7D	E,B=58
[HL],B=7Ø	A,R=ED5F	E,C=59
[HL],C=71	A,n=3E(n)	E,D=5A
[HL],D=72	B,[HL]=46	E,E=5B
[HL],E-73	B,[IX+d]=DD46(d)	E,H=5C
[HL],H=74	B,[IY+d]=FD46(d)	E,L=5D
[HL],L=75	B,A=47	E,n=1E(n)
[HL],n=36(n)	B,B=4Ø	H,[HL]=66
[IX+d],A=DD77(d)	B,C=41	H,[IX+d]=DD66(d)
[IX+d],B=DD7Ø(d)	B,D=42	H,[IY+d]=FD66(d)
[IX+d],C=DD71(d)	B,E=43	H,A=67
[IX+d],D=DD72(d)	B,H=44	H,B=6Ø
[IX+d],E=DD73(d)	B,L=45	H,C=61
[IX+d],H=DD74(d)	B,n=Ø6(n)	H,D=62
[IX+d],L=DD75(d)	BC,[nn]=ED4B(nn)	H,E=63
[IX+d],n=DD36(d)(n)	BC,nn=Ø1(nn)	H,H=64
[IY+d],A=FD77(d)	C,[HL]=4E	H,L=65
[IY+d],B=FD7Ø(d)	C,[IX+d]=DD4E(d)	H,n=26(n)
[IY+d],C=FD71(d)	C,[IY+d]=FD4E(d)	HL,[nn]=2A(nn)
[IY+d],D=FD72(d)	C,A=4F	HL,nn=21(nn)
[IY+d],E=FD73(d)	C,B=48	I,A=ED47
[IY+d],H=FD74(d)	C,C=49	IX,[nn]=DD2A(nn)
[IY+d],L=FD75(d)	C,D=4A	IX,nn=DD21(nn)
[IY+d],n=FD36(d)(n)	C,E=4B	IY,[nn]=FD2A(nn)
[nn],A=32(nn)	C,H=4C	IY,nn=FD21(nn)
[nn],BC=ED43(nn)	C,L=4D	L,[HL]=6E
[nn],DE=ED53(nn)	C,n=ØE(n)	L,[IX+d]=DD6E(d)

[nn],HL=22(nn)	D,[HL]=56	L,[IY+d]=FD6E(d)
[nn],IX=DD22(nn)	D,[IX+d]=DD56(d)	L,A=6F
[nn],IY=FD22(nn)	D,[IY+d]=FD56(d)	L,B=68
[nn],SP=ED73(nn)	D,A=57	L,C=69
A,[BC]=ØA	D,B=5Ø	L,D=6A
A,[DE]=1A	D,C=51	L,E=6B
A,[HL]=7E	D,D=52	L,H=6C
A,[IX+d]=DD7E(d)	D,E=53	L,L=6D
A,[IY+d]=FD7E(d)	D,H=54	L,n=2E(n)
A,[nn]=3A(nn)	D,L=55	R,A=ED4F
A,A=7F	D,n=16(n)	SP,HL=F9
A,B=78	DE,[nn]=ED5B(nn)	SP,IX=DDF9
A,C=79	DE,nn=11(nn)	SP,IY=FDF9
A,D=7A	E,[HL]=5E	SP,[nn]=ED7B(nn)
A,E=7B	E,[IX+d]=DD5E(d)	SP,nn=31(nn)

NOTE

Portions of the object codes enclosed in parentheses are user assigned addresses or data. They will take the form of a hexadecimal number and will become part of the object code.

The LD __, __ instruction will load any register or memory location with any other register, memory location, or value.

Condition codes affected: the following condition codes are affected by the LD A,I and LD A,R instructions: N, P/V, HC, Z, and S. All other instructions have no effect on the condition flags.

Examples:

a. **Label**	**Operation**	**Operand**	**Comment**
	LD	A,I	

The content of interrupt vector register I will be loaded into register A.

b. **Label**	**Operation**	**Operand**	**Comment**
	LD	H,[HL]	

The content of memory location addressed by register pair HL will be loaded into register H.

c. **Label**	**Operation**	**Operand**	**Comment**
	LD	A,[IX+5H]	

The content of memory location addressed by the content of the index register (IX)+5H will be loaded into register A.

Load Memory Location - Decrement Register Pairs

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	LDD		EDA8

The LDD instruction will transfer one byte of data from memory location addressed by register pair HL to a memory location addressed by register pair DE. The three register pairs, BC, DE, and HL, will then be decremented.

Symbolic Operation: [DE] <--- [HL];
DE <--- DE–1, HL <--- HL–1, BC <--- BC–1

Condition codes affected: N, P/V, and HC.

LDDR

Load Memory Location - Decrement Register Pairs - Repeat

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	LDDR		EDB8

The LDDR instruction will transfer one byte of data from memory location addressed by register pair HL to a memory location addressed by register pair DE. The three register pairs, BC, DE, and HL, will then be decremented. If decrementing causes register pair BC to go to zero, the instruction will be terminated. If register pair BC does not go to zero, the instruction will be repeated.

Symbolic Operation: [DE] <--- [HL];
DE <--- DE–1, HL <--- HL–1, BC <--- BC–1;
Repeat until BC=Ø

NOTE

If register pair BC is zeroed prior to the execution of the LDDR instruction, the microprocessor will loop through 64K bytes.

Condition codes affected: N, P/V, and HC.

Load Memory Location - Increment Register Pairs

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	LDI		EDAØ

The LDI instruction will transfer one byte of data from memory location addressed by register pair HL to a memory location addressed by register pair DE. Both register pairs, DE and HL, will then be incremented. Register pair BC will be decremented.

Symbolic Operation: [DE] <--- [HL];
DE <--- DE+1, HL <--- HL+1, BC <--- BC–1;

Condition codes affected: N, P/V, and HC.

Load Memory Location - Increment Register Pairs - Repeat

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	LDIR		EDBØ

The LDIR instruction will transfer one byte of data from memory location addressed by register pair HL to a memory location addressed by register pair DE. Both register pairs, DE and HL, will then be incremented. Register pair BC will be decremented. If decrementing causes register pair BC to go to zero, the instruction will be terminated. If register BC does not go to zero, the instruction will be repeated.

NOTE

If register pair BC is zeroed prior to the execution of the LDIR instruction, the microprocessor will loop through 64K bytes.

Symbolic Operation: [DE] <--- [HL];
DE <--- DE+1, HL <--- HL+1, BC <--- BC-1;
Repeat until BC=Ø

Condition codes affected: N, P/V, and HC.

NEG

Negate Contents of Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	NEG		ED44

The NEG instruction will perform the two's complement on the content in register A.

Symbolic Operation: A <--- $\overline{A}$+1

Condition codes affected: CY, N, P/V, HC, Z, and S.

NOP

No Operation

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	NOP		ØØ

The NOP instruction performs no operation. The label field is optional; however, labels can be assigned to NOP statements for program branching. Operands are not permitted with the NOP instruction.

Condition codes affected: none.

OR s

Logical OR with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OR	[HL]	B6
	OR	[IX+d]	DDB6(d)
	OR	[IY+d]	FDB6(d)
	OR	A	B7
	OR	B	BØ
	OR	C	B1
	OR	D	B2
	OR	E	B3
	OR	H	B4
	OR	L	B5
	OR	n	F6(n)

The OR s instruction will perform a logical OR operation between the byte of data specified by the operand and the content of register A. The result will be stored in register A.

Symbolic Operation: A <--- A$\odot$s

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	OR	[HL]	

The content of memory location addressed by register pair HL will be logically ORed with the content of register A. The result will be stored in register A.

Load Output Port - Decrement Register - Repeat

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OTDR		EDBB

The OTDR instruction will cause the data byte at memory location addressed by register pair HL to be temporarily stored in the microprocessor. Register B will be decremented and the content of register C will be placed on the bottom half of the address bus to select the appropriate I/O port. The content of register B will be placed on the top half of the address bus. Next, the data byte stored temporarily will be placed on the data bus and written into the selected peripheral device. Register pair HL will then be decremented. If the decremented register B is not zero, the instruction will be repeated. If register B has gone to zero, the instruction will be terminated.

NOTE

If register B is zeroed prior to the execution of the OTDR instruction, the microprocessor will output 256 bytes of data.

Symbolic Operation: [C] <--- [HL]; B <--- B-1, HL <--- HL+1;
Repeat until B=Ø

Condition codes affected: N and Z.

OTIR

Load Output Port - Increment Register - Repeat

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OTIR		EDB3

The OTIR instruction will cause the data byte at memory location addressed by register pair HL to be temporarily stored in the microprocessor. Register B will be decremented and the content of register C will be placed on the bottom half of the address bus to select the appropriate I/O port. The content of register B will be placed on the top half of the address bus. Next, the data byte stored temporarily will be placed on the data bus and written into the selected peripheral device. Register pair HL will then be incremented. If the decremented register B is not zero, the instruction will be repeated. If register B has gone to zero, the instruction will be terminated.

NOTE

If register B is zeroed prior to the execution of the OTIR instruction, the microprocessor will output 256 bytes of data.

Symbolic Operation: [C] <--- [HL]; B <--- B-1, HL <--- HL-1;
Repeat until B=∅

Condition codes affected: N and Z.

Load Output Port with Register r

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OUT	[C],A	ED79
	OUT	[C],B	ED41
	OUT	[C],C	ED49
	OUT	[C],D	ED51
	OUT	[C],E	ED59
	OUT	[C],H	ED61
	OUT	[C],L	ED69

The OUT **[C]**,r instruction will place the content of register C onto the bottom half of the address bus to select the appropriate I/O port. The content of register B will be placed on the top half of the address bus. Then the byte of data contained in the designated register will be placed on the data bus and written into the selected peripheral device.

Symbolic Operation: [C] <--- r

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	OUT	[C],B	

This instruction will place the content of register B onto the output data bus. The output port will be selected by the content of register C.

OUT [n],A

Load Output Port with Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OUT	[n],A	D3(n)

The OUT [n],A instruction will cause the operand [n] to be placed on the bottom half of the address bus to select the appropriate I/O port. The content of register A will appear on the top half of the address bus. Then the byte of data contained in register A will be placed on the data bus and written into the selected peripheral device.

Symbolic Operation: [n] <--- A

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	OUT	[Ø5H],A	

This instruction will cause the content of register A to be written into the peripheral device mapped to I/O port address Ø5H.

Load Output Port - Decrement Register

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	OUTD		EDAB

The OUTD instruction will cause the data byte at memory location addressed by register pair HL to be temporarily stored in the microprocessor. Register B will be decremented and the content of register C will be placed on the bottom half of the address bus to select the appropriate I/O port. The content of register B will be placed on the top half of the address bus. Next, the data byte stored temporarily will be placed on the data bus and written into the selected peripheral device. Register pair HL will then be decremented.

Symbolic Operation: [C] <--- **[HL]**; B <--- B–1, HL <--- HL–1

Condition codes affected: N and Z.

OUTI

Load Output Port - Increment Register

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	OUTI		EDA3

The OUTI instruction will cause the data byte at memory location addressed by register pair HL to be temporarily stored in the microprocessor. Register B will be decremented and the content of register C will be placed on the bottom half of the address bus to select the appropriate I/O port. The content of register B will be placed on the top half of the address bus. Next, the data byte stored temporarily will be placed on the data bus and written into the selected peripheral device. Register pair HL will then be incremented.

Symbolic Operation: [C] <--- **[HL]**; B <--- B-1, HL <--- HL+1

Condition codes affected: N and Z.

Load Index Register with Top of Stack

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	POP	IX	DDE1
	POP	IY	FDE1

The POP I_ instruction will pull the top two bytes from the external stack into the designated index register. The stack pointer (SP) register holds the 16-bit address of the current 'top' of the stack.

Symbolic Operation: IX_L <--- [SP] or IY_L <--- [SP]

and

IX_H <--- [SP+1] or IY_H <--- [SP+1]

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	POP	IX	

This instruction will load the byte of data at the memory location corresponding to the current address in the stack pointer into the low-order position of index register IX. The stack pointer will then be incremented and the corresponding adjacent memory location will be loaded into the high-order position of IX.

POP qq

Load Register Pair with Top of Stack

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	POP	AF	F1
	POP	BC	C1
	POP	DE	D1
	POP	HL	E1

The POP qq instruction will pull the top two bytes from the external stack into the register pair designated by the operand. The stack pointer (SP) register holds the 16-bit address of the current 'top' of the stack.

Symbolic Operation: qq_L <--- [SP], qq_H <--- [SP+1]

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	POP	BC	

This instruction will load the byte of data at the memory location corresponding to the current address in the stack pointer into the low-order position of register pair BC. The stack pointer will then be incremented and the corresponding adjacent memory location will be loaded into the high-order position of register pair BC.

PUSH I _

Load Index Register onto Stack

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	PUSH	IX	DDE5
	PUSH	IY	FDE5

The PUSH I _ instruction will push the content of the designated index register onto the stack. The instruction will decrement the stack pointer and then load the high-order byte of the designated index register into the memory location addressed by the stack pointer. The stack pointer will be decremented again and then the low-order byte will be loaded into the memory location corresponding to the new address.

Symbolic Operation: [SP–2] <--- IX_L or [SP–2] <--- IY_L

and

[SP–1] <--- IX_H or [SP–1] <--- IY_H

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	PUSH	IX	

This instruction will decrement the stack pointer and load the high-order byte of index register IX into the memory location corresponding to the contents of the stack pointer. The stack pointer will be decremented again and then load the memory location specified by SP-2 with the low-order byte in the index register.

PUSH qq

Load Register Pair onto Stack

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	PUSH	AF	F5
	PUSH	BC	C5
	PUSH	DE	D5
	PUSH	HL	E5

The PUSH qq instruction will push the content of the designated register pair onto the stack. The instruction decrements the stack pointer and then loads the high-order byte of the designated register pair into the memory location addressed by the stack pointer. The stack pointer will be decremented again and then the low-order byte will be loaded into the memory location corresponding to the new address.

Symbolic Operation: [SP–1] <---qq_H , [SP–2] <--- qq_L

Condition codes affected: none.

Example:

Label	**Operation**	**Operand**	**Comment**
	PUSH	BC	

This instruction will decrement the stack pointer and load the high-order byte of register pair BC into the memory location corresponding to the content of the stack pointer. The stack pointer will be decremented again and then will load the memory location specified by SP-2 with the low-order byte in register pair BC.

RES b,m

Reset Bit in Designated Location

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	RES	b,reg or **[**addr**]**	(see below)

bit	Ø,[HL]=CB86	bit	3,[HL]=CB9E	bit	6,[HL]=CBB6
	Ø,[IX+d]=DDCB(d)86		3,[IX+d]=DDCB(d)9E		6,[IX+d]=DDCB(d)B6
	Ø,[IY+d]=FDCB(d)86		3,[IY+d]=FDCB(d)9E		6,[IY+d]=FDCB(d)B6
	Ø,A=CB87		3,A=CB9F		6,A=CBB7
	Ø,B=CB8Ø		3,B=CB98		6,B=CBBØ
	Ø,C=CB81		3,C=CB99		6,C=CBB1
	Ø,D=CB82		3,D=CB9A		6,D=CBB2
	Ø,E=CB83		3,E=CB9B		6,E=CBB3
	Ø,H=CB84		3,H=CB9C		6,H=CBB4
	Ø,L=CB85		3,L=CB9D		6,L=CBB5
	1,[HL]=CB8E		4,[HL]=CBA6		7,[HL]=CBBE
	1,[IX+d]=DDCB(d)8E		4,[IX+d]=DDCB(d)A6		7,[IX+d]=DDCB(d)BE
	1,[IY+d]=FDCB(d)8E		4,[IY+d]=FDCB(d)A6		7,[IY+d]=FDCB(d)BE
	1,A=CB8F		4,A=CBA7		7,A=CBBF
	1,B=CB88		4,B=CBAØ		7,B=CBB8
	1,C=CB89		4,C=CBA1		7,C=CBB9
	1,D=CB8A		4,D=CBA2		7,D=CBBA
	1,E=CB8B		4,E=CBA3		7,E=CBBB
	1,H=CB8C		4,H=CBA4		7,H=CBBC
	1,L=CB8D		4,L=CBA5		7,L=CBBD
	2,[HL]=CB96		5,[HL]=CBAE		
	2,[IX+d]=DDCB(d)96		5,[IX+d]=DDCB(d)AE		
	2,[IY+d]=FDCB(d)96		5,[IY+d]=FDCB(d)AE		
	2,A=CB97		5,A=CBAF		
	2,B=CB9Ø		5,B=CBA8		
	2,C=CB91		5,C=CBA9		
	2,D=CB92		5,D=CBAA		
	2,E=CB93		5,E=CBAB		
	2,H=CB94		5,H=CBAC		
	2,L=CB95		5,L=CBAD		

The RES b,m instruction will reset the designated bit in the specified register or memory location.

Symbolic Operation: s_b <---Ø

Condition codes affected: none.

Examples:

a.

Label	Operation	Operand	Comment
	RES	5,[HL]	

This instruction will reset bit 5 of the data byte in memory location addressed by register pair HL.

b.

Label	Operation	Operand	Comment
	RES	3,E	

This instruction will reset bit 3 of the data byte in register E.

Return from Subroutine

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RET		C9

The RET instruction will cause the byte at the memory location addressed by the stack pointer (SP) register to be moved to the low-order eight bits of the program counter (PC). The stack pointer will then be incremented and the byte at the memory location now pointed to by the stack pointer will be moved into the high-order eight bits of the program counter. Program execution will continue from the point addressed by the program counter.

Symbolic Operation: PC_L <--- [SP], PC_H <--- [SP+1]

Condition codes affected: none.

RET cc

Conditional Return from Subroutine

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RET	C	D8
	RET	M	F8
	RET	NC	DØ
	RET	NZ	CØ
	RET	P	FØ
	RET	PE	E8
	RET	PO	EØ
	RET	Z	C8

where:

C	=	carry
M	=	sign negative
NC	=	non carry
NZ	=	non zero
P	=	sign positive
PE	=	parity even
PO	=	parity odd
Z	=	zero

The RET cc instruction will cause the condition flags in register F to be tested for the condition designated by the operand. If the condition is true, then a RET instruction will be performed. If the condition is false, then program execution will continue with the next step of the program counter.

Condition codes affected: none.

RETI

Return from Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RETI		ED4D

The RETI instruction may be used at the end of a maskable interrupt service routine to:

a. restore the content of the program counter (PC)

and

b. signal an I/O device that the interrupt routine has been completed.

Condition codes affected: none.

RETN

Return from Non-maskable Interrupt

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RETN		ED45

The RETN instruction may be used at the end of a non-maskable interrupt service routine to restore the content of the program counter (PC).

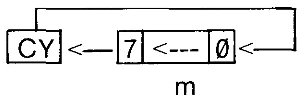
Condition codes affected: none.

Rotate Operand Left Through Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RL	[HL]	CB16
	RL	[IY+d]	DDCB(d)16
	RL	[IY+d]	FDCB(d)16
	RL	A	CB17
	RL	B	CB1Ø
	RL	C	CB11
	RL	D	CB12
	RL	E	CB13
	RL	H	CB14
	RL	L	CB15

The RL m instruction will rotate the content of the designated register or memory location one bit position to the left. The content of bit 7 will be copied into the carry flag bit in register F and the previous carry flag bit will be copied into bit Ø.

Symbolic Operation: 

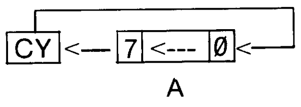
Condition codes affected: CY, N, P/V, HC, Z, and S.

Rotate Register A Left Through Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RLA		17

The RLA instruction will rotate the content of register A one bit position to the left. The content of bit 7 will be copied into the carry flag bit in register F and the previous carry flag bit will be copied into bit Ø.

Symbolic Operation: 

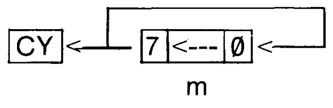
Condition codes affected: CY, N, and HC.

Rotate Operand Left Circular

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RLC	[HL]	CBØ6
	RLC	[IX+d]	DDCB(d)Ø6
	RLC	[IY+d]	FDCB(d)Ø6
	RLC	A	CBØ7
	RLC	B	CBØØ
	RLC	C	CBØ1
	RLC	D	CBØ2
	RLC	E	CBØ3
	RLC	H	CBØ4
	RLC	L	CBØ5

The RLC m instruction will rotate the content of the designated register or memory location one bit position to the left. The content of bit 7 will be copied into bit Ø and also into the carry flag bit position in register F.

Symbolic Operation: 

Condition codes affected: CY, N, P/V, HC, Z, and S.

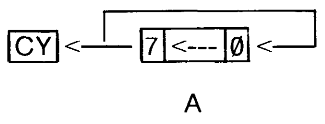
Rotate Register A Left Circular

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RLCA		Ø7

The RLCA instruction will rotate the content of register A one bit position to the left. The content of bit 7 will be copied into bit Ø and also into the carry flag bit position in register F.

Symbolic Operation:



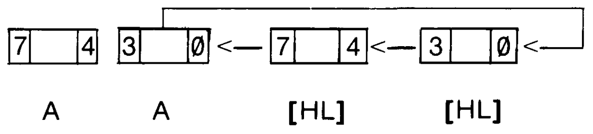
Condition codes affected: CY, N, and HC.

Rotate Register A and Register Pair HL Left

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RLD		ED6F

The RLD instruction will copy the low-order four bits into the high-order four bits of memory location addressed by register pair HL. The previous content of the high-order four bits of the same memory location will be copied into the low-order four bits of register A. The previous content of the low-order four bits of register A will be copied into the low-order four bits of memory location addressed by register pair HL. The high-order four bits of register A will be unaffected.

Symbolic Operation: 

Condition codes affected: N, P/V, HC, Z, and S.

RR m

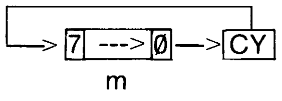
Rotate Operand Right Through Carry

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	RR	[HL]	CB1E
	RR	[IX+d]	DDCB(d)1E
	RR	[IY+d]	FDCB(d)1E
	RR	A	CB1F
	RR	B	CB18
	RR	C	CB19
	RR	D	CB1A
	RR	E	CB1B
	RR	H	CB1C
	RR	L	CB1D

The RR m instruction will rotate the content of the designated register or memory location one bit position to the right. The content of bit Ø will be copied into the carry flag bit in register F and the previous carry flag bit will be copied into bit 7.

Symbolic Operation:



Condition codes affected: CY, N, P/V, HC, Z, and S.

Rotate Register A Right Through Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RRA		1F

The RRA instruction will rotate the content of register A one bit position to the right. The content of bit Ø will be copied into the carry flag bit in register F and the previous carry flag bit will be copied into bit 7.

Symbolic Operation:

$$\begin{tikzcd} \boxed{7} \arrow[r, dashed] & \boxed{\emptyset} \arrow[r] & \boxed{CY} \arrow[ll, bend right] \end{tikzcd}$$

A

Condition codes affected: CY, N, and HC.

RRC m

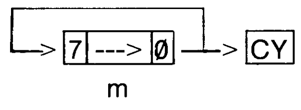
Rotate Operand Right Circular

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RRC	[HL]	CBØE
	RRC	[IX+d]	DDCB(d)ØE
	RRC	[IY+d]	FDCB(d)ØE
	RRC	A	CBØF
	RRC	B	CBØ8
	RRC	C	CBØ9
	RRC	D	CBØA
	RRC	E	CBØB
	RRC	H	CBØC
	RRC	L	CBØD

The RRC m instruction will rotate the content of the designated register or memory location one bit position to the right. The content of bit Ø will be copied into bit 7 and also into the carry flag bit position in register F.

Symbolic Operation:



Condition codes affected: CY, N, P/V, HC, Z, and S.

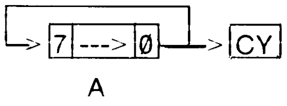
RRCA

Rotate Register A Right Circular

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	RRCA		ØF

The RRCA instruction will rotate the content of register A one bit position to the right. The content of bit Ø will be copied into bit 7 and also into the carry flag bit position in register F.

Symbolic Operation: 

Condition codes affected: CY, N, and HC.

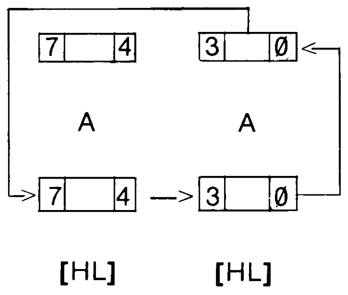
Rotate Register A and Register Pair HL Right

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RRD		ED67

The RRD instruction will copy the low-order four bits of memory location addressed by register pair HL into the low-order four bits of register A. The previous low-order four bits of register A will be copied into the high-order four bits of the memory location addressed by register pair HL. The high-order four bits of this same memory location will be copied into the low-order four bits. The high-order four bits of register A will be unaffected.

Symbolic Operation:



Condition codes affected: N, P/V, HC, Z, and S.

RST p

Restart to Location p

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	RST	ØØH	C7
	RST	Ø8H	CF
	RST	1ØH	D7
	RST	18H	DF
	RST	2ØH	E7
	RST	28H	EF
	RST	3ØH	F7
	RST	38H	FF

The RST p instruction will push the current program counter (PC) content onto the external stack and loads the page zero memory location given in the operand field into the program counter. The restart instruction allows for a jump to one of eight addresses as shown in the syntax block. Since all addresses will be in page zero of memory, the high-order byte of the program counter will be loaded with ØØH. The number selected in the operand column will be loaded into the low-order byte of the program counter.

Symbolic Operation: **[SP–2]** <--- PC_L , **[SP–1]** <--- PC_H ;
PC_H <--- Ø, PC_L <--- p

Condition codes affected: none.

Subtract Operand from Register A with Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SBC	A,[HL]	9E
	SBC	A,[IX+d]	DD9E(d)
	SBC	A,[IY+d]	FD9E(d)
	SBC	A,A	9F
	SBC	A,B	98
	SBC	A,C	99
	SBC	A,D	9A
	SBC	A,E	9B
	SBC	A,H	9C
	SBC	A,L	9D
	SBC	A,n	DE(n)

The SBC A,s instruction will subtract the content of the register or memory location designated in the operand field plus the carry flag bit from the content of register A. The result of the operation will be stored in register A. The instruction may also be used to subtract a numerical value plus the carry flag bit from register A.

Symbolic Operation: A <--- A-s-CY

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	SBC	A,[HL]	

This instruction will subtract the content of memory location addressed by register pair HL plus the carry flag bit from the content of register A. The result of the operation will be stored in register A.

SBC HL,ss

Subtract Operand from Register Pair HL with Carry

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SBC	HL,BC	ED42
	SBC	HL,DE	ED52
	SBC	HL,HL	ED62
	SBC	HL,SP	ED72

The SBC HL,ss instruction will subtract the content of the designated register pair plus the carry flag bit from the content of register pair HL. The result of the operation will be stored in register pair HL.

Symbolic Operation: HL <--- HL-ss-CY

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	SBC	HL,DE	

This instruction will subtract the content of register pair DE plus the carry flag bit from the content of register pair HL. The result of the operation will be stored in register pair HL.

Set Carry Flag

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SCF		37

The SCF instruction will set the carry flag bit in register F.

Symbolic Operation: CY <--- 1

Condition codes affected: CY, N, and HC.

SET b, _ _

Set Bit b

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	SET	b, reg or [addr]	(see below)

bit Ø [HL]=CBC6
Ø,[IX+d]=DDCB(d)C6
Ø,[IY+d]=FDCB(d)C6
Ø,A=CBC7
Ø,B=CBCØ
Ø,C=CBC1
Ø,D=CBC2
Ø,E=CBC3
Ø,H=CBC4
Ø,L=CBC5
1,[HL]=CBCE
1,[IX+d]=DDCB(d)CE
1,[IY+d]=FDCB(d)CE
1,A=CBCF
1,B=CBC8
1,C=CBC9
1,D=CBCA
1,E=CBCB
1,H=CBCC
1,L=CBCD
2,[HL]=CBD6
2,[IX+d]=DDCB(d)D6
2,[IY+d]=FDCB(d)D6
2,A=CBD7
2,B=CBDØ
2,C=CBD1
2,D=CBD2
2,E=CBD3
2,H=CBD4
2,L=CBD5

bit 3,[HL]=CBDE
3,[IX+d]=DDCB(d)DE
3,[IY+d]=FDCB(d)DE
3,A=CBDF
3,B=CBD8
3,C=CBD9
3,D=CBDA
3,E=CBDB
3,H=CBDC
3,L=CBDD
4,[HL]-CBE6
4,[IX+d]=DDCB(d)E6
4,[IY+d]=FDCB(d)E6
4,A=CBE7
4,B=CBEØ
4,C=CBE1
4,D=CBE2
4,E=CBE3
4,H=CBE4
4,H=CBE5
5,[HL]=CBEE
5,[IX+d]=DDCB(d)EE
5,[IY+d]=FDCB(d)EE
5,A=CBEF
5,B=CBE8
5,C=CBE9
5,D=CBEA
5,E=CBEB
5,H=CBEC
5,L=CBED

bit 6,[HL]=CBF6
6,[IX+d]=DDCB(d)F6
6,[IY+d]=FDCB(d)F6
6,A=CBF7
6,B=CBFØ
6,C=CBF1
6,D=CBF2
6,E=CBF3
6,H=CBF4
6,L=CBF5
7,[HL]=CBFE
7,[IX+d]=DDCB(d)FE
7,[IY+d]=FDCB(d)FE
7,A=CBFF
7,B=CBF8
7,C=CBF9
7,D=CBFA
7,E=CBFB
7,H=CBFC
7,L=CBFD

The SET b,_ _ instruction will set bit b in the designated register or memory location.

Symbolic Operation: s_b <--- 1

Condition codes affected: none.

Example:

Label	Operation	Operand	Comment
	SET	4,D	

This instruction will set bit 4 in register D.

Shift Operand Left Arithmetic

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SLA	[HL]	CB26
	SLA	[IX+d]	DDCB(d)26
	SLA	[IY+d]	FDCB(d)26
	SLA	A	CB27
	SLA	B	CB2Ø
	SLA	C	CB21
	SLA	D	CB22
	SLA	E	CB23
	SLA	H	CB24
	SLA	L	CB25

The SLA m instruction will perform an arithmetic shift left one bit position on the content of the designated register or memory location. The content of bit 7 will be copied into the carry flag bit in register F.

Symbolic Operation: $\boxed{\text{CY}} \longleftarrow \underset{m}{\boxed{7}\boxed{\leftarrow\text{---}}\boxed{Ø}} \longleftarrow Ø$

Condition codes affected: CY, N, P/V, HC, Z, and S.

SRA m

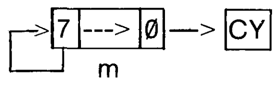
Shift Operand Right Arithmetic

SYNTAX:

Label	**Operation**	**Operand**	**Object Code (hex)**
	SRA	[HL]	CB2E
	SRA	[IX+d]	DDCB(d)2E
	SRA	[IY+d]	FDCB(d)2E
	SRA	A	CB2F
	SRA	B	CB28
	SRA	C	CB29
	SRA	D	CB2A
	SRA	E	CB2B
	SRA	H	CB2C
	SRA	L	CB2D

The SRA m instruction will perform an arithmetic shift right one bit position on the content of the designated register or memory location. The content of bit Ø will be copied into the carry flag bit in register F. The content of bit 7 will remain unchanged.

Symbolic Operation:



Condition codes affected: CY, N, P/V, HC, Z, and S.

SRL m

Shift Operand Right Logical

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SRL	[HL]	CB3E
	SRL	[IX+d]	DDCB(d)3E
	SRL	[IY+d]	FDCB(d)3E
	SRL	A	CB3F
	SRL	B	CB38
	SRL	C	CB39
	SRL	D	CB3A
	SRL	E	CB3B
	SRL	H	CB3C
	SRL	L	CB3D

The SRL m instruction will shift right one bit position the content of the designated register or memory location. The content of bit Ø will be copied into the carry flag bit in register F. Bit 7 will be reset.

Symbolic Operation: $\emptyset \dashrightarrow \underset{m}{\boxed{7 \longrightarrow \emptyset}} \longrightarrow \boxed{CY}$

Condition codes affected: CY, N, P/V, HC, Z, and S.

Subtract Operand from Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	SUB	[HL]	96
	SUB	[IX+d]	DD96(d)
	SUB	[IY+d]	FD96(d)
	SUB	A	97
	SUB	B	90
	SUB	C	91
	SUB	D	92
	SUB	E	93
	SUB	H	94
	SUB	L	95
	SUB	n	D6(n)

The SUB s instruction will subtract the content of the designated register or memory location from the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: A <--- A–s

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	SUB	[HL]	

This instruction will subtract the content of the memory location addressed by register pair HL from the content of register A. The result of the operation will be stored in register A.

XOR s

Exclusive OR Operand and Register A

SYNTAX:

Label	Operation	Operand	Object Code (hex)
	XOR	[HL]	AE
	XOR	[IX+d]	DDAE(d)
	XOR	[IY+d]	FDAE(d)
	XOR	A	AF
	XOR	B	A8
	XOR	C	A9
	XOR	D	AA
	XOR	E	AB
	XOR	H	AC
	XOR	L	AD
	XOR	n	EE(n)

The XOR s instruction will perform an exclusive-OR operation between the content of the designated register or memory location and the content of register A. The result of the operation will be stored in register A.

Symbolic Operation: A <--- A $\oplus$ s

Condition codes affected: CY, N, P/V, HC, Z, and S.

Example:

Label	Operation	Operand	Comment
	XOR	C	

This instruction will perform an exclusive-OR operation between the content of register C and the content of register A. The result of the operation will be stored in register A.