Radio Shack

TRS-80

EDITOR/ ASSEMBLER

Catalog Number 26-2002

User Instruction Manual

CUSTOM MANUFACTURED IN U.S.A. BY RADIO SHACK A DIVISION OF TANDY CORPORATION

TRS-80 EDITOR/ASSEMBLER

OPERATION AND REFERENCE MANUAL

© *Copyright 1978, by Radio Shack, A Division of Tandy Corporation, Ft. Worth, Texas 76102*

TABLE OF CONTENTS

Appendices

Introduction

The TRS-80 Editor/Assembler is a RAM-resident text editor and assembler for a 16K RAM TRS-80 microcomputer system. The Editor/Assembler was designed to provide the ease of use required by the novice, while providing capabilities powerful enough for the expert. LEVEL II BASIC is capable of directly loading the Editor/Assembler cassette tape. LEVEL I BASIC must read-in the Editor/Assembler using the SYSTEM tape (included).

The text editing features of the Editor/Assembler facilitate the manipulation of alphanumeric text files. The most common use of the editing capability is in the creation and maintenance of assembly language source programs.

The assembler portion of the Editor/Assembler facilitates the translation of symbolic language source programs into machine executable code. This object code may then be executed with the SYSTEM tape for LEVEL I BASIC or directly with the SYSTEM command under LEVEL II BASIC. Previous knowledge of machine language and the hexidecimal number system is assumed throughout this manual.

The Assemble command (A) supports the assembler language specifications set forth in the Zilog **Z80-Assembly Language Program Manual**, 3.Ø D.S., REL.2.1, FEB 1977, with the following exceptions.

Macros are not supported.

Operand expressions may only contain the + and –, & (logical AND), and 〈 (shift) operators, and are evaluated on a strictly left to right basis. Parentheses are not allowed!

Conditional assembly commands, where a programmer may control which portions of the source code are assembled, are not supported.

Constants may only be decimal (D), hexidecimal (H), or octal (O). See section under operands.

The only Assembler commands supported are *LIST OFF and *LIST ON.

A label can contain only alphanumeric characters. (Use of the – and ? is not supported.) A label can be up to 6 characters long. The first character must be alphabetic. The other characters must be alphanumeric.

NOTATION CONVENTIONS

[ ] Square brackets enclose optional information:

P[line1[:line2]]

The :line2 is optional, and the P need not be followed by anything at all since all options following P are enclosed in brackets. The brackets are never actually typed.

. . . The ellipses represent repetition of a previous item:

A[[ƀ filename] [/switch[/switch] . . .]]

The /switch may be repeated several times.

CAPITALS Capital letters must be as shown for input, and will be as shown in examples of output.

lowercase The user must substitute in his own values (eg: inc, filename, line)

underscore Underscored information is output printed by the Editor/Assembler unless specified otherwise. This distinguishes user input from computer output but is never actually typed by the user.

ƀ A lowercase B with slash specifies a mandatory blank(space).

line Any decimal number from Ø to 65529

line1:line2 Numbers specify two different line numbers (line #1 is usually less than line #2)

● A period may be used in place of any line number. It represents a pointer to the current line of source code being assembled, printed, or edited.

A pound sign may be used in place of any line number. It represents the **first** (lowest line number) source code line in the text buffer.

* An asterisk may be used in place of any line number. It represents the **last** (highest line number) source code line in the text buffer.

inc A number representing an increment between successive line numbers.

filename A character string specifying the name of a cassette file. See section on Cassette Tapes.

Editor/Assembler

In brief the Editor/Assembler is designed for a user to type in source assembler code. This source code is assembled and the resulting object code may be recorded onto tape. The Editor/Assembler may also read-in, record, and edit other source code files stored on tape. Of course, the source files manipulated by the Editor/Assembler need not be assembly programs only. The files may be any text information created by the Editor/Assembler. BASIC program tapes may NOT be edited by the Editor/Assembler.

The limit to the size of an assembly language program is the amount of RAM memory in the user's computer system. The Editor/Assembler maintains a "text buffer." This buffer starts at the end of the Editor/Assembler program and continues to the end of memory. This usually leaves around 7K of memory for the text buffer which will contain the source file.

LOADING

LEVEL II BASIC

Since the Editor/Assembler is a machine language program, it may only be loaded using the SYSTEM command. Place the Editor/Assembler tape into the cassette recorder and depress PLAY. The volume should be set to 5 or 6 (this is a 5ØØ baud tape).

Type SYSTEM and then press ENTER. The computer will respond by typing:

```
*?
```

Now type EDTASM, the filename of the Editor/Assembler, and the tape will be read into memory. Once loading is completed, type a / (slash) and press ENTER, the monitor screen is cleared and the message:

```
TRS-80 EDITOR/ASSEMBLER 1.2

*
```

is printed. The asterisk is the Editor/Assembler prompt symbol. This is its way of requesting a command. Depressing the BREAK key will always return you to an asterisk except when reading a tape, writing a tape, or editing a line. The BREAK key may be used to abort an assembly or a printout in progress.

LEVEL I BASIC

Since the Editor/Assembler is recorded on tape at 5ØØ baud, LEVEL I BASIC CAN NOT DIRECTLY read-in the tape. You must **first** load the SYSTEM tape provided. This program can then read-in the 5ØØ baud Editor/Assembler tape.

Load the SYSTEM tape into the cassette recorder. Set volume to 8 or 9 (this is a 25Ø baud tape). Type CLOAD and BASIC I will read-in the SYSTEM tape. The program will start as soon as loading is finished.

The computer will type:

```
*
```

Now load your cassette with the Editor/Assembler tape. Set volume to 5-6 (this is a 5ØØ baud tape). Type EDTASM and press ENTER. The Editor/Assembler will be read-in. When the reading is complete, another * will be typed. Now type a slash (/) and then the number 18Ø58. Press ENTER to execute the Editor/Assembler. The number 18Ø58 is the entry address of the Editor/Assembler.

```
TRS-80 EDITOR/ASSEMBLER 1.2

*
```

You may now use the Editor/Assembler as described under the section on Assembly Language.

The BREAK key works the same way as described in the third paragraph of this section.

COMMANDS

The TRS-80 Editor/Assembler can perform the following commands. These commands may be typed after the prompt symbol * where applicable. The asterisk indicates the "command level" of the Editor/Assembler. The following list contains all command level instructions recognized by the Editor/Assembler with a brief description of each.

A	Assemble source currently in text buffer
B	Return to BASIC in ROM
D	Delete specified line(s)
E	Edit a specified command; almost exactly like LEVEL II BASIC's EDIT command
F	Find a specified string of characters in the text buffer
H	Same as P command except that output goes to lineprinter
I	Insert source line(s) at a specified line with a specified increment
L	Load a source file from cassette tape into text buffer
N	Renumber source lines in the text buffer
P	Print specified range of source code currently in the text buffer
R	Replace lines currently in text buffer. Like the Insert command only lines are overwritten
T	Same as H only no line numbers are printed – text only.
↑ or ↓	Scroll up or down. Will print the next or previous source line
→	Horizontal tab
W	Write current text buffer onto tape

Assemble (A)

form: *A[[ƀ filename] [/switch[/switch] . . .]]

switch may be any of the following four options

NL	No listing written to screen. Errors and bad source lines are still typed.
NO	No object code. Inhibits recording of an object code tape.

NS	No symbol table is to be printed
LP	Send listing, errors, and symbol table to the TRS-80 LINEPRINTER
WE	Cause assembly to wait when an error occurs. Depressing any key will continue assembly until another error is found. Depressing the "C" key will cause assembly to continue without stopping for errors. Pressing BREAK returns to command level at any time.

The contents of the edit buffer are assembled. The object code is written to cassette tape under the specified filename (if no filename is specified the filename is automatically set to NONAME.) An assembly error is usually written to the monitor screen immediately before the line the error occurred on.

After the assembly is completed the total number of errors is printed. Finally, the symbol table is printed. The computer then types:

```
READY CASSETTE
```

Prepare your object tape for recording and press ENTER. If you don't want the object code, simply press BREAK and an asterisk (command level) will be returned to you. This is the default procedure which may be altered with the proper switches.

Examples:

*A	Assemble with filename of NONAME; list on screen
*Ab̸IKKY	Same as above; object file is IKKY
*A/NS	Assemble with filename of NONAME, no symbol table
*A/NS/LP	Same as above yet all output is to line-printer
*Ab̸Q/NL	Assemble with filename Q; no listing b̸ is a mandatory blank

Basic (B)

form: *B

Typing a B and then ENTER will return you to a MEMORY SIZE (power up) condition in LEVEL II BASIC or a READY state in LEVEL I BASIC.

Example:

```
*B

MEMORY SIZE?
```

Delete (D)

form: *D[line1[:line2]]

Deletes the line or lines specified from the text buffer.

Examples:

*D1ØØ:5ØØ	Deletes lines 1ØØ through 5ØØ (inclusive) from the text buffer
D#:	Deletes entire text buffer. Clears text buffer
*D.	Deletes line currently pointed to by period (.).
*D1Ø5	Deletes the single line 1Ø5

Edit (E)

form: *E[line]

Allows user to edit/modify source lines just like the EDIT command in LEVEL II BASIC. The only difference is that the Delete command does not enclose deleted information in exclamation points(!).

Examples:

*E.	Edits current line pointed to by period (.).
*E211	Edit line 211

Sub-commands for Edit are A,C,D,E,H,I,K,L,Q,S,X.

Edit Subcommands

A	Restart edit
nC	Change n characters
nD	Delete n characters
E	End editing and enter changes
H	Delete remainder of line and insert string. The H command should not be used to delete an entire line of text. There must always be at least one character on a line, or future use of that line will cause problems.
I	Insert string
nKx	Kill all characters up to the nth occurrence of X
L	Print the rest of the line and go back to starting position
Q	Quit and ignore all editing
nSx	Search for the nth occurrence of X
X	Move to the end of the line and insert
Backspace	Move edit pointer back one space
(SHIFT) (↑)	Escape from any edit mode subcommand
ENTER	ENTER the line in its present (edited) form

The user should experiment with these or refer to the LEVEL II BASIC Manual.

Find (F)

form: *F[string]

where string is a sequence of 16 characters or less

The edit buffer is searched starting at .+1 for the first occurrence of the specified string. If no string is specified, the search is the same as that of the last F command in which a string was specified. If the search string is found the line containing it is printed and period (.) is updated to the printed line. If the string is not found STRING NOT FOUND is printed and period (.) remains unchanged. P# is often used to move period (.) to the beginning of the buffer prior to a search.

Example:

```
*P#

ØØ1ØØ              ORG              7ØØØH

*F3CØØ

ØØ1ØØ VIDEO        ORG              3CØØH

*F

ØØ211              LD               HL,3CØØH

*
```

Hardcopy (H)

form: *H[line1[:line2]]

Prints a line or group of lines onto the TRS-80 LINEPRINTER. Period (.) is updated to point to the last line printed. This command is exactly like the P command.

Example:

H#:	Sends all lines in the text buffer to printer
*H1ØØ:5ØØ	Sends lines 1ØØ through 5ØØ to printer
*H.	Send current line pointed to by period (.) to the lineprinter.
*H	Prints 15 lines starting with the current line to the printer. Not very useful for lineprinter use.

Insert (I)

form: * I line [,inc]

The I command is used to insert lines of text into the edit buffer. All lines of source are usually entered with the I command. After the I command is issued, line numbers are generated and lines of text are inserted into the edit buffer until one of the following conditions occurs:

a BREAK is typed (usually way to exit)

the edit buffer is full

The line number of the next line to be inserted would be greater than or equal to the next exit line in the buffer. The NO ROOM BETWEEN LINES message is typed.

The line number of the next line to be inserted would be greater than 65529.

If inc is not specified it is assumed to be the last specified value. Period (.) is updated to point to the last line inserted. See section, Sample Use of the I command.

Note: Source lines may be up to 128 characters long. This size line is usually not needed. It is recommended that you use lines of approximately 6Ø characters each (printout and listings will be neater).

Load (L)

form: *L[ƀfilename]

The tape is searched for the file specified by filename. If the specified file is found, its contents are **added** to the current contents of the edit buffer. Note that this may result in improperly sequenced line numbers which must be corrected by use of the N command for proper operation. If the user does not wish to add to the current text buffer, then the buffer must be cleared by the D#:* command.

If filename is not given, the next file on the tape is loaded.

When reading, the familiar asterisks will flash in the upper right corner of the screen. The L command can only read source files created by the Editor/Assembler.

Example:

*L	Loads next source file
*LƀMYPROG	Searches for and loads source file named MYPROG. ƀ is a mandatory blank

Number (N)

form: *N[line[,inc]]

The N command is used to renumber the lines in the edit buffer. The first line in the buffer is assigned the number specified or the default ØØ1ØØ if line is not specified. The remaining lines in the buffer are renumbered according to the increment (inc) or the previous inc in an N,R, or I command if inc was not specified. Period (.) points to the same line it did before the N command was used, but the number of this line may be changed.

Examples:

*N	Renumbers from 1ØØ with previous increment
*N5	Renumbers from 5 with previous increment
*N1Ø,5	Renumber from 1Ø in steps of 5

Print (P)

form: *P[line1[:line2]]

Prints a line or group of lines on the monitor screen. Period (.) is updated to point to the last line printed.

Example:

P#:	Prints all lines in the text buffer
*P1ØØ:5ØØ	Prints lines 1ØØ through 5ØØ inclusive
*P.	Prints current line pointed to by period (.)
*P	Prints 15 lines starting with the current line. Repeated use of P allows printout of source without lines being scrolled off the screen

Replace (R)

form: *R[line[,inc]]

The R command only replaces one line and goes into insert mode. If line exists, it is deleted then inserted. If line doesn't exist it is inserted as with the I command. If inc is not specified, the last inc specified by an I, R or N command is used. Period (.) is always updated to the current line.

Example:

*R.	Replace current line
*R1ØØ,1Ø	Start replacing lines beginning at line 1ØØ and incrementing with 1Ø.
*R1ØØ	Start replacing at line 1ØØ using last specified increments.

Type (T)

form: *T[line1[:line2]]

Prints a line or group of lines onto the TRS-80 LINE PRINTER. Period (.) is updated to point to the last line printed. This command is much like the H command, only no line numbers are printed. Only the source text is printed.

Example:

T#:	Sends all lines in the text buffer to printer
*T1ØØ:5ØØ	Sends text for lines 1ØØ through 5ØØ to printer
*T.	Sends current line pointed to by period (.) to the lineprinter.

Scroll and Tab

The Editor/Assembler recognizes the following special characters:

Scroll up

The ↑ command prints the line preceding the current line and updates period (.) to the printed line. (If the current line is the first line in the edit buffer, it is printed and period (.) remains unchanged.)

Scroll down

The ↓ command prints the line following the current line and updates period (.) to point to the printed line. (If the current line is the last line in the buffer, it is printed and period (.) remains unchanged.)

Note: Both ↑ and ↓ must be the first character of the command line or they will be ignored.

Tab

Typing → tabs right to the next 8 character field. Calling the first position of a source line 1 (line number not counted), the tabs are at positions 9,17,25,33,41,49,51 and continue on in increments of 8 up to 255. Tabs should always be used instead of spaces to conserve text buffer space. A tab (Ø9 hex) only takes up one byte.

Delete character

Backarrow (←) will delete the last character typed. If the last character was a tab, the cursor jumps backwards to the next non-blank character.

(Shift ←) Delete Line

A (Shift ←) will delete all of the line currently being entered. This is true for both source lines and commands.

(Shift @) Pause

At any time during an Assembly or printout a (Shift @) may be typed to halt the computer. Pressing ENTER will continue the process. The (Shift @) will not be accepted while a line is being printed or assembled; only between lines. A pause received while assembling will not be recognized

```
TEXT DEFM 'TRS-80 MICROCOMPUTER'
```

while bytes of the text string are being assembled. Another pause must be typed after this line is finished being assembled.

Write (W)

form: *W[b̸filename]

The contents of the edit buffer are written onto a cassette file under the name filename. If filename is not specified no file name is used. Period (.) is always left unchanged.

Example:

*W	Records text buffer to tape with no filename
*Wb̸DEMO	Records text buffer to tape with a filename of DEMO. b̸ is a mandatory blank.

Cassette Tapes

All cassette tapes created by the Editor/Assembler are written at 5ØØ baud. The cassette tape containing the Editor/Assembler is also at 5ØØ baud. Whenever reading a 5ØØ baud tape the VOLUME LEVEL MUST BE BETWEEN 5 AND 6.

The SYSTEM tape, included with the Editor/Assembler, allows LEVEL I BASIC to read-in the 5ØØ baud Editor/Assembler tape. First read-in the 25Ø baud SYSTEM tape (with volume at 8 to 9), and then load in the Editor/Assembler (at volune 5 to 6) as specified in section on Loading.

LEVEL II BASIC may directly read-in the 5ØØ baud Editor/Assembler tape.

Execution of object code programs stored on tapes is performed with the SYSTEM command in LEVEL II BASIC. LEVEL I BASIC must again use the SYSTEM tape to read-in and then execute object code from a 5ØØ baud tape. Examples of creating object code and then executing it are given in section on Sample Use.

Filenames

Cassette filenames must begin with an alphabetic character. The remaining characters must be alphanumeric. The length may not exceed 6 characters. Filenames need not be specified for the A or W commands and in the event that a name is not specified, the file is assigned the NONAME filename. If a filename is not specified when using the L command, the first file encountered on the tape is loaded into memory.

Sample Use

The following is a sample session using the Editor/Assembler to write a program. Comments to the reader are enclosed in [ ] and are not part of the program.

Note: Labels are not preceded by blanks.

```
TRS-80 EDITOR/ASSEMBLER 1.2

*I1ØØ,1Ø
ØØ1ØØ [→|        ORG      5ØØØH        [→|IS A TAB]
ØØ11Ø VIDEO    EQU      3CØØH
ØØ12Ø          LD       HL,VIDEO      ;SOURCE ADDRESS
ØØ13Ø          LD       DE,VIDEO+1    ;DEST. ADDRESS
ØØ14Ø          LD       BC,3FFH       ;BYTE COUNT
ØØ15Ø          LD       (HL),ØBHF     ;GRAPHICS BYTE
ØØ16Ø          LDIR                   ;WHITE OUT SCREEN
ØØ17Ø ;DELAY LOOP TO KEEP WHITE OUT SCREEN ON
ØØ18Ø          LD       B,5
ØØ19Ø LP1      LD       HL,ØFFFFH     ;VALUE TO DECREMENT
ØØ2ØØ LP2      DEC      HL
ØØ21Ø          LD       A,H
ØØ22Ø          OR       L             ;HL=Ø?
ØØ23Ø          JP       NZ,LP2        ;IF NO DEC AGAIN
ØØ24Ø          DJNZ     LP1           ;DEC. B--B
ØØ25Ø          JP       ØH            ;RETURN TO BASIC
00260          END
ØØ27Ø [BREAK]
```

*A XXX [Assemble] [All the following is computer output]

```
5ØØØ            ØØ1ØØ           ORG     5ØØØH
3CØØ            ØØ11Ø  VIDEO    EQU     3CØØH
5ØØØ  21ØØ3C    ØØ12Ø           LD      HL,VIDEO        ;SOURCE ADDRESS
5ØØ3  11Ø13C    ØØ13Ø           LD      DE,VIDEO+1      ;DEST. ADDRESS
5ØØ6  Ø1FFØ3    ØØ14Ø           LD      BC,3FFH         ;BYTE COUNT
5ØØ9  36BF      ØØ15Ø           LD      (HL),ØBFH       ;GRAPHICS BYTE
5ØØB  EDBØ      ØØ16Ø           LDIR                    ;WHITE OUT SCREEN
                ØØ17Ø  ;DELAY LOOP TO KEEP WHITE OUT SCREEN ON
5ØØD  Ø6Ø5      ØØ18Ø           LD      B,5
5ØØF  21FFFF    ØØ19Ø  LP1      LD      HL,ØFFFFH       ;VALUE TO DECREMENT
5Ø12  2B        ØØ2ØØ  LP2      DEC     HL
5Ø13  7C        ØØ21Ø           LD      A,H
5Ø14  B5        ØØ22Ø           OR      L               ;HL=Ø?
5Ø15  C2125Ø    ØØ23Ø           JP      NZ,LP2          ;IF NO DEC AGAIN
5Ø18  1ØF5      ØØ24Ø           DJNZ    LP1             ;DEC.B--B
5Ø1A  C3ØØØØ    ØØ25Ø           JP      ØH              ;RETURN TO BASIC
ØØØØ            ØØ26Ø           END
ØØØØØ  TOTAL  ERRORS
LP2    5Ø12
LP1    5ØØF
VIDEO  3CØ0

READY CASSETTE       [Load tape; set to RECORD]

[ENTER]              [Press ENTER to record object code]

*
```

Now you can save the information in the text buffer (YOUR SOURCE PROGRAM) onto another tape.

*W MYPROG

The tape file MYPROG may be read in by the Editor/Assembler's L command.

Any assembler errors are printed immediately before the line the error occurred in.

Execution in LEVEL I BASIC

First load the SYSTEM tape (included with your Edi Assembler). Put the SYSTEM tape into your cassette. volume is between 8 and 9. Type CLOAD, to load in t SYSTEM tape. The program will execute as soon as loa is completed and will type:

*

Now enter the filename of your object tape, which was created by the Editor/Assembler. Note that you **must** use the filename NONAME if a filename was not specified. With the example program type XXX, the filename of the object tape.

```
*  XXX
```

At this point put the object tape XXX into the cassette recorder and press PLAY. The volume must be at 5 to 6 (this is a 5ØØ baud tape). Asterisks will flash in the upper right screen corner. Once loading is complete the computer will type * again. Now you must enter the starting address of the machine code program. The starting address (ORG) was 5ØØØH which is a decimal 2Ø48Ø. Now type this decimal number preceded with a slash (/). The command looks like this:

```
*  /2Ø48Ø
```

Press ENTER, of course, and the machine code program will execute. The sample program will white-out the video screen with solid graphics characters. This will stay on the screen for about 5 seconds. The program will then return to a READY condition in BASIC.

Executing in LEVEL II BASIC

Execution is much simpler in LEVEL II BASIC. The object tape is again loaded at 5 to 6 volume (as are all 5ØØ baud tapes). The typing is as follows; comments are in brackets[ ] :

```
READY

> SYSTEM

*? XXX          [read in object tape]

*? /2Ø48Ø
```

The program will now execute and then return to a power up condition (ENTER MEMORY SIZE?).

Multiple Modules

You may load several machine language programs into memory, one after the other. The ORG addresses of these instructions must be such that each object program does not conflict with other modules. If you have the following files:

XXX	7ØØØ to 7ØFF hexidecimal
YYY	71ØØ to 71FF hexidecimal
ZZZ	72ØØ to 72FF hexidecimal

You may then enter the three programs as follows:

```
*? XXX

*? YYY

*? ZZZ

*? /28672       [jump to XXX program]
```

ASSEMBLY LANGUAGE

Syntax

The basic format of an assembly command is:

```
[LABEL]    OPCODE    [OPERAND(S)]    [COMMENT]
```

Examples:

```
             ORG       7ØØØH
VIDEO        EQU       3CØØH
             LD        DE,VIDEO+1 ;DESTINATION
```

LABELS

A label is a symbolic name of a line code. Labels are always optional. A label is a string of characters no greater than 6 characters. The first character must be a letter. A label may not contain the $ character. $ is reserved for the value of the reference counter of the current instruction. All labels must start in column 1 of the source line.
The following labels are reserved for referring to registers only and may not be used for other purposes: A,B,C,D,E,H,L,I,R, IX,IY,SP,PC,AF,BC,DE, and HL.

The following 8 labels are reserved for branching conditions and may not be used for other purposes (these conditions apply to status flags):

FLAG	CONDITION SET	CONDITION NOT SET
Carry	C	NC
Zero	Z	NZ
Sign	M(minus)	P(plus)
Parity	PE(even)	PO(odd)

Example: JP NZ, LOOP

If the zero flag is clear (not set), the above instruction jumps to the instruction specified by LOOP.

OPCODES

The opcodes for the TRS-80 Editor/Assembly exactly correspond to those in the Z-80-Assembly Language Programming Manual, 3.0 D.S., REL. 2.1, FEB 1977. See section Index to Instruction Set for the instruction in question.

OPERANDS

Operands are always one or two values separated by commas. Some instructions require no operands at all.

Examples:

```
LD          HL, 3CØØH
```

```
XOR         A

LD          (HL), 2ØH
```

A value in parentheses ( ) specifies indirect addressing when used with registers, or "contents of" otherwise.

Constants may end in any of the following letters:

H – hexidecimal

D – decimal

O – octal

A constant not followed by one of these letters is assumed to be a decimal. A constant must begin with a digit. Thus FFH is illegal, while ØFFH is legal.

Expressions using the +, –, &, operations are described in section, Expressions.

COMMENTS

All comments must begin with a semicolon (;). If a source line starts with a semicolon in column 1 of the line, the entire line is a comment.

Expressions

A value of an operand may be an expression consisting of +,–,&, or < symbols. These operations are executed in a strictly left to right order. **No parentheses are allowed**. All four operators are binary. Both + and – have unary uses also.

Addition (+)

The plus will add two constants and/or symbolic values. When used as a unary operator, it simply echoes the value.

Example:

```
ØØ1E        CON3Ø     EQU     3Ø

ØØ1Ø        CON16     EQU     1ØH

ØØØ3        CON3      EQU     3

3CØØ        VIDEO     EQU     3CØØH

3CØ3        A1        EQU     VIDEO + CON3

ØØ2E        A2        EQU     CON3Ø + CON16

3CØØ        A3        EQU     + VIDEO
```

Subtraction (–)

The minus operator will subtract two constants and/or symbolic values. Unary minus produces a 2's complement.

Examples:

```
3BFD        A1        EQU     VIDEO–CON3

ØØØE        A2        EQU     CON3Ø–CON16

C4ØØ        A3        EQU     –VIDEO
```

Logical AND (&)

The logical AND operator logically adds two constants and/or symbolic values.

Examples:

```
3CØØ        A1        EQU     3CØØH & FFH

ØØØØ        A2        EQU     Ø & 15

ØØØØ        A3        EQU     ØAAAAH & 5555H
```

Shift (<)

The shift operator can be used to shift a value left or right. The form is:

```
        VALUE     <     AMOUNT
```

If AMOUNT is positive, VALUE is shifted left. If AMOUNT is negative, VALUE is shifted right.

Examples:

```
CØØØ        A1        EQU     3CØØH < 4

Ø3CØ        A2        EQU     3CØØH < –4

BBFF        A3        EQU     3CBBH < 8 + 255

Ø3CØ        A4        EQU     15 + 3CØØH < –4
```

Z80 STATUS INDICATORS (FLAGS)

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

7	6	5	4	3	2	1	Ø
S	Z	X	H	X	P/V	N	C

WHERE:

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

Each of the two Z-80 Flag Registers contains 6 bits of status information which are set or reset by CPU operations. (Bits 3 and 5 are not used.) Four of these bits are testable (C,P/V, Z and S) for use with conditional jump, call or return instructions. Two flags are not testable (H,N) and are used for BCD arithmetic.

CARRY FLAG (C)

The carry bit is set or reset depending on the operation being performed. For 'ADD' instructions that generate a carry and 'SUBTRACT' instructions that generate no borrow, the Carry Flag will be set. The Carry Flag is reset by an ADD that does not generate a carry and a 'SUBTRACT' that generates a borrow. This saved carry facilitates software routines for extended precision arithmetic. Also, the 'DAA' instruction will set the Carry Flag if the conditions for making the decimal adjustment are met.

For instructions RLA, RRA, RLS and RRS, the carry bit is used as a link between the LSB and MSB for any register or memory location. During instructions RLCA, RLC s and SLA s, the carry contains the last value shifted out of bit 7 of any register or memory location. During instructions RRCA, RRC s, SRA s and SRL s the carry contains the last value shifted out of bit Ø of any register or memory location.

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

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

ADD/SUBTRACT FLAG (N)

This flag is used by the decimal adjust accumulator instruction (DAA) to distinguish between 'ADD' and 'SUBTRACT' instructions. For all 'ADD' instructions, N will be set to a 'Ø'. For all 'SUBTRACT' instructions, N will be set to a "1".

PARITY/OVERFLOW FLAG

This flag is set to a particular state depending on the operation being performed.

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

For addition, operands with different signs will never cause overflow. When adding operands with like signs and the result has a different sign, the overflow flag is set. For example:

```
+120 = Ø111 1ØØØ          ADDEND
+1Ø5 = Ø11Ø 1ØØ1          AUGEND
+225   111Ø ØØØ1          (–95) SUM
```

The two numbers added together has resulted in a number that exceeds +127 and the two positive operands has resulted in a negative number (–95) which is incorrect. The overflow flag is therefore set.

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

```
   +127  Ø111 1111      MINUEND
(–)–64   11ØØ ØØØØ      SUBTRAHEND
   +191  1Ø11 1111      DIFFERENCE
```

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

Another method for predicting an overflow is to observe the carry into and out of the sign bit. If there is a carry in and no carry out, or if there is no carry in and a carry out, then overflow has occurred.

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

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

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

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

THE HALF CARRY FLAG (H)

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

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

THE ZERO FLAG (Z)

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

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

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

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

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

THE SIGN FLAG (S)

The Sign Flag (S) stores the state of the most significant bit of the Accumulator (Bit 7). When the Z80 performs arithmetic operations on signed numbers, binary two's complement notation is used to represent and process numeric information. A positive number is identified by a 'Ø' in bit 7. A negative number is identified by a '1'. The binary equivalent of the magnitude of a positive number is stored in bits Ø to 6 for a total range of from Ø to 127. A negative number is represented by the two's complement of the equivalent positive number. The total range for negative numbers is from −1 to −128.

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

PSEUDO-OPS

There are nine pseudo-op (assembler directives) which the assembler will recognize. These assembler directives, although written much like processor instructions, are commands to the assembler instead of the processor. They direct the assembler to perform specific tasks during the assembly process but have no meaning to the Z80 processor. These assembler pseudo-ops are:

ORG nn — Sets address reference counter to the value nn.

EQU nn — Sets value of a label to nn in the program: can occur only once for any label.

DEFL nn — Sets value of a label to nn and can be repeated in the program with different values for the same label.

END — Signifies the end of the source program so that any following statements are ignored. If no END statement is found, a warning is produced. The END statement can specify a start address i.e. END LABEL, END 6ØØØH. This address is used by the system program if no start address is given with the slash (/).

DEFB n — Defines the contents of a byte at the current reference counter to be n.

DEFB 's' — Defines the content of one byte of memory to be the ASCII representation of character s.

DEFW nn — Defines the contents of a two-byte word to be nn. The least significant byte is located at the current reference counter while the most significant byte is located at the reference counter plus one.

DEFS nn — Reserves nn bytes of memory starting at the current value of the reference counter.

DEFM 's' — Defines the content of n bytes of memory to be the ASCII representation of string s, where n is the length of s and must be in the range Ø ⟨=n ⟨= 63.

Assembler Commands

The TRS-80 Editor/Assembler supports only two assembler commands. Each command must start in column one of a source line, and must start with an asterisk (*). The assembler commands are:

*LIST OFF — Causes the assembler listing to be suspended, starting with the next line. Errors and bad source lines will still be printed.

*LIST ON — Causes assembler listing to resume, starting with this line.

Z80 INDEX TO INSTRUCTION SET

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

M CYCLES: 2   T STATES: 7(4,3)   4 MHZ E.T.: 1.75

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

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

INSTRUCTION SET TABLE OF CONTENTS

OPERAND NOTATION

The following notation is used in the assembly language:

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

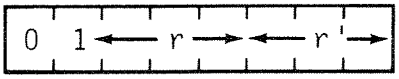
8 BIT LOAD GROUP

LD r, r'

Operation: r ← r'

Format:

Opcode	Operands
LD	r,r'



Description:

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

Register		r,r'
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.Ø

Condition Bits Affected: None

Example:

If the H register contains the number 8AH, and the E register contains 1ØH, the instruction

```
LD H, E
```

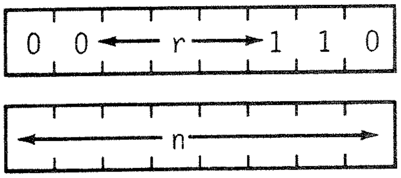
would result in both registers containing 1ØH.

LD r, n

Operation: r ← n

Format:

Opcode	Operands
LD	r, n



Description:

The eight-bit integer n is loaded into any register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

After the execution of

```
LD E, A5H
```

the contents of register E will be A5H.

LD r, (HL)

Operation: r ← (HL)

Format:

Opcode	Operands
LD	r, (HL)

```
| 0 | 1 |←— r —→| 1 | 1 | 0 |
```

Description:

The eight-bit contents of memory location (HL) are loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

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

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

will result in 58H in register C.

LD r, (IX+d)

Operation: r←(IX+d)

Format:

Opcode	Operands
LD	r, (IX+d)

```
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |   DD
| 0 | 1 |←— r —→| 1 | 1 | 0 |
|←———————— d ————————→|
```

Description:

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

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

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

```
LD B, (IX+19H)
```

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

LD r, (IY+d)

Operation: r ← (IY+d)

Format:

Opcode	Operands
LD	r, (IY+d)

1	1	1	1	1	1	0	1	FD

0	1	←	r	→	1	1	0

← d →

Description:

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

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

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

```
LD B, (IY+19H)
```

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

LD (HL), r

Operation: (HL) ← r

Format:

Opcode	Operands
LD	(HL), r

0	1	1	1	0	←	r	→

Description:

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

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

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

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

memory address 2146H will also contain 29H.

LD (IX+d), r

Operation: (IX+d) ←r

Format:

Opcode	Operands
LD	(IX+d), r

1	1	0	1	1	1	0	1	DD
0	1	1	1	0	←	r	→	
←			d				→	

Description:

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

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 1CH, and the Index Register IX contains 31ØØH, then the instruction

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

will perform the sum 31ØØH + 6H and will load 1CH into memory location 31Ø6H.

LD (IY+d), r

Operation: (IY+d) ← r

Format:

Opcode	Operands
LD	(IY+d), r

1	1	1	1	1	1	0	1	FD
0	1	1	1	0	←	r	→	
←			d				→	

Description:

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

Register		r
A	=	111
B	=	ØØØ
C	=	ØØ1
D	=	Ø1Ø
E	=	Ø11
H	=	1ØØ
L	=	1Ø1

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

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

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

will perform the sum 2A11H + 4H, and will load 48H into memory location 2A15.

LD (HL), n

Operation: (HL) ← n

Format:

Opcode	Operand
LD	(HL),n

0	0	1	1	0	1	1	0	36
←			n				→	

Description:

Integer n is loaded into the memory address specified by the contents of the HL register pair.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the HL register pair contains 4444H, the instruction

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

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

LD (IX+d), n

Operation: (IX+d) ← n

Format:

Opcode	Operands
LD	(IX+d), n

1	1	0	1	1	1	0	1	DD
0	0	1	1	0	1	1	0	36
←			d				→	
←			n				→	

Description:

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

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

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

```
LD (IX+5H), 5AH
```

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

LD (IY+d), n

Operation: (IY+d) ← n

Format:

Opcode	Operands
LD	(IY+d),n

1	1	1	1	1	1	0	1	FD
0	0	1	1	0	1	1	0	36
← d →								
← n →								

Description:

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

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IY contains the number A94ØH, the instruction

```
LD (IY+1ØH), 97H
```

would result in byte 97 in memory location A95ØH.

LD A, (BC)

Operation: A ← (BC)

Format:

Opcode	Operands
LD	A, (BC)

0	0	0	0	1	0	1	0	0A

Description:

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

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

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

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

will result in byte 12H in register A.

LD A, (DE)

Operation: A ← (DE)

Format:

Opcode	Operands
LD	A, (DE)

```
0 0 0 1 1 0 1 0    1A
```

Description:

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

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the DE register pair contains the number 3ØA2H and memory address 3ØA2H contains the byte 22H, then the instruction

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

will result in byte 22H in register A.

LD A, (nn)

Operation: A ← (nn)

Format:

Opcode	Operands
LD	A, (nn)

```
0 0 1 1 1 0 1 0    3A
←       n       →
←       n       →
```

Description:

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

M CYCLES: 4 T STATES: 13(4,3,3,3) 4 MHZ E.T.: 3.25

Condition Bits Affected: None

Example:

If the contents of nn is number 8832H, and the content of memory address 8832H is byte Ø4H, after the instruction

```
LD A, (nn)
```

byte Ø4H will be in the Accumulator.

LD (BC), A

Operations: (BC) ← A

Format:

Opcode	Operands
LD	(BC),A

0	0	0	0	0	0	1	0	02

Description:

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

M CYCLES: 2   T STATES: 7(4,3)   4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the Accumulator contains 7AH and the BC register pair contains 1212H the instruction

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

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

LD (DE), A

Operation: (DE) ← A

Format:

Opcode	Operands
LD	(DE),A

0	0	0	1	0	0	1	0	12

Description:

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

M CYCLES: 2   T STATES: 7(4,3)   4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

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

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

will result in AØH being in memory location 1128H.

LD (nn), A

Operation: (nn) ← A

Format:

Opcode	Operands
LD	(nn),A

0	0	1	1	0	0	1	0	32

← n →

← n →

Description:

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

M CYCLES: 4 T STATES: 13(4,3,3,3) 4 MHZ E.T.: 3.25

Condition Bits Affected: None

Example:

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

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

D7H will be in memory location 3141H.

LD A, I

Operation: A ← I

Format:

Opcode	Operands
LD	A, I

1	1	1	0	1	1	0	1	ED
0	1	0	1	0	1	1	1	57

Description:

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

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected:

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

Example:

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

```
LD A, I
```

the accumulator will also contain 4AH.

LD A, R

Operation: A ← R

Format:

Opcode	Operands
LD	A,R

1	1	1	0	1	1	0	1	ED
0	1	0	1	1	1	1	1	5F

Description:

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

M CYCLES: 2   T STATES: 9(4,5)   4 MHZ E.T.: 2.25

Condition Bits Affected:

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

Example:

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

```
LD  A,R
```

the Accumulator will also contain 4AH.

LD I, A

Operation: I ← A

Format:

Opcode	Operands
LD	I,A

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	1	1	47

Description:

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

M CYCLES: 2   T STATES: 9(4,5)   4 MHZ E.T.: 2.25

Condition Bits Affected: None

Example:

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

```
LD  I,A
```

the Interrupt Vector Register will also contain 81H.

LD R, A

Operation: R ← A

Format:

Opcode	**Operands**
LD	R,A,

1	1	1	0	1	1	0	1	ED

0	1	0	0	1	1	1	1	4F

Description:

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

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected: None

Example:

If the Accumulator contains the number B4H, after the instruction

```
LD R,A
```

the Memory Refresh Register will also contain B4H.

16 BIT LOAD GROUP

LD dd, nn

Operation: dd ← nn

Format:

Opcode	Operands
LD	dd, nn

```
0 0 d d 0 0 0 1

←—— n ——→

←—— n ——→
```

Description:

The two-byte integer nn is loaded into the dd register pair, where dd defines the BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

The first n operand in the assembled object code is the low order byte.

M CYCLES: 3 T STATES: 1Ø(4,3 3) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

After the execution of

```
LD HL, 5ØØØH
```

the contents of the HL register pair will be 5ØØØH.

LD IX, nn

Operation: IX ← nn

Format:

Opcode	Operands
LD	IX,nn

```
1 1 0 1 1 1 0 1    DD

0 0 1 0 0 0 0 1    21

←—— n ——→

←—— n ——→
```

Description:

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

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.5Ø

Condition Bits Affected: None

Example:

After the instruction

```
LD IX,45A2H
```

the Index Register will contain integer 45A2H.

LD IY, nn

Operation: IY ← nn

Format:

Opcode	Operands
LD	IY,nn

1	1	1	1	1	1	0	1	FD
0	0	1	0	0	0	0	1	21
← n →								
← n →								

Description:

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

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.5Ø

Condition Bits Affected: None

Example:

After the instruction:

```
LD IY,7733H
```

the Index Register IY will contain the integer 7733H.

LD HL, (nn)

Operation: H ← (nn+1), L ← (nn)

Format:

Opcode	Operands
LD	HL, (nn)

0	0	1	0	1	0	1	0	2A
← n →								
← n →								

Description:

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

M CYCLES: 5 T STATES: 16(4,3,3,3,3) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected: None

Example:

If address 4545H contains 37H and address 4546H contains A1H after the instruction

```
LD HL,(4545H)
```

the HL register pair will contain A137H.

LD dd, (nn)

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

Format:

Opcode	Operands
LD	dd,(nn)

1	1	1	0	1	1	0	1	ED
0	1	d	d	1	0	1	1	
←			n				→	
←			n				→	

Description:

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

Pair	dd
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

The first n operand in the assembled object code above is the low order byte of (nn).

M CYCLES: 6 T STATES: 2Ø(4,4,3,3,3,3) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected: None

Example:

If Address 213ØH contains 65H and address 2131M contains 78H after the instruction

```
LD BC,(213ØH)
```

the BC register pair will contain 7865H.

LD IX, (nn)

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

Format:

Opcode	Operands
LD	IX,(nn)

1	1	0	1	1	1	0	1	DD
0	0	1	0	1	0	1	0	2A
←			n				→	
←			n				→	

Description:

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

M CYCLES: 6 T STATES: 2Ø(4,4,3,3,3,3) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected: None

Example:

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

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

the Index Register IX will contain DA92H.

LD IY, (nn)

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

Format:

Opcode	Operands
LD	IY,(nn)

1	1	1	1	1	1	0	1	FD
0	0	1	0	1	0	1	0	2A
←			n				→	
←			n				→	

Description:

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

M CYCLES: 6 T STATES: 2Ø(4,4,3,3,3,3) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected: None

Example:

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

```
LD  IY,(6666H)
```

the Index Register IY will contain DA92H.

LD (nn), HL

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

Format:

Opcode	Operands
LD	(nn),HL

0	0	1	0	0	0	1	0	22
←			n				→	
←			n				→	

Description:

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

M.CYCLES: 5 T STATES: 16(4,3,3,3,3) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected: None

Example:

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

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

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

LD (nn), dd

Operation: (nn+1) ← dd_H, (nn) ← dd_L

Format:

Opcode	Operands
LD	(nn),dd

1	1	1	0	1	1	0	1	ED
0	1	d	d	0	0	1	1	
←			n				→	
←			n				→	

Description:

The low order byte of register pair dd is loaded into memory address nn ; the upper byte is loaded into memory address nn+1 . Register pair dd defines either BC, DE, HL, or SP, assembled as follows in the object code:

Pair	dd
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

The first n operand in the assembled object code is the low order byte of a two byte memory address.

M CYCLES: 6 T STATES: 2Ø(4,4,3,3,3,3) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected: None

Example:

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

```
LD (1ØØØH),BC
```

will result in 44H in memory location 1ØØØH, and 46H in memory location 1ØØ1H.

LD (nn), IX

Operation: (nn+1) ← IX_H, (nn) ← IX_L

Format:

Opcode	Operands
LD	(nn),IX

1	1	0	1	1	1	0	1	DD
0	0	1	0	0	0	1	0	22
←			n				→	
←			n				→	

Description:

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

M CYCLES: 6 T STATES: 2Ø(4,4,3,3,3,3) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected: None

Example:

If the Index Register IX contains 5A3ØH, after the instruction

```
LD (4392H),IX
```

memory location 4392H will contain number 3ØH and location 4393H will contain 5AH.

LD (nn), IY

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

Format:

Opcode **Operands**

LD (nn),IY

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

| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |   22

| ←———————— n ————————→ |

| ←———————— n ————————→ |
```

Description:

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

M CYCLES: 6 T STATES: 2Ø(4,4,3,3,3,3) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected: None

Example:

If the Index Register IY contains 4174H after the instruction

```
LD 8838H,IY
```

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

LD SP, HL

Operation: SP ← HL

Format:

Opcode **Operands**

LD SP,HL

```
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |   F9
```

Description:

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

M CYCLES: 1 T STATES: 6 4 MHZ E.T.: 1.5Ø

Condition Bits Affected: None

Example:

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

```
LD SP,HL
```

the Stack Pointer will also contain 442EH.

LD SP, IX

Operation: SP ← IX

Format:

Opcode	Operands
LD	SP,IX

1	1	0	1	1	1	0	1	DD
1	1	1	1	1	0	0	1	F9

Description:

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

M CYCLES: 2 T STATES: 1Ø(4,6) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

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

```
LD SP,IX
```

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

LD SP, IY

Operation: SP ← IY

Format:

Opcode	Operands
LD	SP,IY

1	1	1	1	1	1	0	1	FD
1	1	1	1	1	0	0	1	F9

Description:

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

M CYCLES: 2 T STATES: 1Ø(4,6) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

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

```
LD SP,IY
```

the Stack Pointer will also contain A227H.

PUSH qq

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

Format:

Opcode	Operands
PUSH	qq

1	1	q	q	0	1	0	1

Description:

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

Pair	qq
BC	ØØ
DE	Ø1
HL	1Ø
AF	11

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the AF register pair contains 2233H and the Stack Pointer contains 1ØØ7H, after the instruction

```
PUSH AF
```

memory address 1ØØ6H will contain 22H, memory address 1ØØ5H will contain 33H, and the Stack Pointer will contain 1ØØ5H.

PUSH IX

Operation: (SP–2) ← IX_L, (SP–1) ← IX_H

Format:

Opcode	Operands
PUSH	IX

1	1	0	1	1	1	0	1	DD
1	1	1	0	0	1	0	1	E5

Description:

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

M CYCLES: 3 T STATES: 15(4,5,3,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 2233H and the Stack Pointer contains 1ØØ7H, after the instruction

```
PUSH IX
```

memory address 1ØØ6H will contain 22H, memory address 1ØØ5H will contain 33H, and the Stack Pointer will contain 1ØØ5H.

PUSH IY

Operation: $(SP-2) \leftarrow IY_L$, $(SP-1) \leftarrow IY_H$

Format:

Opcode	Operands
PUSH	IY

1	1	1	1	1	1	0	1	FD
1	1	1	0	0	1	0	1	E5

Description:

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

M CYCLES: 4 T STATES: 15(4,5,3,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 2233H and the Stack Pointer contains 1ØØ7H, after the instruction

```
PUSH  IY
```

memory address 1ØØ6H will contain 22H, memory address 1ØØ5H will contain 33H, and the Stack Pointer will contain 1ØØ5H.

POP qq

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

Format:

Opcode	Operands
POP	qq

1	1	q	q	0	0	0	1

Description:

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

Pair	r
BC	ØØ
DE	Ø1
HL	1Ø
AF	11

M CYCLES: 3 T STATES: 1Ø(4,3,3) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1ØØØH, memory location 1ØØØH contains 55H, and location 1ØØ1H contains 33H, the instruction

```
POP  HL
```

will result in register pair HL containing 3355H, and the Stack Pointer containing 1ØØ2H.

POP IX

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

Format:

Opcode	Operands
POP	IX

1	1	0	1	1	1	0	1	DD
1	1	1	0	0	0	0	1	E1

Description:

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

M CYCLES: 4   T STATES: 14(4,4,3,3)   4 MHZ E.T.: 3.5Ø

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1ØØØH, memory location 1ØØØH contains 55H, and location 1ØØ1H contains 33H, the instruction

```
POP  IX
```

will result in the Index Register IX containing 3355H, and the Stack Pointer containing 1ØØ2H.

POP IY

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

Format:

Opcode	Operands
POP	IY

1	1	1	1	1	1	0	1	FD
1	1	1	0	0	0	0	1	E1

Description:

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

M CYCLES: 4   T STATES: 14(4,4,3,3)   4 MHZ E.T.: 3.5Ø

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1ØØØH, memory location 1ØØØH contains 55H, and location 1ØØ1H contains 33H, the instruction

```
POP  IY
```

will result in Index Register IY containing 3355H, and the Stack Pointer containing 1ØØ2H.

EXCHANGE, BLOCK TRANSFER AND SEARCH GROUP

EX DE, HL

Operation: DE ↔ HL

Format:

Opcode	Operands
EX	DE,HL

1	1	1	0	1	0	1	1	EB

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

Example:

If the content of register pair DE is the number 2822H, and the content of the register pair HL is number 499AH, after the instruction

```
EX DE,HL
```

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

EX AF, AF'

Operation: AF ↔ AF'

Format:

Opcode	Operands
EX	AF,AF'

0	0	0	0	1	0	0	0	08

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

Example:

If the content of register pair AF is number 99ØØH, and the content of register pair AF' is number 5944H, after the instruction

```
EX AF,AF'
```

the contents of AF will be 5944H, and the contents of AF will be 99ØØH.

EXX

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

Format:

Opcode **Operands**

EXX

1	1	0	1	1	0	0	1	D9

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

Example:

If the contents of register pairs BC, DE, and HL are the numbers 445AH, 3DA2H, and 8859H, respectively, and the contents of register pairs BC′, DE′, and HL′ are Ø988H, 93ØØH, and ØØE7H, respectively, after the instruction

```
EXX
```

the contents of the register pairs will be as follows: BC: Ø988H; DE: 93ØØH; HL: ØØE7H; BC′: 445AH; DE′: 3DA2H; and HL′: 8859H.

EX (SP), HL

Operation: H ↔ (SP+1), L ↔ (SP)

Format:

Opcode **Operands**

EX (SP),HL

1	1	1	0	0	0	1	1	E3

Description:

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

M CYCLES: 5 T STATES: 19(4,3,4,3,5) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the HL register pair contains 7Ø12H, the SP register pair contains 8856H, the memory location 8856H contains the byte 11H, and the memory location 8857H contains the byte 22H, then the instruction

```
EX (SP),HL
```

will result in the HL register pair containing number 2211H, memory location 8856H containing the byte 12H, the memory location 8857H containing the byte 7ØH and the Stack Pointer containing 8856H.

EX (SP), IX

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

Format:

Opcode	**Operands**
EX	(SP),IX

1	1	0	1	1	1	0	1	DD
1	1	1	0	0	0	1	1	E3

Description:

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

M CYCLES: 6 T STATES: 23(4,4,3,4,3,5) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 3988H, the SP register pair contains Ø1ØØH, the memory location Ø1ØØH contains the byte 9ØH, and memory location Ø1Ø1H contains byte 48H, then the instruction

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

will result in the IX register pair containing number 489ØH, memory location Ø1ØØH containing 88H, memory location Ø1Ø1H containing 39H and the Stack Pointer containing Ø1ØØH.

EX (SP), IY

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

Format:

Opcode	**Operands**
EX	(SP),IY

1	1	1	1	1	1	0	1	FD
1	1	1	0	0	0	1	1	E3

Description:

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

M CYCLES: 6 T STATES: 23(4,4,3,4,3,5) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 3988H, the SP register pair contains Ø1ØØH, the memory location Ø1ØØH contains the byte 9ØH, and memory location Ø1Ø1H contains byte 48H, then the instruction

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

will result in the IY register pair containing number 489ØH, memory location Ø1ØØH containing 88H, memory location Ø1Ø1H containing 39H, and the Stack Pointer containing Ø1ØØH.

LDI

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

Format:

Opcode **Operands**

LDI

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	0	0	A0

Description:

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

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

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

Example:

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

```
LDI
```

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

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

LDIR

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

Format:

Opcode **Operands**

LDIR

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	0	0	BO

Description:

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

For BC≠Ø:

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=Ø:

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

If the HL register pair contains 1111H, the DE register pair contains 2222H, the BC register pair contains ØØØ3H, and memory locations have these contents:

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

then after the execution of

LDIR

the contents of register pairs and memory locations will be:

HL	:	1114H
DE	:	2225H
BC	:	ØØØØH

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

LDD

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

Format:

Opcode   Operands

LDD

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	0	0	A8

Description:

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

M CYCLES: 4   T STATES: 16(4,4,3,5)   4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

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

LDD

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

HL	:	111ØH
(1111H)	:	88H
DE	:	2221H
(2222H)	:	88H
BC	:	6H

LDDR

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

Format:

Opcode	**Operands**
LDDR	

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	0	0	B8

Description:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these registers as well as the BC (Byte Counter) are decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero, the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes. Also, interrupts will be recognized after each data transfer.

For BC≠Ø:

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC≠Ø:

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

If the HL register pair contains 1114H, the DE register pair contains 2225H, the BC register pair contains ØØØ3H, and memory locations have these contents:

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

then after the execution of

LDDR

the contents of register pairs and memory locations will be:

HL	:	1111H
DE	:	2222H
BC	:	ØØØØH

(1114H)	:	A5H	(2225H)	:	A5H
(1113H)	:	36H	(2224H)	:	36H
(1112H)	:	88H	(2223H)	:	88H

CPI

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

Format:

Opcode **Operands**

CPI

1	1	1	0	1	1	0	1	ED
1	0	1	0	0	0	0	1	A1

Description:

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

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if A=(HL); reset otherwise
H: Set if no borrow from Bit 4; reset otherwise
P/V: Set if BC–1≠Ø; reset otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains ØØØ1H, then after the execution of

```
CPI
```

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

CPIR

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

Format:

Opcode **Operands**

CPIR

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	0	1	B1

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL is incremented and the Byte Counter (register pair BC) is decremented. If decrementing causes the BC to go to zero or if A=(HL), the instruction is terminated. If BC is not zero and A≠(HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero before the execution, the instruction will loop through 64K bytes, if no match is found. Also, interrupts will be recognized after each data comparison.

For BC≠Ø and A≠(HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=Ø or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if A=(HL); reset otherwise
H: Set if no borrow from Bit 4; reset otherwise
P/V: Set if BC–1≠Ø; reset otherwise
N: Set
C: Not affected

Example:

If the HL register pair contains 1111H, the Accumulator contains F3H, the Byte Counter contains ØØØ7H, and memory locations have these contents:

(1111H)	:	52H
(1112H)	:	ØØH
(1113H)	:	F3H

then after the execution of

```
CPIR
```

the contents of register pair HL will be 1114H, the contents of the Byte Counter will be ØØØ4H, the P/V flag in the F register will be set and the Z flag in the F register will be set.

CPD

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

Format:

Opcode	Operands
CPD	

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	0	1	A9

Description:

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

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if BC–1≠Ø; reset otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains ØØØ1H, then after the execution of

```
CPD
```

the Byte Counter will contain ØØØØH, the HL register pair will contain 111ØH, the Z flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

CPDR

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

Format:

Opcode	Operands
CPDR	

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	0	1	B9

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and BC (Byte Counter) register pairs are decremented. If decrementing causes the BC to go to zero or if A=(HL), the instruction is terminated. If BC is not zero and A≠(HL), the program counter is decremented by 2 and the instruction is repeated. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes, if no match is found. Also, interrupts will be recognized after each data comparison.

For BC≠Ø and A≠(HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=Ø or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if BC–1≠Ø; reset otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1118H, the Accumulator contains F3H, the Byte Counter contains ØØØ7H, and memory locations have these contents:

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

then after the execution of

```
CPDR
```

the contents of register pair HL will be 1115H, the contents of the Byte Counter will be ØØØ4H, the P/V flag in the F register will be set, and the Z flag in the F register will be set.

8 BIT ARITHMETIC AND LOGICAL GROUP

ADD A, r

Operation: A ← A + r

Format:

Opcode	Operands
ADD	A,r

1	0	0	0	0	← r →

Description:

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

Register	r
A	111
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

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

```
ADD  A,C
```

the contents of the Accumulator will be 55H.

ADD A, n

Operation: A ← A + n

Format:

Opcode	Operands
ADD	A,n

1	1	0	0	0	1	1	0	C6

← n →

Description:

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

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected:

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

Example:

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

```
ADD  A,33H
```

the contents of the Accumulator will be 56H.

ADD A, (HL)

Operation: A ← A + (HL)

Format:

Opcode	**Operands**
ADD	A,(HL)

1	0	0	0	0	1	1	0	86

Description:

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

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are AØH, and the content of the register pair HL is 2323H, and memory location 2323H contains byte Ø8H, after the execution of

```
ADD A,(HL)
```

the Accumulator will contain A8H.

ADD A, (IX+d)

Operation: A ← A + (IX+d)

Format:

Opcode	**Operands**
ADD	A,(IX+d)

1	1	0	1	1	1	0	1	DD
1	0	0	0	0	1	1	0	86
←			d				→	

Description:

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

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected:

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

Example:

If the Accumulator contents are 11H, the Index Register IX contains 1ØØØH, and if the content of memory location 1ØØ5H is 22H, after the execution of

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

the contents of the Accumulator will be 33H.

ADD A, (IY+d)

Operation: A ← A+(IY+d)

Format:

Opcode	Operands
ADD	A,(IY+d)

```
1 1 1 1 1 1 0 1    FD

1 0 0 0 0 1 1 0    86

←――――― d ―――――→
```

Description:

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

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected:

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

Example:

If the Accumulator contents are 11H, the Index Register pair IY contains 1ØØØH, and if the content of memory location 1ØØ5H is 22H, after the execution of

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

the contents of the Accumulator will be 33H.

ADC A, s

Operation: A ← A + s + CY

Format:

Opcode	**Operands**
ADC	A,s

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

ADC A,r	1 0 0 0 1 ←r*→	
ADC A,n	1 1 0 0 1 1 1 0	CE
	← n →	
ADC A,(HL)	1 0 0 0 1 1 1 0	8E
ADC A,(IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 0 0 1 1 1 0	8E
	← d →	
ADC A,(IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 0 0 1 1 1 0	8E
	← d →	

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

Register	r
B	∅∅∅
C	∅∅1
D	∅1∅
E	∅11
H	1∅∅
L	1∅1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
ADC A,r	1	4	1.∅∅
ADC A,n	2	7(4,3)	1.75
ADC A,(HL)	2	7(4,3)	1.75
ADC A,(IX+d)	5	19(4,4,3,5,3)	4.75
ADC A,(IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

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

Example:

If the Accumulator contains 16H, the Carry Flag is set, the HL register pair contains 6666H, and address 6666H contains 1∅H, after the execution of

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

the Accumulator will contain 27H.

SUB s

Operation: A ← A − s

Format:

Opcode	**Operands**
SUB	s

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

SUB r	1	0	0	1	0	← r →			
SUB n	1	1	0	1	0	1	1	0	D6
	← n →								
SUB (HL)	1	0	0	1	0	1	1	0	96
SUB (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	0	1	0	1	1	0	96
	← d →								
SUB (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	0	1	0	1	1	0	96
	← d →								

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

Register	**r**
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	**M CYCLES**	**T STATES**	**4 MHZ E.T.**
SUB r	1	4	1.ØØ
SUB n	2	7(4,3)	1.75
SUB (HL)	2	7(4,3)	1.75
SUB (IX+d)	5	19(4,4,3,5,3)	4.75
SUB (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

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

```
SUB  D
```

the Accumulator will contain 18H.

SBC A, s

Operation: A ← A – s – CY

Format:

Opcode	Operands
SBC	A,s

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

SBC A,r	1	0	0	1	1	← r →			
SBC A,n	1	1	0	1	1	1	1	0	DE
	← n →								
SBC A,(HL)	1	0	0	1	1	1	1	0	9E
SBC A,(IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	0	1	1	1	1	0	9E
	← d →								
SBC A,(IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	0	1	1	1	1	0	9E
	← d →								

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SBC A,r	1	4	1.ØØ
SBC A,n	2	7(4,3)	1.75
SBC A,(HL)	2	7(4,3)	1.75
SBC A,(IX+d)	5	19(4,4,3,5,3)	4.75
SBC A,(IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

If the Accumulator contains 16H, the Carry Flag is set, the HL register pair contains 3433H, and address 3433H contains Ø5H, after the execution of

```
SBC  A,(HL)
```

the Accumulator will contain 1ØH.

AND s

Operation: A ← A ∧ s

Format:

Opcode	**Operands**
AND	s

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

AND r	1	0	1	0	0	←	r	→	
AND n	1	1	1	0	0	1	1	0	E6
	←			n				→	
AND (HL)	1	0	1	0	0	1	1	0	A6
AND (IX+d)	1	1	0	1	1	1	0	1	DD
	1	0	1	0	0	1	1	0	A6
	←			d				→	
AND (IY+d)	1	1	1	1	1	1	0	1	FD
	1	0	1	0	0	1	1	0	A6
	←			d				→	

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
AND r	1	4	1.ØØ
AND n	2	7(4,3)	1.75
AND (HL)	2	7(4,3)	1.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

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

Example:

If the B register contains 7BH (Ø1111Ø11) and the Accumulator contains C3H (11ØØØØ11) after the execution of

```
AND B
```

the Accumulator will contain 43H (Ø1ØØØØ11).

OR s

Operation: A ← A V s

Format:

Opcode	Operands
OR	s

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

OR r	1 0 1 1 0 ←r→	
OR n	1 1 1 1 0 1 1 0	F6
	← n →	
OR (HL)	1 0 1 1 0 1 1 0	B6
OR (IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 1 1 0 1 1 0	B6
	← d →	
OR (IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 1 1 0 1 1 0	B6
	← d →	

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
OR r	1	4	1.ØØ
OR n	2	7(4,3)	1.75
OR (HL)	2	7(4,3)	1.75
OR (IX+d)	5	19(4,4,3,5,3)	4.75
OR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

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

Example:

If the H register contains 48H (Ø1ØØ1ØØØ) and the Accumulator contains 12H (ØØØ1ØØ1Ø) after the execution of

```
OR  H
```

the Accumulator will contain 5AH (Ø1Ø11Ø1Ø).

XOR s

Operation: $A \leftarrow A \oplus s$

Format:

Opcode	Operands
XOR	s

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

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

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
XOR r	1	4	1.ØØ
XOR n	2	7(4,3)	1.75
XOR (HL)	2	7(4,3)	1.75
XOR (IX+d)	5	19(4,4,3,5,3)	4.75
XOR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

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

Example:

If the Accumulator contains 96H (1ØØ1Ø11Ø), after the execution of

XOR 5DH (Note: 5DH = Ø1Ø111Ø1)

the Accumulator will contain CBH (11ØØ1Ø11).

CP s

Operation: A — s

Format:

Opcode	**Operands**
CP	s

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

```
CP r        | 1 | 0 | 1 | 1 | 1 |←—— r ——→|

CP n        | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |    FE
            |←—————————— n ——————————————→|

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

CP (IX+d)   | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |    DD
            | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |    BE
            |←—————————— d ——————————————→|

CP (IY+d)   | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |    FD
            | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |    BE
            |←—————————— d ——————————————→|
```

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

Register	**r**
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	**M CYCLES**	**T STATES**	**4 MHZ E.T.**
CP r	1	4	1.ØØ
CP n	2	7(4,3)	1.75
CP (HL)	2	7(4,3)	1.75
CP (IX+d)	5	19(4,4,3,5,3)	4.75
CP (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

If the Accumulator contains 63H, the HL register pair contains 6ØØØH and memory location 6ØØØH contains 6ØH, the instruction

```
CP  (HL)
```

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

INC r

Operation: r ← r + 1

Format:

Opcode	Operands
INC	r

0	0	← r →	1	0	0

Description:

Register r is incremented. r identifies any of the registers A,B, C,D,E,H or L, assembled as follows in the object code.

Register	r
A	111
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

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

```
INC  D
```

the contents of register D will be 29H.

INC (HL)

Operation: (HL) ← (HL)+1

Format:

Opcode	Operands
INC	(HL)

0	0	1	1	0	1	0	0	34

Description:

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

M CYCLES: 3 T STATES: 11(4,4,3) 4 MHZ E.T.: 2.75

Condition Bits Affected:

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

Example:

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

```
INC  (HL)
```

memory location 3434H will contain 83H.

INC (IX+d)

Operation: (IX+d) ← (IX+d)+1

Format:

Opcode	Operands
INC	(IX+d)

1	1	0	1	1	1	0	1	DD
0	0	1	1	0	1	0	0	34
←			d				→	

Description:

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

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

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

Example:

If the contents of the Index Register pair IX are 2Ø2ØH, and the memory location 2Ø3ØH contains byte 34H, after the execution of

```
INC (IX+1ØH)
```

the contents of memory location 2Ø3ØH will be 35H.

INC (IY+d)

Operation: (IY+d) ← (IY+d)+1

Format:

Opcode	Operands
INC	(IY+d)

1	1	1	1	1	1	0	1	FD
0	0	1	1	0	1	0	0	34
←			d				→	

Description:

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

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

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

Example:

If the contents of the Index Register pair IY are 2Ø2ØH, and the memory location 2Ø3ØH contain byte 34H, after the execution of

```
INC (IY+1ØH)
```

the contents of memory location 2Ø3ØH will be 35H.

DEC m

Operation: m ← m-1

Format:

Opcode	Operands
DEC	m

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

DEC r	0 0 ← r → 1 0 1	
DEC (HL)	0 0 1 1 0 1 0 1	35
DEC (IX+d)	1 1 0 1 1 1 0 1	DD
	0 0 1 1 0 1 0 1	35
	← d →	
DEC (IY+d)	1 1 1 1 1 1 0 1	FD
	0 0 1 1 0 1 0 1	35
	← d →	

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

The byte specified by the m operand is decremented.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
DEC r	1	4	1.ØØ
DEC (HL)	3	11(4,4,3)	2.75
DEC (IX+d)	6	23 (4,4,3,5,4,3)	5.75
DEC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if m was 8ØH before operation; reset otherwise
N:	Set
C:	Not affected

Example:

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

```
DEC D
```

register D will contain 29H.

DAA

Operation: ——

Format:

Opcode

DAA

0	0	1	0	0	1	1	1	27

Description:

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

OPERATION	C BEFORE DAA	HEX VALUE IN UPPER DIGIT (bit 7-4)	H BEFORE DAA	HEX VALUE IN LOWER DIGIT (bit 3-0)	NUMBER ADDED TO BYTE	C AFTER DAA
	Ø	Ø–9	Ø	Ø–9	ØØ	Ø
	Ø	Ø–8	Ø	A–F	Ø6	Ø
	Ø	Ø–9	1	Ø–3	Ø6	Ø
ADD	Ø	A–F	Ø	Ø–9	6Ø	1
ADC	Ø	9–F	Ø	A–F	66	1
INC	Ø	A–F	1	Ø–3	66	1
	1	Ø–2	Ø	Ø–9	6Ø	1
	1	Ø–2	Ø	A–F	66	1
	1	Ø–3	1	Ø–3	66	1
SUB	Ø	Ø–9	Ø	Ø–9	ØØ	Ø
SBC	Ø	Ø–8	1	6–F	FA	Ø
DEC	1	7–F	Ø	Ø–9	AØ	1
NEG	1	6–F	1	6–F	9A	1

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

S: Set if most significant bit of Acc, is 1 after operation; reset otherwise
Z: Set if Acc. is zero after operation; reset otherwise
H: See instruction
P/V: Set if Acc. is even parity after operation; reset otherwise
N: Not affected
C: See instruction

Example:

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

```
 15
+27
 42
```

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

```
 ØØØ1   Ø1Ø1
+ØØ1Ø   Ø111
 ØØ11   11ØØ  =   3C
```

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

```
 ØØ11   11ØØ
+ØØØØ   Ø11Ø
 Ø1ØØ   ØØ1Ø  =   42
```

CPL

Operation: A ← $\overline{A}$

Format:

Opcode

CPL

0	0	1	0	1	1	1	1	2F

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are 1Ø11 Ø1ØØ, after the execution of

```
CPL
```

the Accumulator contents will be Ø1ØØ 1Ø11.

NEG

Operation: A ← o-A

Format:

Opcode

NEG

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	0	0	44

Description:

Contents of the Accumulator are negated (two's complement). This is the same as subtracting the contents of the Accumulator from zero. Note that 8ØH is left unchanged.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if no borrow from Bit 4; reset otherwise
P/V:	Set if Acc. was 8ØH before operation; reset otherwise
N:	Set
C:	Set if Acc. was not ØØH before operation; reset otherwise

Example:

If the contents of the Accumulator are

1	0	0	1	1	0	0	0

after the execution of

```
NEG
```

the Accumulator contents will be

0	1	1	0	1	0	0	0

CCF

Operation: CY ← $\overline{\mathrm{CY}}$

Format:

Opcode

CCF

0	0	1	1	1	1	1	1	3F

Description:

The C flag in the F register is inverted.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

SCF

Operation: CY ← 1

Format:

Opcode

SCF

0	0	1	1	0	1	1	1	37

Description:

The C flag in the F register is set.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

NOP

Operation: ——

Format:

Opcode

NOP

0	0	0	0	0	0	0	0	00

Description:

CPU performs no operation during this machine cycle.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

HALT

Operation: ——

Format:

Opcode

HALT

0	1	1	1	0	1	1	0	76

Description:

The HALT instruction suspends CPU operation until a subsequent interrupt or reset is received. While in the halt state, the processor will execute NOP's to maintain memory refresh logic.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

DI

Operation: IFF ← Ø

Format:

Opcode

DI

1	1	1	1	0	0	1	1	F3

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

Example:

When the CPU executes the instruction

```
DI
```

the maskable interrupt is disabled until it is subsequently re-enabled by an EI instruction. The CPU will not respond to an Interrupt Request (INT) signal.

EI

Operation: IFF ← 1

Format:

Opcode

EI

1	1	1	1	1	0	1	1	FB

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

Example:

When the CPU executes instruction

```
EI
```

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

IM 0

Operation: ——

Format:

Opcode	**Operands**
IM	Ø

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	1	0	46

Description:

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

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected: None

IM 1

Operation: ——

Format:

Opcode	**Operands**
IM	1

1	1	1	0	1	1	0	1	ED
0	1	0	1	0	1	1	0	56

Description:

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

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected: None

IM 2

Operation: ——

Format:

Opcode	Operands
IM	2

1	1	1	0	1	1	0	1	ED
0	1	0	1	1	1	1	0	5E

Description:

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

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

ADD HL, ss

Operation: HL ← HL+ss

Format:

Opcode	**Operands**
ADD	HL,ss

0	0	s	s	1	0	0	1

Description:

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

Register Pair	**ss**
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

M CYCLES: 3 T STATES: 11(4,4,3) 4 MHZ E.T.: 2.75

Condition Bits Affected:

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

Example:

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

```
ADD HL,DE
```

the HL register pair will contain 5353H.

ADC HL, ss

Operation: HL ← HL+ss+CY

Format:

Opcode	**Operands**
ADC	HL,ss

1	1	1	0	1	1	0	1	ED
0	1	s	s	1	0	1	0	

Description:

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

Register Pair	**ss**
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If the register pair BC contains 2222H, register pair HL contains 5437H and the Carry Flag is set, after the execution of

```
ADC HL,BC
```

the contents of HL will be 765AH.

SBC HL, ss

Operation: HL ← HL—ss—CY

Format:

Opcode	Operands
SBC	HL,ss

1	1	1	0	1	1	0	1	ED
0	1	s	s	0	0	1	0	

Description:

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

Register Pair	ss
BC	ØØ
DE	ØØ
HL	1Ø
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if no borrow from Bit 12; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

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

```
SBC  HL,DE
```

the contents of HL will be 8887H.

ADD IX, pp

Operation: IX ← IX + pp

Format:

Opcode	Operands
ADD	IX,pp

1	1	0	1	1	1	0	1	DD
0	0	p	p	1	0	0	1	

Description:

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

Register Pair	pp
BC	ØØ
DE	Ø1
IX	1Ø
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

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

Example:

If the contents of Index Register IX are 333H and the contents of register pair BC are 5555H, after the execution of

```
ADD  IX,BC
```

the contents of IX will be 8888H.

ADD IY, rr

Operation: IY ← IY+rr

Format:

Opcode	Operands
ADD	IY,rr

1	1	1	1	1	1	0	1	FD

0	0	r	r	1	0	0	1

Description:

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

Register Pair	rr
BC	ØØ
DE	Ø1
IY	1Ø
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

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

Example:

If the contents of Index Register IY are 333H and the contents of register pair BC are 555H, after the execution of

```
ADD  IY,BC
```

the contents of IY will be 8888H.

INC ss

Operation: ss ← ss + 1

Format:

Opcodes	Operands
INC	ss

0	0	s	s	0	0	1	1

Description:

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

Register Pair	ss
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

M CYCLES: 1 T STATES: 6 4 MHZ E.T. 1.5Ø

Condition Bits Affected: None

Example:

If the register pair contains 1ØØØH, after the execution of

```
INC  HL
```

HL will contain 1ØØ1H.

INC IX

Operation: IX ← IX + 1

Format:

Opcode	Operands
INC	IX

1	1	0	1	1	1	0	1	DD
0	0	1	0	0	0	1	1	23

Description:

The contents of the Index Register IX are incremented.

M CYCLES: 2 T STATES: 1Ø(4,6) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

If the Index Register IX contains the integer 33ØØH after the execution of

```
INC IX
```

the contents of Index Register IX will be 33Ø1H.

INC IY

Operation: IY ← IY + 1

Format:

Opcode	Operands
INC	IY

1	1	1	1	1	1	0	1	FD
0	0	1	0	0	0	1	1	23

Description:

The contents of the Index Register IY are incremented.

M CYCLES: 2 T STATES: 1Ø(4,6) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

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

```
INC IY
```

the contents of Index Register IY will be 2978H.

DEC ss

Operation: ss ← ss −1

Format:

Opcode	Operands
DEC	ss

0	0	s	s	1	0	1	1

Description:

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

Pair	ss
BC	ØØ
DE	Ø1
HL	1Ø
SP	11

M CYCLES: 1   T STATES: 6   4 MHZ E.T.: 1.5Ø

Condition Bits Affected: None

Example:

If register pair HL contains 1ØØ1H, after the execution of

```
DEC  HL
```

the contents of HL will be 1ØØØH.

DEC IX

Operation: IX ← IX −1

Format:

Opcode	Operands
DEC	IX

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

Description:

The contents of Index Register IX are decremented.

M CYCLES: 2   T STATES: 1Ø(4,6)   4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

If the contents of Index Register IX are 2ØØ6H, after the execution of

```
DEC  IX
```

the contents of Index Register IX will be 2ØØ5H.

DEC IY

Operation: IY ← IY −1

Format:

Opcode **Operands**

DEC IY

1	1	1	1	1	1	0	1	FD
0	0	1	0	1	0	1	1	2B

Description:

The contents of the Index Register IY are decremented.

M CYCLES: 2 T STATES: 1Ø (4,6) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

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

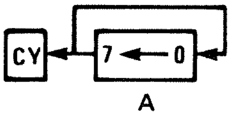
```
DEC  IY
```

the contents of Index Register IY will be 7648H.

ROTATE AND SHIFT GROUP

RLCA

Operation:



Format:

Opcode **Operands**

RLCA

0	0	0	0	0	1	1	1	07

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are

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

after the execution of

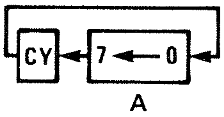
RLCA

the contents of the Accumulator and Carry Flag will be

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

RLA

Operation:



Format:

Opcode **Operands**

RLA

0	0	0	1	0	1	1	1	17

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

If the contents of the Accumulator and the Carry Flag are

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

after the execution of

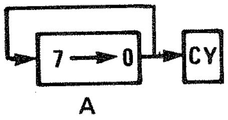
RLA

the contents of the Accumulator and the Carry Flag will be

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

RRCA

Operation:



Format:

Opcode **Operands**

RRCA

0	0	0	0	1	1	1	1

0F

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

If the contents of the Accumulator are

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

After the execution of

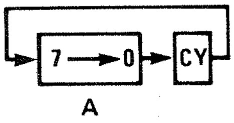
RRCA

the contents of the Accumulator and the Carry Flag will be

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

RRA

Operation:



Format:

Opcode **Operands**

RRA

0	0	0	1	1	1	1	1

1F

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected:

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

Example:

If the contents of the Accumulator and the Carry Flag are

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

after the execution of

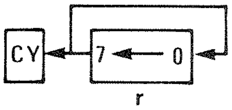
RRA

the contents of the Accumulator and the Carry Flag will be

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

RLC r

Operation:



Format:

Opcode	Operands
RLC	r

1	1	0	0	1	0	1	1	CB

0	0	0	0	0	←	r	→

Description:

The eight-bit contents of register r are rotated left: the content of bit Ø is copied into bit 1; the previous content of bit 1 is copied into bit 2; this pattern is continued throughout the register. The content of bit 7 is copied into the Carry Flag (C flag in register F) and also into bit Ø. Operand r is specified as follows in the assembled object code:

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Note: Bit Ø is the least significant bit.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of register r are

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

after the execution of

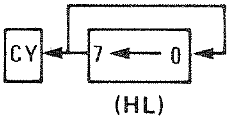
```
RLC r
```

the contents of register r and the Carry Flag will be

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

RLC (HL)

Operation:



Format:

Opcode	Operands
RLC	(HL)

1	1	0	0	1	0	1	1	CB

0	0	0	0	0	1	1	0	06

Description:

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

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

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

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

after the execution of

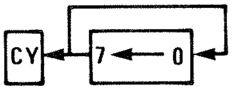
```
RLC (HL)
```

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

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

RLC (IX+d)

Operation:



Format:

Opcode	Operands
	(IX+d)

RLC				(IX+d)				
1	1	0	1	1	1	0	1	DD
1	1	0	0	1	0	1	1	CB
←			d				→	
0	0	0	0	0	1	1	0	06

Description:

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

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of the Index Register IX are 1ØØØH, and the contents of memory location 1Ø22H are

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

after the execution of

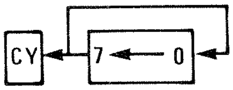
```
RLC (IX+2H)
```

the contents of memory location 1ØØ2H and the Carry Flag will be

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

RLC (IY+d)

Operation:



Format:

Opcode	Operands
	(IY+d)

RLC				(IY+d)				
1	1	1	1	1	1	0	1	FD
1	1	0	0	1	0	1	1	CB
←			d				→	
0	0	0	0	0	1	1	0	06

Description:

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

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of the Index Register IY are 1ØØØH, and the contents of memory location 1ØØ2H are

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

after the execution of

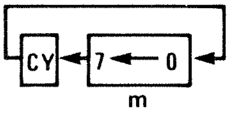
```
RLC (IY+2H)
```

the contents of memory location 1ØØ2H and the Carry Flag will be

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

RL m

Operation:



Format:

Opcode	Operands
RL	m

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

RL r	1	1	0	0	1	0	1	1	CB
	0	0	0	1	0	←	r	→	
RL (HL)	1	1	0	0	1	0	1	1	CB
	0	0	0	1	0	1	1	0	16
RL (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	←			d				→	
	0	0	0	1	0	1	1	0	16
RL (IY+d)	1	1	1	1	1	1	0	1	FD
	1	1	0	0	1	0	1	1	CB
	←			d				→	
	0	0	0	1	0	1	1	0	16

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	Ø11
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RL r	2	8(4,4)	2.ØØ
RL (HL)	4	15(4,4,4,3)	3.75
RL (IX+d)	6	23(4,4,3,5,4,3)	5.75
RL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of register D and the Carry Flag are

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

after the execution of

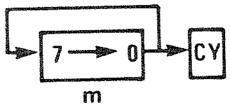
RL D

the contents of register D and the Carry Flag will be

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

RRC m

Operation:



Format:

Opcode	Operands
RRC	m

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

RRC r	1 1 0 0 1 0 1 1	CB
	0 0 0 0 1 ←r→	
RRC (HL)	1 1 0 0 1 0 1 1	CB
	0 0 0 0 1 1 1 0	OE
RRC (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 0 0 1 1 1 0	OE
RRC (IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 0 0 1 1 1 0	OE

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RRC r	2	8(4,4)	2.ØØ
RRC (HL)	4	15(4,4,4,3)	3.75
RRC (IX+d)	6	23(4,4,3,5,4,3)	5.75
RRC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Reset
P/V: Set if parity even; reset otherwise
N: Reset
C: Data from Bit Ø of source register

Example:

If the contents of register A are

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

after the execution of

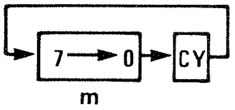
RRC A

the contents of register A and the Carry Flag will be

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

RR m

Operation:



Format:

Opcode	**Operand**
RR	m

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

RR r	1	1	0	0	1	0	1	1	CB
	0	0	0	1	1	←	r	→	
RR (HL)	1	1	0	0	1	0	1	1	CB
	0	0	0	1	1	1	1	0	1E
RR (IX+d)	1	1	0	1	1	1	0	1	DD
	1	1	0	0	1	0	1	1	CB
	←			d				→	
	0	0	0	1	1	1	1	0	1E
RR (IY+d)	0	0	0	1	1	1	1	0	1E
	1	1	0	0	1	0	1	1	CB
	←			d				→	
	0	0	0	1	1	1	1	0	1E

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

Register	**r**
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	**M CYCLES**	**T STATES**	**4 MHZ E.T.**
RR r	2	8(4,4)	2.ØØ
RR (HL)	4	15(4,4,4,3)	3.75
RR (IX+d)	6	23(4,4,3,5,4,3)	5.75
RR (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit Ø of source register

Example:

If the contents of the HL register pair are 4343H, and the contents of memory location 4343H and the Carry Flag are

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

after the execution of

RR (HL)

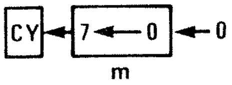
the contents of location 4343H and the Carry Flag will be

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

SLA m

Operation:

Format:



Opcode	Operands
SLA	m

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

SLA r	1 1 0 0 1 0 1 1	CB
	0 0 1 0 0 ←r→	
SLA (HL)	1 1 0 0 1 0 1 1	CB
	0 0 1 0 0 1 1 0	26
SLA (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 1 0 0 1 1 0	26
SLA (IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 1 0 0 1 1 0	26

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SLA r	2	8(4,4)	2.ØØ
SLA (HL)	4	15(4,4,4,3)	3.75
SLA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SLA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit 7

Example:

If the contents of register L are

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

after the execution of

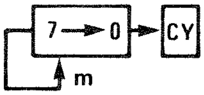
```
SLA L
```

the contents of register L and the Carry Flag will be

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

SRA m

Operation:



Format:

Opcode	Operands
SRA	m

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

SRA r	1 1 0 0 1 0 1 1	CB
	0 0 1 0 1 ←r→	
SRA(HL)	1 1 0 0 1 0 1 1	CB
	0 0 1 0 1 1 1 0	2E
SRA (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 1 0 1 1 1 0	2E
SRA(IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 1 0 1 1 1 0	2E

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

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SRA r	2	8(4,4)	2.ØØ
SRA (HL)	4	15(4,4,4,3)	3.75
SRA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit Ø of source register

Example:

If the contents of the Index Register IX are 1ØØØH, and the contents of memory location 1ØØ3H are

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

after the execution of

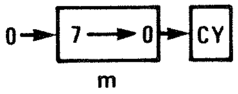
```
SRA (IX+3H)
```

the contents of memory location 1ØØ3H and the Carry Flag will be

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

SRL m

Operation:



Format:

Opcode	**Operands**
SRL	m

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

SRL r	1 1 0 0 1 0 1 1	CB
	0 0 1 1 1 ←r→	
SRL (HL)	1 1 0 0 1 0 1 1	CB
	0 0 1 1 1 1 1 0	3E
SRL (IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 1 1 1 1 1 0	3E
SRL (IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	←d→	
	0 0 1 1 1 1 1 0	3E

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

Register	**r**
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

Description:

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

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SRL r	2	8(4,4)	2.ØØ
SRL (HL)	4	15(4,4,4,3)	3.75
SRL (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit Ø of source register

Example:

If the contents of register B are

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

after the execution of

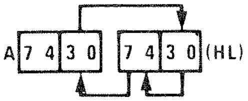
SRL B

the contents of register B and the Carry Flag will be

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

RLD

Operation:



Format:

Opcode **Operands**

RLD

1	1	1	0	1	1	0	1	ED
0	1	1	0	1	1	1	1	6F

Description:

The contents of the low order four bits (bits 3,2,1 and ∅) of the memory location (HL) are copied into the high order four bits (7,6,5 and 4) of that same memory location; the previous contents of those high order four bits are copied into the low order four bits of the Accumulator (register A), and the previous contents of the low order four bits of the Accumulator are copied into the low order four bits of memory location (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

M CYCLES: 5 T STATES: 18(4,4,3,4,3) 4 MHZ E.T.: 4.5∅

Condition Bits Affected:

S: Set if Acc. is negative after operation; reset otherwise
Z: Set if Acc. is zero after operation; reset otherwise
H: Reset
P/V: Set if parity of Acc. is even after operation; reset otherwise
N: Reset
C: Not affected

Example:

If the contents of the HL register pair are 5∅∅∅H, and the contents of the Accumulator and memory location 5∅∅∅H are

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

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

after the execution of

RLD

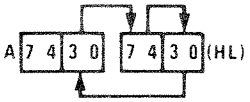
the contents of the Accumulator and memory location 5∅∅∅H will be

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

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

RRD

Operation:



Format:

Opcode **Operands**

RRD

1	1	1	0	1	1	0	1	ED
0	1	1	0	0	1	1	1	67

Description:

The contents of the low order four bits (bits 3,2,1 and Ø) of memory location (HL) are copied into the low order four bits of the Accumulator (register A); the previous contents of the low order four bits of the Accumulator are copied into the high order four bits (7,6,5 and 4) of location (HL); and the previous contents of the high order four bits of (HL) are copied into the low order four bits of (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

M CYCLES: 5 T STATES: 18(4,4,3,4,3) 4 MHZ E.T.: 4.5Ø

Condition Bits Affected:

S:	Set if Acc. is negative after operation; reset otherwise
Z:	Set if Acc, is zero after operation; reset otherwise
H:	Reset
P/V:	Set if parity of Acc. is even after operation; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of the HL register pair are 5ØØØH, and the contents of the Accumulator and memory location 5ØØØH are

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

7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	0	(5ØØØH)

after the execution of

RRD

the contents of the Accumulator and memory location 5ØØØH will be

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

7	6	5	4	3	2	1	0	
0	1	0	0	0	0	1	0	(5ØØØH)

BIT SET, RESET AND TEST GROUP

BIT b, r

Operation: $Z \leftarrow \overline{r}_b$

Format:

Opcode	Operands
BIT	b,r

1	1	0	0	1	0	1	1	CB

0	1	←b→	←r→

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the indicated register. Operands b and r are specified as follows in the assembled object code:

Bit Tested	b	Register	r
Ø	ØØØ	B	ØØØ
1	ØØ1	C	ØØ1
2	Ø1Ø	D	Ø1Ø
3	Ø11	E	Ø11
4	1ØØ	H	1ØØ
5	1Ø1	L	1Ø1
6	11Ø	A	111
7	111		

M CYCLES: 2   T STATES: 8(4,4)   4 MHZ E.T.: 2.ØØ

Condition Bits Affected:

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

Example:

If bit 2 in register B contains Ø, after the execution of

```
BIT 2,B
```

the Z flag in the F register will contain 1, and bit 2 in register B will remain Ø. Bit Ø in register B is the least significant bit.

BIT b, (HL)

Operation: $Z \leftarrow \overline{(HL)}_b$

Format:

Opcode	Operands
BIT	b,(HL)

1	1	0	0	1	0	1	1	CB

0	1	←b→	1	1	0

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the HL register pair. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M CYCLES: 3   T STATES: 12(4,4,4)   4 MHZ E.T.: 3.ØØ

Condition Bits Affected:

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

Example:

If the HL register pair contains 4444H, and bit 4 in the memory location 444H contains 1, after the execution of

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

the Z flag in the F register will contain Ø, and bit 4 in memory location 444H will still contain 1. (Bit Ø in memory location 444H is the least significant bit.)

BIT b, (IX+d)

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

Format:

Opcode	Operands
BIT	b,(IX+d)

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

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

| ←――――――――― d ―――――――――→ |

| 0 | 1 | ←― b ―→ | 1 | 1 | 0 |
```

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents register pair IX (Index Register IX) and the two's complement displacement integer d. Operand b is specified as follows in the assembled object code.

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M CYCLES: 5 T STATES: 2Ø(4,4,3,5,4) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected:

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

Example:

If the contents of Index Register IX are 2ØØØH, and bit 6 in memory location 2ØØ4H contains 1, after the execution of

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

the Z flag in the F register will contain Ø, and bit 6 in memory location 2ØØ4H will still contain 1. (Bit Ø in memory location 2ØØ4H is the least significant bit.)

BIT b, (IY+d)

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

Format:

Opcode	Operands
BIT	b,(IY+d)

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

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

| ←――――――――― d ―――――――――→ |

| 0 | 1 | ←― b ―→ | 1 | 1 | 0 |
```

Description:

After the execution of this instruction, the Z flag in the F register will contain the complement of the indicated bit within the contents of the memory location pointed to by the sum of the contents of register pair IY (Index Register IY) and the two's complement displacement integer d. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M CYCLES: 5 T STATES: 2Ø(4,4,3,5,4) 4 MHZ E.T.: 5.ØØ

Condition Bits Affected:

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

Example:

If the contents of Index Register are 2ØØØH, and bit 6 in memory location 2ØØ4H contains 1, after the execution of

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

the Z flag in the F register still contain Ø, and bit 6 in memory location 2ØØ4H will still contain 1. (Bit Ø in memory location 2ØØ4H is the least significant bit.)

SET b, r

Operation: $r_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,r

1	1	0	0	1	0	1	1	CB

1	1	←	b	→	←	r	→

Description:

Bit b (any bit, 7 through Ø) in register r (any of register B,C,D,E,H,L or A) is set. Operands b and r are specified as follows in the assembled object code:

Bit	b	Register	r
Ø	ØØØ	B	ØØØ
1	ØØ1	C	ØØ1
2	Ø1Ø	D	Ø1Ø
3	Ø11	E	Ø11
4	1ØØ	H	1ØØ
5	1Ø1	L	1Ø1
6	11Ø	A	111
7	111		

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected: None

Example:

After the execution of

```
SET 4,A
```

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

SET b, (HL)

Operation: $(HL)_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,(HL)

1	1	0	0	1	0	1	1	CB

1	1	←	b	→	1	1	0

Description:

Bit b (any bit, 7 through Ø) in the memory location addressed by the contents of register pair HL is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the contents of the HL register pair are 3ØØØH, after the execution of

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

bit 4 in memory location 3ØØØH will be 1. (Bit Ø in memory location 3ØØØH is the least significant bit.)

SET b, (IX+d)

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

Format:

Opcode	Operands
SET	b,(IX+d)

1 1 0 1 1 1 0 1 DD

1 1 0 0 1 0 1 1 CB

←d→

1 1 ←b→ 1 1 0

Description:

Bit b (any bit, 7 through Ø) in the memory location addressed by the sum of the contents of the IX register pair (Index Register IX) and the two's complement integer d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register are 2ØØØH, after the execution of

```
SET Ø,(IX+3H)
```

bit Ø in memory location 2ØØ3H will be 1. (Bit Ø in memory location 2ØØ3H is the least significant bit.)

SET b, (IY+d)

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

Format:

Opcode	Operands
SET	b,(IY+d)

1 1 1 1 1 1 0 1 FD

1 1 0 0 1 0 1 1 CB

←d→

1 1 ←b→ 1 1 0

Description:

Bit b (any bit, 7 through Ø) in the memory location addressed by the sum of the contents of the IY register pair (Index Register IY) and the two's complement displacement d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
Ø	ØØØ
1	ØØ1
2	Ø1Ø
3	Ø11
4	1ØØ
5	1Ø1
6	11Ø
7	111

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register IY are 2ØØØH, after the execution of

```
SET Ø,(IY+3H)
```

bit Ø in memory location 2ØØ3H will be 1. (Bit Ø in memory location 2ØØ3H is the least significant bit.)

RES b, m

Operation: $s_b \leftarrow Ø$

Format:

Opcode	Operands
RES	b,m

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

RES b,r	1 1 0 0 1 0 1 1	CB
	1 0 ←b→ ←r→	
RES b,(HL)	1 1 0 0 1 0 1 1	CB
	1 0 ←b→ 1 1 0	
RES b,(IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	←d→	
	1 0 ←b→ 1 1 0	
RES b,(IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	←d→	
	1 0 ←b→ 1 1 0	

Bit Reset	b	Register	r
Ø	ØØØ	B	ØØØ
1	ØØ1	C	ØØ1
2	Ø1Ø	D	Ø1Ø
3	Ø11	E	Ø11
4	1ØØ	H	1ØØ
5	1Ø1	L	1Ø1
6	11Ø	A	111
7	111		

Description:

Bit b in operand m is reset.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RES r	4	8(4,4)	2.ØØ
RES (HL)	4	15(4,4,4,3)	3.75
RES (IX+d)	6	23(4,4,3,5,4,3)	5.75
RES (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected: None

Example:

After the execution of

```
RES 6,D
```

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

JUMP GROUP

JP nn

Operation: PC ← nn

Format:

Opcode	Operands
JP	nn

1	1	0	0	0	0	1	1	C3

←— n —→
←— n —→

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

Description:

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

M CYCLES: 3 T STATES: 1Ø(4,3,3) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

JP cc, nn

Operation: IF cc TRUE, PC ← nn

Format:

Opcode	Operands
JP	cc, nn

1	1	←— cc —→	0	1	0

←— n —→
←— n —→

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

Description:

If condition cc is true, the instruction loads operand nn into register pair PC (Program Counter), and the program continues with the instruction beginning at address nn. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status which corresponds to condition bits in the Flag Register (register F). These eight status are defined in the table below which also specifies the corresponding cc bit fields in the assembled object code.

cc	CONDITION	RELEVANT FLAG
ØØØ	NZ non zero	Z
ØØ1	Z zero	Z
Ø1Ø	NC no carry	C
Ø11	C carry	C
1ØØ	PO parity odd	P/V
1Ø1	PE parity even	P/V
11Ø	P sign positive	S
111	M sign negative	S

M CYCLES: 3 T STATES: 1Ø(4,3,3) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

If the Carry Flag (C flag in the F register) is set and the contents of address 152Ø are Ø3H, after the execution of

```
JP C,152ØH
```

the Program Counter will contain 152ØH, and on the next machine cycle the CPU will fetch from address 152ØH the byte Ø3H.

JR e

Operation: PC ← PC + e

Format:

Opcode	Operand
JR	e

0	0	0	1	1	0	0	0	18

← e-2 →

Description:

This instruction provides for unconditional branching to other segments of a program. The value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. This jump is measured from the address of the instruction opcode and has a range of –126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

M CYCLES: 3   T STATES: 12(4,3,5)   4 MHZ E.T.: 3.ØØ

Condition Bits Affected: None

Example:

To jump forward 5 locations from address 48Ø, the following assembly language statement is used:

```
JR  $+5
```

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

Location	Instruction
48Ø	18
481	Ø3
482	—
483	—
484	—
485	←PC after jump

JR C, e

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

Format:

Opcode	Operands
JR	C,e

0	0	1	1	1	0	0	0	38

← e-2 →

Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to a '1', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of –126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a 'Ø', the next instruction to be executed is taken from the location following this instruction.

If condition is met:

M CYCLES: 3   T STATES: 12(4,3,5)   4 MHZ E.T.:3.ØØ

If condition is not met:

M CYCLES: 2   T STATES: 7(4,3)   4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is set and it is required to jump back 4 locations from 48Ø. The assembly language statement is:

```
JR  C,$-4
```

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

Location	Instruction
47C	← PC after jump
47D	—
47E	—
47F	—
48Ø	38
481	FA (2's complement–6)

JR NC, e

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

Format:

Opcode	Operands
JR	NC,e

0	0	1	1	0	0	0	0	30

← e-2 →

Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to 'Ø', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of −126 to +129 byte. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a '1', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.ØØ

If the condition is not met:

M CYCLES: 7 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is reset and it is required to repeat the jump instruction. The assembly language statement is:

```
JR  NC,$
```

The resulting object code and PC after the jump are shown below:

Location	Instruction
48Ø	3Ø ← PC after jump
481	ØØ

JR Z, e

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

Format:

Opcode	Operands
JR	Z,e

0	0	1	0	1	0	0	0	28

← e-2 →

Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '1', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of −126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a 'Ø', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.ØØ

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is set and it is required to jump forward 5 locations from address 3ØØ. The following assembly language statement is used:

```
JR  Z,$  +5
```

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

Location	Instruction
3ØØ	28
3Ø1	Ø3
3Ø2	—
3Ø3	—
3Ø4	—
3Ø5	← PC after jump

JR NZ, e

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

Format:

Opcode	Operands
JR	NZ,e

0	0	1	0	0	0	0	0	20

← e-2 →

Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '0', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of –126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '1', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.ØØ

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is reset and it is required to jump back 4 locations from 48Ø. The assembly language statement is:

```
JR  NZ,$-4
```

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

Location	Instruction
47C	← PC after jump
47D	—
47E	—
47F	—
48Ø	2Ø
481	FA (2' complement–6)

JP (HL)

Operation: PC ← HL

Format:

Opcode	Operands
JP	(HL)

1	1	1	0	1	0	0	1	E9

Description:

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

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.ØØ

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1ØØØH and the contents of the HL register pair are 48ØØH, after the execution of

```
JP  (HL)
```

the contents of the Program Counter will be 48ØØH.

JP (IX)

Operation: PC ← IX

Format:

Opcode	Operands
JP	(IX)

1	1	0	1	1	1	0	1	DD
1	1	1	0	1	0	0	1	E9

Description:

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

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1ØØØH, and the contents of the IX Register Pair are 48ØØH, after the execution of

```
JP (IX)
```

the contents of the Program Counter will be 48ØØH.

JP (IY)

Operation: PC ← IY

Format:

Opcode	Operands
JP	(IY)

1	1	1	1	1	1	0	1	FD
1	1	1	0	1	0	0	1	E9

Description:

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

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1ØØØH and the contents of the IY Register Pair are 48ØØH, after the execution of

```
JP (IY)
```

the contents of the Program Counter will be 48ØØH.

DJNZ, e

Operation: ——

Format:

Opcode	**Operands**
DJNZ	e

```
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |   10

| ←———————— e-2 ————————→ |
```

Description:

The instruction is similar to the conditional jump instructions except that a register value is used to determine branching. The B register is decremented and if a non zero value remains, the value of the displacement e is added to the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of –126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the result of decrementing leaves B with a zero value, the next instruction to be executed is taken from the location following this instruction.

If B≠Ø:

M CYCLES: 3 T STATES: 13(5,3,5) 4 MHZ E.T.: 3.25

If B=Ø:

M CYCLES: 2 T STATES: 8(5,3) 4 MHZ E.T.: 2.ØØ

Condition Bits Affected: None

Example:

A typical software routine is used to demonstrate the use of the DJNZ instruction. This routine moves a line from an input buffer (INBUF) to an output buffer (OUTBUF). It moves the bytes until it finds a CR, or until it has moved 80 bytes, whichever occurs first.

```
        LD      B,8Ø            ;Set up counter
        LD      HL,Inbuf        ;Set up pointers
        LD      DE,Outbuf

LOOP:   LD      A,(HL)          ;Get next byte from
                                ;input buffer
        LD      (DE),A          ;Store in output buffer
        CP      ØØH             ;Is it a CR?
        JR      Z,DONE          ;Yes finished
        INC     HL              ;Increment pointers
        INC     DE
        DJNZ    LOOP            ;Loop back if 8Ø
                                ;bytes have not
                                ;been moved
DONE:
```

CALL AND RETURN GROUP

CALL nn

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

Format:

Opcode	Operands
CALL	nn

```
| 1  1  0  0  1  1  0  1 |   CD

|←———————— n ————————→|

|←———————— n ————————→|
```

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

Description:

After pushing the current contents of the Program Counter (PC) onto the top of the external memory stack, the operands nn are loaded into PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into PC.) The push is accomplished by first decrementing the current contents of the Stack Pointer (register pair SP), loading the high-order byte of the PC contents into the memory address now pointed to by the SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before the push is executed.

M CYCLES: 5 T STATES: 17(4,3,4,3,3) 4 MHZ E.T.: 4.25

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3ØØ2H, and memory locations have the contents:

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

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

```
CALL 2135H
```

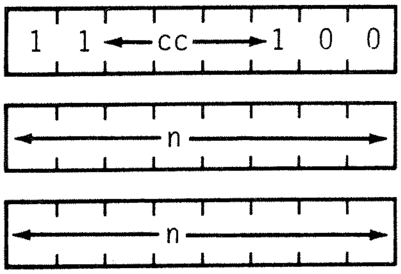
After the execution of this instruction, the contents of memory address 3ØØ1H will be 1AH, the contents of address 3ØØØH will be 4AH, the contents of the Stack Pointer will be 3ØØØH, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

CALL cc, nn

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

Format:

Opcode	Operands
CALL	cc,nn



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

Description:

If condition cc is true, this instruction pushes the current contents of the Program Counter (PC) onto the top of the external memory stack, then loads the operands nn into PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into PC.) If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. The stack push is accomplished by first decrementing the current contents of the Stack Pointer (SP), loading the high-order byte of the PC contents into the memory address now pointed to by SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of the stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before the push is executed. Condition cc is programmed as one of eight status which corresponds to condition bits in the Flag Register (register F). Those eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code:

cc	Condition	Relevant Flag
ØØØ	NZ non zero	Z
ØØ1	Z zero	Z
Ø1Ø	NC non carry	C
Ø11	C carry	C
1ØØ	PO parity odd	P/V
1Ø1	PE parity even	P/V
11Ø	P sign positive	S
111	M sign negative	S

If cc is true:

M CYCLES: 5 T STATES: 17(4,3,4,3,3) 4 MHZ E.T.: 4.25

If cc is false:

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

If the C Flag in the F register is reset, the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3ØØ2H, and memory locations have the contents:

Location	Contents
1A47H	D4H
1A48H	35H
1A49H	21H

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

```
CALL NC,2135H
```

After the execution of this instruction, the contents of memory address 3ØØ1H will be 1AH, the contents of address 3ØØØH will be 4AH, the contents of the Stack Pointer will be 3ØØØH, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

RET

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

Format:

Opcode

RET

1	1	0	0	1	0	0	1	C9

Description:

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

M CYCLES: 3 T STATES: 1Ø(4,3,3) 4 MHZ E.T.: 2.5Ø

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2ØØØH, the contents of memory location 2ØØØH are B5H, and the contents of memory location 2ØØ1H are 18H, then after the execution of

RET

the contents of the Stack Pointer will be 2ØØ2H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

RET cc

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

Format:

Opcode	**Operand**
RET	cc

1	1	←cc→	0	0	0

Description:

If condition cc is true, control is returned to the original program flow by popping the previous contents of the Program Counter (PC) off the top of the external memory stack, where they were pushed by the CALL instruction. This is accomplished by first loading the low-order byte of the PC with the contents of the memory address pointed to by the Stack Pointer (SP), then incrementing the SP, and loading the high-order byte of the PC with the contents of the memory address now pointed to by the SP. (The SP is now incremented a second time.) On the following machine cycle the CPU will fetch the next program opcode from the location in memory now pointed to by the PC. If condition cc is false, the PC is simply incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status which correspond to condition bits in the Flag Register (register F). These eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code.

cc	Condition	Relevant Flag
ØØØ	NZ non zero	Z
ØØ1	Z zero	Z
Ø1Ø	NC non carry	C
Ø11	C carry	C
1ØØ	PO parity odd	P/V
1Ø1	PE parity even	P/V
11Ø	P sign positive	S
111	M sign negative	S

If cc is true:

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

If cc is false:

M CYCLES: 1 T STATES: 5 4 MHZ E.T.; 1.25

Condition Bits Affected: None

Example:

If the S flag in the F register is set, the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2ØØØH, the contents of memory location 2ØØØH are B5H, and the contents of memory location 2ØØ1H are 18H, then after the execution of

```
RET M
```

the contents of the Stack Pointer will be 2ØØ2H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

RETI

Operation: Return from interrupt

Format:

Opcode

RETI

1	1	1	0	1	1	0	1	ED
0	1	0	0	1	1	0	1	4D

Description:

This instruction is used at the end of an interrupt service routine to:

1. Restore the contents of the Program Counter (PC) (analogous to the RET instruction).

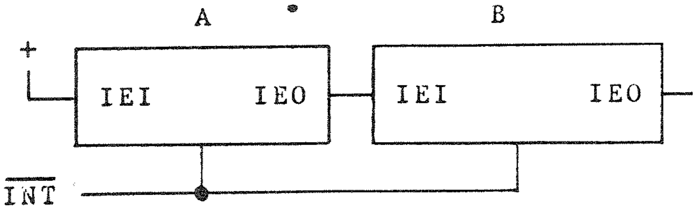
2. To signal an I/O device that the interrupt routine has been completed. The RETI instruction facilitates the nesting of interrupts allowing higher priority devices to suspend service of lower priority service routines. This instruction also resets the IFF1 and IFF2 flip flops.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.5Ø

Condition Bits Affected: None

Example:

Given: Two interrupting devices, A and B connected in a daisy chain configuration with A having a higher priority than B.



B generates an interrupt and is acknowledged. (The interrupt enable out, IEO, of B goes low, blocking any lower priority devices from interrupting while B is being serviced). Then A generates an interrupt, suspending service of B. (The IEO of A goes 'low' indicating that a higher priority device is being serviced.) The A routine is completed and a RETI is issued resetting the IEO of A, allowing the B routine to continue. A second RETI is issued on completion of the B routine and the IEO of B is reset (high) allowing lower priority devices interrupt access.

RETN

Operation: Return from non maskable interrupt

Format:

Opcode

RETN

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	0	1	45

Description:

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

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.5Ø

Condition Bits Affected: None

Example:

If the contents of the Stack Pointer are 1ØØØH and the contents of the Program Counter are 1A45H when a non maskable interrupt (NMI) signal is received, the CPU will ignore the next instruction and will instead restart to memory address ØØ66H. That is, the current Program Counter contents of 1A45H will be pushed onto the external stack address of OFFFH and OFFEH, high order-byte first, and ØØ66H will be loaded onto the Program Counter. That address begins an interrupt service routine which ends with RETN instruction. Upon the execution of RETN, the former Program Counter contents are popped off the external memory stack, low-order first, resulting in a Stack Pointer contents again of 1ØØØH. The program flow continues where it left off with an opcode fetch to address 1A45H.

RST p

Operation:

$(SP-1) \leftarrow PC_H, (SP-2) \leftarrow PC_L, PC_H \leftarrow O, PC_L \leftarrow P$

Format:

Opcode	**Operand**
RST	P

1	1	←——	t	——→	1	1	1

Description:

The current Program Counter (PC) contents are pushed onto the external memory stack, and the page zero memory location given by operand p is loaded into the PC. Program execution then begins with the opcode in the address now pointed to by PC. The push is performed by first decrementing the contents of the Stack Pointer (SP), loading the high-order byte of PC into the memory address now pointed to by SP, decrementing SP again, and loading the low-order byte of PC into the address now pointed to by SP. The ReSTart instruction allows for a jump to one of eight addresses as shown in the table below. The operand p is assembled into the object code using the corresponding T state. Note: Since all addresses are in page zero of memory, the high order byte of PC is loaded with ØØH. The number selected from the "p" column of the table is loaded into the low-order byte of PC.

P	t
ØØH	ØØØ
Ø8H	ØØ1
1ØH	Ø1Ø
18H	Ø11
2ØH	1ØØ
28H	1Ø1
3ØH	11Ø
38H	111

M CYCLES: 3   T STATES: 11(5,3,3)   4 MHZ E.T.: 2.75

Example:

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

RST 18H   (Object code 11Ø1111)

the PC will contain ØØ18H, as the address of the next opcode to be fetched.

INPUT AND OUTPUT GROUP

IN A, (n)

Operation: A ← (n)

Format:

Opcode	Operands
IN	A,(n)

1	1	0	1	1	0	1	1	DB
←			n				→	

Description:

The operand n is placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also appear on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into the Accumulator (register A) in the CPU.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H and the byte 7BH is available at the peripheral device mapped to I/O port address Ø1H, then after the execution of

```
IN A,(Ø1H)
```

the Accumulator will contain 7BH.

IN r, (C)

Operation: r ← (C)

Format:

Opcode •	Operands
IN	r,(C)

1	1	1	0	1	1	0	1	ED
0	1	←	r	→	0	0	0	

Description:

The contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into register r in the CPU. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each. The flags will be affected, checking the input data.

Reg.	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.ØØ

Condition Bits Affected:

S:	Set if input data is negative; reset otherwise
Z:	Set if input data is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of register C are Ø7H, the contents of register B are 1ØH, and the byte 7BH is available at the peripheral device mapped to I/O port address Ø7H, then after the execution of

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

register D will contain 7BH

INI

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

Format:

Opcode

INI

1	1	1	0	1	1	0	1	ED
1	0	1	0	0	0	1	0	A2

Description:

The contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are then placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter is decremented and register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S:	Unknown
Z:	Set if B–1=Ø; reset otherwise
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are Ø7H, the contents of register B are 1ØH, the contents of the HL register pair are 1ØØØH, and the byte 7BH is available at the peripheral device mapped to I/O port address Ø7H, then after the execution of

```
INI
```

memory location 1ØØØH will contain 7BH, the HL register pair will contain 1ØØ1H, and register B will contain ØFH.

INIR

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

Format:

Opcode

INIR

1	1	1	0	1	1	0	1	ED

1	0	1	1	0	0	1	0	B2

Description:

The contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then register pair HL is incremented, the byte counter is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also interrupts will be recognized after each data transfer.

If B≠Ø:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=Ø:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

If the contents of register C are Ø7H, the contents of register B are Ø3H, the contents of the HL register pair are 1ØØØH, and the following sequence of bytes are available at the peripheral device mapped to I/O port of address Ø7H:

51H
A9H
Ø3H

then after the execution of

INIR

the HL register pair will contain 1ØØ3H, register B will contain zero, and memory locations will have contents as follows:

Location	Contents
1ØØØH	51H
1ØØ1H	A9H
1ØØ2H	Ø3H

IND

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

Format:

Opcode

IND

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	1	0	AA

Description:

The contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter and register pair HL are decremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S:	Unknown
Z:	Set if B−1=Ø; reset otherwise
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are Ø7H, the contents of register B are 1ØH, the contents of the HL register pair are 1ØØØH, and the byte 7BH is available at the peripheral device mapped to I/O port address Ø7H, then after the execution of

```
IND
```

memory location 1ØØØH will contain 7BH, the HL register pair will contain ØFFFH, and register B will contain ØFH.

INDR

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

Format:

Opcode

INDR

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	1	0	BA

Description:

The contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then HL and the byte counter are decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input. Also interrupts will be recognized after each data transfer.

If B≠Ø:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=Ø:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

If the contents of register C are Ø7H, the contents of register B are Ø3H, the contents of the HL register pair are 1ØØØH, and the following sequence of bytes are available at the peripheral device mapped to I/O port address Ø7H:

51H
A9H
Ø3H

then after the execution of

```
INDR
```

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

Location	**Contents**
ØFFEH	Ø3H
ØFFFH	A9H
1ØØØH	51H

OUT (n), A

Operation: (n) ← A

Format:

Opcode	Operands
OUT	(n) ,A

```
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |   D3

| ←——————————— n ———————————→ |
```

Description:

The operand n is placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator (register A) also appear on the top half (A8 through A15) of the address bus at this time. Then the byte contained in the Accumulator is placed on the data bus and written into the selected peripheral device.

M CYCLES: 3   T STATES: 11(4,3,4)   4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

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

```
OUT Ø1H,A
```

the byte 23H will have been written to the peripheral device mapped to I/O port address Ø1H.

OUT (D), r

Operation: (C) ← r

Format:

Opcode	Operands
OUT	(C),r

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

| 0 | 1 | ←—— r ——→ | 0 | 0 | 1 |
```

Description:

The contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then the byte contained in register r is placed on the data bus and written into the selected peripheral device. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each which appears in the assembled object code:

Register	r
B	ØØØ
C	ØØ1
D	Ø1Ø
E	Ø11
H	1ØØ
L	1Ø1
A	111

M CYCLES: 3   T STATES: 12(4,4,4)   4 MHZ E.T.: 3.ØØ

Condition Bits Affected: None

Example:

If the contents of register C are Ø1H and the contents of register D are 5AH, after the execution of

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

the byte 5AH will have been written to the peripheral device mapped to I/O port address Ø1H.

OUTI

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

Format:

Opcode

OUTI

1	1	1	0	1	1	0	1	ED

1	0	1	0	0	0	1	1	A3

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus. The byte to be output is placed on the data bus and written into selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S:	Unknown
Z:	Set if B–1=Ø; reset otherwise
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are Ø7H, the contents of register B are 1ØH, the contents of the HL register pair are 1ØØØH, and the contents of memory address 1ØØØH are 59H, then after the execution of

```
OUTI
```

register B will contain ØFH, the HL register pair will contain 1ØØ1H, and the byte 59H will have been written to the peripheral device mapped to I/O port address Ø7H.

OTIR

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

Format:

Opcode

OTIR

1	1	1	0	1	1	0	1	ED

1	0	1	1	0	0	1	1	B3

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is incremented. If the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data. Also, interrupts will be recognized after each data transfer.

If B≠Ø:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=Ø:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MYZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

If the contents of register C are Ø7H, the contents of register B are Ø3H, the contents of the HL register pair are 1ØØØH, and memory locations have the following contents:

Location	Contents
1ØØØH	51H
1ØØ1H	A9H
1ØØ2H	Ø3H

then after the execution of

```
OTIR
```

the HL register pair will contain 1ØØ3H, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address Ø7H in the following sequence:

51H
A9H
Ø3H

OUTD

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

Format:

Opcode

OUTD

1	1	1	0	1	1	0	1	ED

1	0	1	0	1	0	1	1	AB

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

S:	Unknown
Z:	Set if B–1=Ø; reset otherwise
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are Ø7H, the contents of register B are 1ØH, the contents of the HL register pair are 1ØØØH, and the contents of memory location 1ØØØH are 59H, after the execution of

```
OUTD
```

register B will contain ØFH, the HL register pair will contain ØFFFH, and the byte 59H will have been written to the peripheral device mapped to I/O port address Ø7H.

OTDR

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

Format:

Opcode

OTDR

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	1	1	BB

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (AØ through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is decremented and if the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Note that if B is set to zero prior to instruction execution, the instruction will output 256 byte of data. Also, interrupts will be recognized after each data transfer.

If B≠0:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=Ø:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.ØØ

Condition Bits Affected:

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

Example:

If the contents of register C are Ø7H, the contents of register B are Ø3H, the contents of the HL register pair are 1ØØØH, and memory locations have the following contents:

Location	Contents
ØFFEH	51H
ØFFFH	A9H
1ØØØH	Ø3H

then after the execution of

```
OTDR
```

the HL register pair will contain ØFFDH, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address Ø7H in the following sequence:

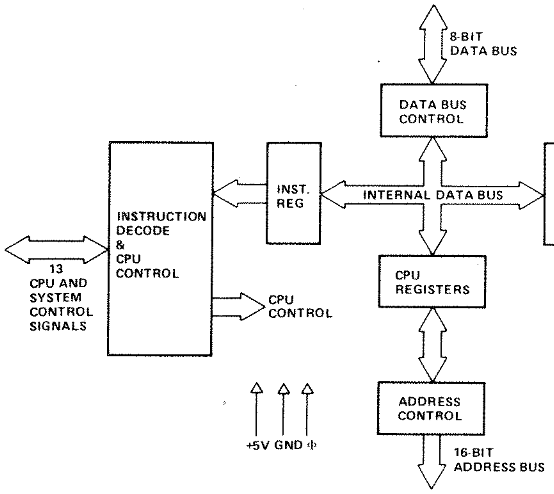
ØЗH
A9H
51H

Z-80 Hardware Configuration

This section gives information about the actual Z80 chip.

Z-80 CPU ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in Figure 1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



Z-80 CPU BLOCK DIAGRAM
FIGURE 1

CPU REGISTERS

The Z-80 CPU contains 2Ø8 bits of R/W memory that are accessible to the programmer. Figure 2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag resistors.

Special Purpose Registers

1. **Program Counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.
2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.

MAIN REG SET		ALTERNATE REG SET		
ACCUMULATOR A	FLAGS F	ACCUMULATOR A'	FLAGS F'	
B	C	B'	C'	GENERAL PURPOSE REGISTERS
D	E	D'	E'	
H	L	H'	L'	

INTERRUPT VECTOR I	MEMORY REFRESH R	SPECIAL PURPOSE REGISTERS
INDEX REGISTER IX		
INDEX REGISTER IY		
STACK POINTER SP		
PROGRAM COUNTER PC		

Z-80 CPU REGISTER CONFIGURATION
FIGURE 2

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

4. **Interrupt Page Address Register (I).** The Z-80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8 bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while

the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

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

General Purpose Registers

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

ARITHMETIC & LOGIC UNIT (ALU)

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

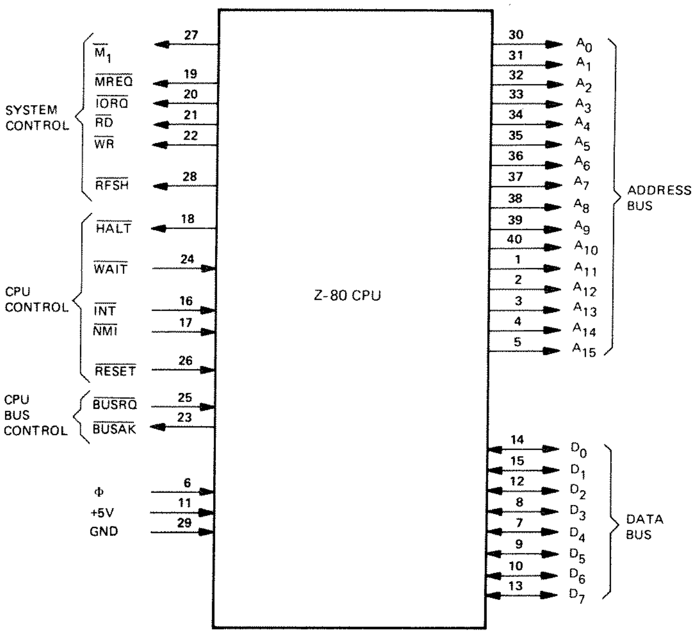
Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

INSTRUCTION REGISTER AND CPU CONTROL

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

Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in Figure 3 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3

$A_\emptyset$-A_{15} (Address Bus) — Tri-state output, active high. $A_\emptyset$-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. $A_\emptyset$ is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

$D_\emptyset$-D_7 (Data Bus) — Tri-state input/output, active high. $D_\emptyset$-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one) — Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M_1}$ is generated as each op-code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M_1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{\text{MREQ}}$ (Memory Request)
Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{IORQ}}$ (Input/Output Request)
Tri-state output, active low. The $\overline{\text{IORQ}}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{\text{IORQ}}$ signal is also generated with an $\overline{\text{MI}}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

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

$\overline{\text{WR}}$ (Memory Write)
Tri-state output, active low. $\overline{\text{WR}}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{\text{RFSH}}$ (Refresh)
Output, active low. $\overline{\text{RFSH}}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{\text{MREQ}}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{\text{HALT}}$ (Halt state)
Output, active low. $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{\text{WAIT}}$ (Wait)
Input, active low. $\overline{\text{WAIT}}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.

$\overline{\text{INT}}$ (Interrupt Request)
Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$ during M_1 time) is sent out at the beginning of the next instruction cycle.

$\overline{\text{NMI}}$ (Non Maskable Interrupt)
Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z-80 CPU to restart to location $\emptyset\emptyset 66_H$. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{\text{BUSRQ}}$ will override a $\overline{\text{NMI}}$.

$\overline{\text{RESET}}$
Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = $\emptyset\emptyset_H$
3) Set Register R = $\emptyset\emptyset_H$
4) Set Interrupt Mode Ø

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{\text{BUSRQ}}$ (Bus Request)
Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$ (Bus Acknowledge)
Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ
Single phase TTL level clock which requires only a 33Ø ohm pull-up resistor to +5 volts to meet all clock requirements.

Z-80 CPU INSTRUCTION SET

The Z-80 CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

INTRODUCTION TO INSTRUCTION TYPES

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

A unique set of block transfer instructions is provided in the Z-80. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z-80 block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found or the end of the block is reached, the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

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

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

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of call is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

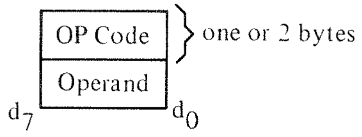
The input/output group of instructions in the Z-80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z-80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z-80 CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z-80 CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

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

ADDRESSING MODES

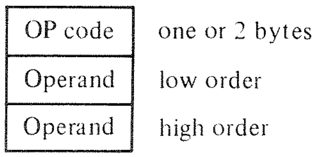
Most of the Z-80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z-80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



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

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the OP codes are the operand.



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

Modified Page Zero Addressing. The Z-80 has a special single byte CALL instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

OP Code	one byte

b_7 b_0 Effective address is $(b_5\ b_4\ b_3\ 000)_2$

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

OP Code	Jump relative (one byte OP code)
Operand	8-bit two's complement displacement added to Address (A+2)

The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signed displacement can range between +127 and -128 from A + 2. This allows for a total displacement of +129 to -126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

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

OP Code	one or two bytes
Low Order Address or Low order operand	
High Order Address or high order operand	

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

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (12ØØ) refers to the contents of memory at location 12ØØ.

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

OP Code	two byte OP code
OP Code	
Displacement	Operand added to index register to form a pointer to memory.

An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z-80 are referred to as IX and IY. To indicate indexed addressing the notation:

(IX+d) or (IY+d)

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

Register Addressing. Many of the Z-80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

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

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

OP Code	one or two bytes

An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z-80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the low order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z-80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

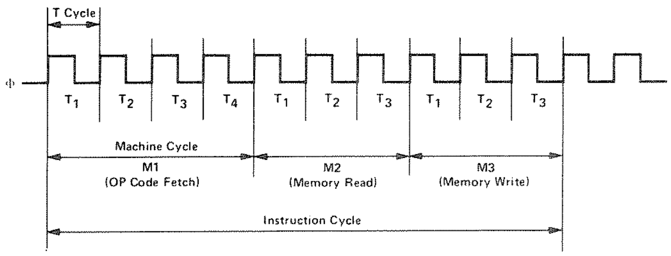
Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

CPU TIMING

The Z-80 CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write
I/O device read or write
Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T cycles and the basic operations are referred to as M (for machine) cycles. Figure 4 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T cycles long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 10, the exact timing for each instruction is specified.



BASIC CPU TIMING EXAMPLE
FIGURE 4

NUMERIC LIST OF INSTRUCTION SET

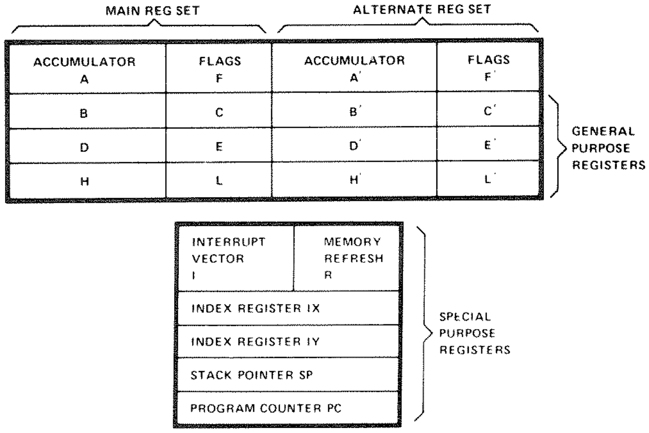
```
                              Z-80 CROSS ASSEMBLER    VERSION 1.06 OF 06/18/76
07/09/76  10:20:50                            OPCODE LISTING
LOC     OBJ CODE     STMT SOURCE STATEMENT

0000    00            1         NOP
0001    018405        2         LD BC,NN
0004    02            3         LD (BC),A
0005    03            4         INC BC
0006    04            5         INC B
0007    05            6         DEC B
0008    0620          7         LD B,N
000A    07            8         RLCA
000B    08            9         EX AF,AF'
000C    09           10         ADD HL,BC
000D    0A           11         LD A,(BC)
000E    0B           12         DEC BC
000F    0C           13         INC C
0010    OD           14         DEC C
0011    0E20         15         LD C,N
0013    0F           16         RRCA
0014    102E         17         DJNZ DIS
0016    118405       18         LD DE,NN
0019    12           19         LD (DE),A
001A    13           20         INC DE
001B    14           21         INC D
001C    15           22         DEC D
001D    1620         23         LD D,N
001F    17           24         RLA
0020    182E         25         JR DIS
0022    19           26         ADD HL,DE
0023    1A           27         LD A,(DE)
0024    1B           28         DEC DE
0025    1C           29         INC E
0026    ID           30         DEC E
0027    1E20         31         LD E,N
0029    1F           32         RRA
002A    202E         33         JR NZ,DIS
002C    218405       34         LD HL,NN
002F    228405       35         LD (NN),HL
0032    23           36         INC HL
0033    24           37         INC H
0034    25           38         DEC H
0035    2620         39         LD H,N
0037    27           40         DAA
0038    282E         41         JR Z,DIS
003A    29           42         ADD HL,HL
003B    2A8405       43         LD HL,(NN)
003E    2B           44         DEC HL
003F    2C           45         INC L
0040    2D           46         DEC L
0041    2E20         47         LD L,N
0043    2F           48         CPL
0044    302E         49         JR NC,DIS
0046    318405       50         LD SP,NN
0049    328405       51         LD (NN),A
004C    33           52         INC SP
004D    34           53         INC (HL)
004E    35           54         DEC (HL)
004F    3620         55         LD (HL),N
0051    37           56         SCF
0052    382E         57         JR C,DIS
0054    39           58         ADD HL,SP
0055    3A8405       59         LD A,(NN)
0058    3B           60         DEC SP
0059    3C           61         INC A
005A    3D           62         DEC A
005B    3E20         63         LD A,N
005D    3F           64         CCF
005E    40           65         LD B,B
005F    41           66         LD B,C
0060    42           67         LD B,D
0061    43           68         LD B,E
0062    44           69         LD B,H,NN
0063    45           70         LD B,L
0064    46           71         LD B,(HL)

LOC     OBJ CODE     STMT SOURCE STATEMENT

0065    47           72         LD B,A
0066    48           73         LD C,B
0067    49           74         LD C,C
0068    4A           75         LD C,D
0069    4B           76         LD C,E
006A    4C           77         LD C,H
006B    4D           78         LD C,L
006C    4E           79         LD C,(HL)
006D    4F           80         LD C,A
006E    50           81         LD D,B
006F    51           82         LD D,C
0070    52           83         LD D,D
0071    53           84         LD D,E
0072    54           85         LD D,H
0073    55           86         LD D,L
0074    56           87         LD D,(HL)
0075    57           88         LD D,A
0076    58           89         LD E,B
0077    59           90         LD E,C
0078    5A           91         LD E,D
0079    5B           92         LD E,E
007A    5C           93         LD E,H
007B    5D           94         LD E,L
007C    5E           95         LD E,(HL)
007D    5F           96         LD E,A
007E    60           97         LD H,B
007F    61           98         LD H,C
0080    62           99         LD H,D
0081    63          100         LD H,E
0082    64          101         LD H,H
0083    65          102         LD H,L
0084    66          103         LD H,(HL)
0085    67          104         LD H,A
0086    68          105         LD L,B
0087    69          106         LD L,C
0088    6A          107         LD L.D
0089    6B          108         LD L,E
008A    6C          109         LD L,H
008B    6D          110         LD L,L
008C    6E          111         LD L,(HL)
008D    6F          112         LD L,A
008E    70          113         LD (HL),B
008F    71          114         LD (HL),C
0090    72          115         LD (HL),D
0091    73          116         LD (HL),E
0092    74          117         LD (HL),H
0093    75          118         LD (HL),L
0094    76          119         HALT
0095    77          120         LD (HL),A
0096    78          121         LD A,B
0097    79          122         LD A,C
0098    7A          123         LD A,D
0099    7B          124         LD A,E
009A    7C          125         LD A,H
009B    7D          126         LD A,L
009C    7E          127         LD A,(HL)
009D    7F          128         LD A,A
009E    80          129         ADD A,B
009F    81          130         ADD A,C
00A0    82          131         ADD A,D
00A1    83          132         ADD A,E
00A2    84          133         ADD A,H
00A3    85          134         ADD A,L
00A4    86          135         ADD A,(HL)
00A5    87          136         ADD A,A
00A6    88          137         ADC A,B
00A7    89          138         ADC A,C
00A8    8A          139         ADC A,D
00A9    8B          140         ADC A,E
00AA    8C          141         ADC A,H
00AB    8D          142         ADC A,L
```

```
                    Z-80 CROSS ASSEMBLER    VERSION 1.06 OF 06/18/76
07/09/76  10:20:50                    OPCODE LISTING
LOC    OBJ CODE    STMT SOURCE STATEMENT
00AC   8E          143  ADC A,(HL)
00AD   8F          144  ADC A,A
00AE   90          145  SUB B
00AF   91          146  SUB C
00B0   92          147  SUB D
00B1   93          148  SUB E
00B2   94          149  SUB H
00B3   95          150  SUB L
00B4   96          151  SUB (HL)
00B5   97          152  SUB A
00B6   98          153  SBC A,B
00B7   99          154  SBC A,C
00B8   9A          155  SBC A,D
00B9   9B          156  SBC A,E
00BA   9C          157  SBC A,H
00BB   9D          158  SBC A,L
00BC   9E          159  SBC A,(HL)
00BD   9F          160  SBC A,A
00BE   A0          161  AND B
00BF   A1          162  AND C
00C0   A2          163  AND D
00C1   A3          164  AND E
00C2   A4          165  AND H
00C3   A5          166  AND L
00C4   A6          167  AND (HL)
00C5   A7          168  AND A
00C6   A8          169  XOR B
00C7   A9          170  XOR C
00C8   AA          171  XOR D
00C9   AB          172  XOR E
00CA   AC          173  XOR H
00CB   AD          174  XOR L
00CC   AE          175  XOR (HL)
00CD   AF          176  XOR A
00CE   B0          177  OR B
00CF   B1          178  OR C
00D0   B2          179  OR D
00D1   B3          180  OR E
0002   B4          181  OR H
00D3   B5          182  OR L
00D4   B6          183  OR (HL)
00D5   B7          184  OR A
00D6   B8          185  CP B
00D7   B9          186  CP C
00D8   BA          187  CP D
00D9   BB          188  CP E
00DA   BC          189  CP H
00DB   BD          190  CP L
00DC   BE          191  CP (HL)
00DD   BF          192  CP A
00DE   C0          193  RET NZ
00DF   C1          194  POP BC
00E0   C28405      195  JP NZ, NN
00E3   C38405      196  JP NN
00E6   C48405      197  CALLNZ,NN
00E9   C5          198  PUSH BC
00EA   C620        199  ADD A,N
00EC   C7          200  RST 0
00ED   C8          201  RET Z
00EE   C9          202  RET
00EF   CA8405      203  JP Z,NN
00F2   CC8405      204  CALL Z,NN
00F5   CD8405      205  CALL NN
00F8   CE20        206  ADC A,N
00FA   CF          207  RST 8
00FB   D0          208  RET NC
00FC   D1          209  POP DE
00FD   D28405      210  JP NC,NN
0100   D320        211  OUT N,A
0102   D48405      212  CALLNC,NN
0105   D5          213  PUSH DE
0106   D620        214  SUB N
0108   D7          215  RST 10H
0109   D8          216  RET C
010A   D9          217  EXX

LOC    OBJ CODE    STMT SOURCE STATEMENT
010B   DA8405      218  JP C,NN
010E   DB20        219  IN A,N
0110   DC8405      220  CALL C,NN
0113   DE20        221  SBC A,N
0115   DF          222  RST 18H
0116   E0          223  RET PO
0117   E1          224  POP HL
0118   E28405      225  JP PO,NN
011B   E3          226  EX (SP),HL
011C   E48405      227  CALL PO,NN
011F   E5          228  PUSH HL
0120   E620        229  AND N
0122   E7          230  RST 20H
0123   E8          231  RET PE
0124   E9          232  JP (HL)
0125   EA8405      233  JP PE,NN
0128   EB          234  EX DE,HL
0129   EC8405      235  CALL PE,NN
012C   EE20        236  XOR N
012E   EF          237  RST 28H
012F   F0          238  RET P
0130   F1          239  POP AF
0131   F28405      240  JP P,NN
0134   F3          241  DI
0135   F48405      242  CALL P,NN
0138   F5          243  PUSH AF
0139   F620        244  OR N
013B   F7          245  RST 30H
013C   F8          246  RET M
013D   F9          247  LD SP,HL
013E   FA8405      248  JP M,NN
0141   FB          249  EI
0142   FC8405      250  CALL M,NN
0145   FE20        251  CP N
0147   FF          252  RST 38H
0148   CB00        253  RLC B
014A   CB01        254  RLC C
014C   CB02        255  RLC D
014E   CB03        256  RLC E
0150   CB04        257  RLC H
0152   CB05        258  RLC L
0154   CB06        259  RLC (HL)
0156   CB07        260  RLC A
0158   CB08        261  RRC B
015A   CB09        262  RRC C
015C   CB0A        263  RRC D
015E   CB0B        264  RRC E
0160   CB0C        265  RRC H
0162   CB0D        266  RRC L
0164   CB0E        267  RRC (HL)
0166   CB0F        268  RRC A
0168   CB10        269  RL B
016A   CB11        270  RL C
016C   CB12        271  RL D
016E   CB13        272  RL E
0170   CB14        273  RL H
0172   CB15        274  RL L
0174   CB16        275  RL (HL)
0176   CB17        276  RL A
0178   CB18        277  RR B
017A   CB19        278  RR C
017C   CB1A        279  RR D
017E   CB1B        280  RR E
0180   CB1C        281  RR H
0182   CB1D        282  RR L
0184   CB1E        283  RR (HL)
0186   CB1F        284  RR A
0188   CB20        285  SLA B
018A   CB21        286  SLA C
018C   CB22        287  SLA D
018E   CB23        288  SLA E
0190   CB24        289  SLA H
0192   CB25        290  SLA L
0194   CB26        291  SLA (HL)
0196   CB27        292  SLA A
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
0198  CB28       293       SRA B
019A  CB29       294       SRA C
019C  CB2A       295       SRA D
019E  CB2B       296       SRA E
01A0  CB2C       297       SRA H
01A2  CB2D       298       SRA L
01A4  CB2E       299       SRA (HL)
01A6  CB2F       300       SRA A
01A8  CB38       301       SRL B
01AA  CB39       302       SRL C
01AC  CB3A       303       SRL D
01AE  CB3B       304       SRL E
01B0  CB3C       305       SRL H
01B2  CB3D       306       SRL L
01B4  CB3E       307       SRL (HL)
01B6  CB3F       308       SRL A
01B8  CB40       309       BIT 0,B
01BA  CB41       310       BIT 0,C
01BC  CB42       311       BIT 0,D
01BE  CB43       312       BIT 0,E
01C0  CB44       313       BIT 0,H
01C2  CB45       314       BIT 0,L
01C4  CB46       315       BIT 0,(HL)
01C6  CB47       316       BIT 0,A
01C8  CB48       317       BIT 1,B
01CA  CB49       318       BIT 1,C
01CC  CB4A       319       BIT 1,D
01CE  CB4B       320       BIT 1,E
01D0  CB4C       321       BIT 1,H
01D2  CB4D       322       BIT 1,L
01D4  CB4E       323       BIT 1,(HL)
01D6  CB4F       324       BIT 1,A
01D8  CB50       325       BIT 2,B
01DA  CB51       326       BIT 2,C
01DC  CB52       327       BIT 2,D
01DE  CB53       328       BIT 2,E
01E0  CB54       329       BIT 2,H
01E2  CB55       330       BIT 2,L
01E4  CB56       331       BIT 2,(HL)
01E6  CB57       332       BIT 2,A
01E8  CB58       333       BIT 3,B
01EA  CB59       334       BIT 3,C
01EC  CB5A       335       BIT 3,D
01EE  CB5B       336       BIT 3,E
01F0  CB5C       337       BIT 3,H
01F2  CB5D       338       BIT 3,L
01F4  CB5E       339       BIT 3,(HL)
01F6  CB5F       340       BIT 3,A
01F8  CB60       341       BIT 4,B
01FA  CB61       342       BIT 4,C
01FC  CB62       343       BIT 4,D
01FE  CB63       344       BIT 4,E
0200  CB64       345       BIT 4,H
0202  CB65       346       BIT 4,L
0204  CB66       347       BIT 4,(HL)
0206  CB67       348       BIT 4,A
0208  CB68       349       BIT 5,B
020A  CB69       350       BIT 5,C
020C  CB6A       351       BIT 5,D
020E  CB6B       352       BIT 5,E
0210  CB6C       353       BIT 5,H
0212  CB6D       354       BIT 5,L
0214  CB6E       355       BIT 5,(HL)
0216  CB6F       356       BIT 5,A
0218  CB70       357       BIT 6,B
021A  CB71       358       BIT 6,C
021C  CB72       359       BIT 6,D
021E  CB73       360       BIT 6,E
0220  CB74       361       BIT 6,H
0222  CB75       362       BIT 6,L
0224  CB76       363       BIT 6,(HL)
0226  CB77       364       BIT 6,A
0228  CB78       365       BIT 7,B
022A  CB79       366       BIT 7,C
022C  CB7A       367       BIT 7,D
022E  CB7B       368       BIT 7,E

LOC   OBJ CODE   STMT SOURCE STATEMENT
0230  CB7C       369       BIT 7,H
0232  CB7D       370       BIT 7,L
0234  CB7E       371       BIT 7,(HL)
0236  CB7F       372       BIT 7,A
0238  CB80       373       RES 0,B
023A  CB81       374       RES 0,C
023C  CB82       375       RES 0,D
023E  CB83       376       RES 0,E
0240  CB84       377       RES 0,H
0242  CB85       378       RES 0,L
0244  CB86       379       RES 0,(HL)
0246  CB87       380       RES 0,A
0248  CB88       381       RES 1,B
024A  CB89       382       RES 1,C
024C  CB8A       383       RES 1,D
024E  CB8B       384       RES 1,E
0250  CB8C       385       RES 1,H
0252  CB8D       386       RES 1,L
0254  CB8E       387       RES 1,(HL)
0256  CB8F       388       RES 1,A
0258  CB90       389       RES 2,B
025A  CB91       390       RES 2,C
025C  CB92       391       RES 2,D
025E  CB93       392       RES 2,E
0260  CB94       393       RES 2,H
0262  CB95       394       RES 2,L
0264  CB96       395       RES 2,(HL)
0266  CB97       396       RES 2,A
0268  CB98       397       RES 3,B
026A  CB99       398       RES 3,C
026C  CB9A       399       RES 3,D
026E  CB9B       400       RES 3,E
0270  CB9C       401       RES 3,H
0272  CB9D       402       RES 3,L
0274  CB9E       403       RES 3,(HL)
0276  CB9F       404       RES 3,A
0278  CBA0       405       RES 4,B
027A  CBA1       406       RES 4,C
027C  CBA2       407       RES 4,D
027E  CBA3       408       RES 4,E
0280  CBA4       409       RES 4,H
0282  CBA5       410       RES 4,L
0284  CBA6       411       RES 4,(HL)
0286  CBA7       412       RES 4,A
0288  CBA8       413       RES 5,B
028A  CBA9       414       RES 5,C
028C  CBAA       415       RES 5,D
028E  CBAB       416       RES 5,E
0290  CBAC       417       RES 5,H
0292  CBAD       418       RES 5,L
0294  CBAE       419       RES 5,(HL)
0296  CBAF       420       RES 5,A
0298  CBB0       421       RES 6,B
029A  CBB1       422       RES 6,C
029C  CBB2       423       RES 6,D
029E  CBB3       424       RES 6,E
02A0  CBB4       425       RES 6,H
02A2  CBB5       426       RES 6,L
02A4  CBB6       427       RES 6,(HL)
02A6  CBB7       428       RES 6,A
02A8  CBB8       429       RES 7,B
02AA  CBB9       430       RES 7,C
02AC  CBBA       431       RES 7,D
02AE  CBBB       432       RES 7,E
0280  CBBC       433       RES 7,H
0282  CBBD       434       RES 7,L
0284  CBBE       435       RES 7,(HL)
0286  CBBF       436       RES 7,A
0288  CBC0       437       SET 0,B
02BA  CBC1       438       SET 0,C
02BC  CBC2       439       SET 0,D
02BE  CBC3       440       SET 0,E
02C0  CBC4       441       SET 0,H
02C2  CBC5       442       SET 0,L
02C4  CBC6       443       SET 0,(HL)
02C6  CBC7       444       SET 0,A
```

```
LOC    OBJ CODE     STMT SOURCE STATEMENT
02C8   CBC8         445         SET 1,B
02CA   CBC9         446         SET 1,C
02CC   CBCA         447         SET 1,D
02CE   CBCB         448         SET 1,E
02D0   CBCC         449         SET 1,H
02D2   CBCD         450         SET 1,L
02D4   CBCE         451         SET 1,(HL)
02D6   CBCF         452         SET 1,A
02D8   CBD0         453         SET 2,B
02DA   CBD1         454         SET 2,C
02DC   CBD2         455         SET 2,D
02DE   CBD3         456         SET 2,E
02E0   CBD4         457         SET 2,H
02E2   CBD5         458         SET 2,L
02E4   CBD6         459         SET 2,(HL)
02E6   CBD7         460         SET 2,A
02E8   CBD8         461         SET 3,B
02EA   CBD9         462         SET 3,C
02EC   CBDA         463         SET 3,D
02EE   CBDB         464         SET 3,E
02F0   CBDC         465         SET 3,H
02F2   CBDD         466         SET 3,L
02F4   CBDE         467         SET 3,(HL)
02F6   CBDF         468         SET 3,A
02F8   CBE0         469         SET 4,B
02FA   CBE1         470         SET 4,C
02FC   CBE2         471         SET 4,D
02FE   CBE3         472         SET 4,E
0300   CBE4         473         SET 4,H
0302   CBE5         474         SET 4,L
0304   CBE6         475         SET 4,(HL)
0306   CBE7         476         SET 4,A
0308   CBE8         477         SET 5,B
030A   CBE9         478         SET 5,C
030C   CBEA         479         SET 5,D
030E   CBEB         480         SET 5,E
0310   CBEC         481         SET 5,H
0312   CBED         482         SET 5,L
0314   CBEE         483         SET 5,(HL)
0316   CBEF         484         SET 5,A
0318   CBF0         485         SET 6,B
031A   CBF1         486         SET 6,C
031C   CBF2         487         SET 6,D
031E   CBF3         488         SET 6,E
0320   CBF4         489         SET 6,H
0322   CBF5         490         SET 6,L
0324   CBF6         491         SET 6,(HL)
0326   CBF7         492         SET 6,A
0328   CBF8         493         SET 7,B
032A   CBF9         494         SET 7,C
032C   CBFA         495         SET 7,D
032E   CBFB         496         SET 7,E
0330   CBFC         497         SET 7,H
0332   CBFD         498         SET 7,L
0334   CBFE         499         SET 7,(HL)
0336   CBFF         500         SET 7,A
0338   DD09         501         ADD IX,BC
033A   DD19         502         ADD IX,DE
033C   DD218405     503         LD IX,NN
0340   DD228405     504         LD (NN),IX
0344   DD23         505         INC IX
0346   DD29         506         ADD IX,IX
0348   DD2A8405     507         LD IX,(NN)
034C   DD2B         508         DEC IX
034E   DD3405       509         INC (IX+IND)
0351   DD3505       510         DEC (IX+IND)
0354   DD360520     511         LD (IX+IND),N
0358   DD39         512         ADD IX,SP
035A   DD4605       513         LD B,(IX+IND)
035D   DD4E05       514         LD C,(IX+IND)
0360   DD5605       515         LD D,(IX+IND)
0363   DD5E05       516         LD E,(IX+IND)
0366   DD6605       517         LD H,(IX+IND)
0369   DD6E05       518         LD L,(IX+IND)
036C   DD7005       519         LD (IX+IND),B

LOC    OBJ CODE     STMT SOURCE STATEMENT
036F   DD7105       520         LD (IX+IND),C
0372   DD7205       521         LD (IX+IND),D
0375   DD7305       522         LD (IX+IND),E
0378   DD7405       523         LD (IX+IND),H
037B   DD7505       524         LD (IX+IND),L
037E   DD7705       525         LD (IX+IND),A
0381   DD7E05       526         LD A,(IX+IND)
0384   DD8605       527         ADD A,(IX+IND)
0387   DD8E05       528         ADC A,(IX+IND)
038A   DD9605       529         SUB (IX+IND)
038D   DD9E05       530         SBC A,(IX+IND)
0390   DDA605       531         AND (IX+IND)
0393   DDAE05       532         XOR (IX+IND)
0396   DDB605       533         OR (IX+IND)
0399   DDBE05       534         CP (IX+IND)
039C   DDE1         535         POP IX
039E   DDE3         536         EX (SP),IX
03A0   DDE5         537         PUSH IX
03A2   DDE9         538         JP (IX)
03A4   DDF9         539         LD SP,IX
03A6   DDCB0506     540         RLC (IX+IND)
03AA   DDCB050E     541         RRC (IX+IND)
03AE   DDCB0516     542         RL (IX+IND)
03B2   DDCB051E     543         RR (IX+IND)
03B6   DDCB0526     544         SLA (IX+IND)
03BA   DDCB052E     545         SRA (IX+IND)
03BE   DDCB053E     546         SRL (IX+IND)
03C2   DDCB0546     547         BIT 0,(IX+IND)
03C6   DDCB054E     548         BIT 1,(IX+IND)
03CA   DDCB0556     549         BIT 2,(IX+IND)
03CE   DDCB055E     550         BIT 3,(IX+IND)
03D2   DDCB0566     551         BIT 4,(IX+IND)
03D6   DDCB056E     552         BIT 5,(IX+IND)
03DA   DDCB0576     553         BIT 6,(IX+IND)
03DE   DDCB057E     554         BIT 7,(IX+IND)
03E2   DDCB0586     555         RES 0,(IX+IND)
03E6   DDCB058E     556         RES 1,(IX+IND)
03EA   DDCB0596     557         RES 2,(IX+IND)
03EE   DDCB059E     558         RES 3,(IX+IND)
03F2   DDCB05A6     559         RES 4,(IX+IND)
03F6   DDCB05AE     560         RES 5,(IX+IND)
03FA   DDCB05B6     561         RES 6,(IX+IND)
03FE   DDCB05BE     562         RES 7,(IX+IND)
0402   DDCB05C6     563         SET 0,(IX+IND)
0406   DDCB05CE     564         SET 1,(IX+IND)
040A   DDCB05D6     565         SET 2,(IX+IND)
040E   DDCB05DE     566         SET 3,(IX+IND)
0412   DDCB05E6     567         SET 4,(IX+IND)
0416   DDCB05EE     568         SET 5,(IX+IND)
041A   DDCB05F6     569         SET 6,(IX+IND)
041E   DDCB05FE     570         SET 7,(IX+IND)
0422   ED40         571         IN B,(C)
0424   ED41         572         OUT (C),B
0426   ED42         573         SBC HL,BC
0428   ED438405     574         LD (NN),BC
042C   ED44         575         NEG
042E   ED45         576         RETN
0430   ED46         577         IM 0
0432   ED47         578         LD I,A
0434   ED48         579         IN C,(C)
0436   ED49         580         OUT (C),C
0438   ED4A         581         ADC HL,BC
043A   ED4B8405     582         LD BC,(NN)
043E   ED4D         583         RETI
0440   ED50         584         IN D,(C)
0442   ED51         585         OUT (C),D
0444   ED52         586         SBC HL,DE
0446   ED538405     587         LD (NN),DE
044A   ED56         588         IM I
044C   ED57         589         LD A,I
044E   ED58         590         IN E,(C)
0450   ED59         591         OUT (C),E
0452   ED5A         592         ADC HL,DE
0454   ED5B8405     593         LD DE,(NN)
0458   ED5E         594         IM 2
```

```
                          Z-80 CROSS ASSEMBLER   VERSION 1.06 OF 06/18/76
07/09/76  10:20:50                    OPCODE LISTING
LOC      OBJ CODE      STMT SOURCE STATEMENT
045A     ED60          595          IN H,(C)
045C     ED61          596          OUT (C),H
045E     ED62          597          SBC HL,HI
0460     ED67          598          RRD
0462     ED68          599          IN L,(C)
0464     ED69          600          OUT (C),L
0466     ED6A          601          ADC HL,HL
0468     ED6F          602          RLD
046A     ED72          603          SBC HL,SP
046C     ED738405      604          LD (NN),SP
0470     ED78          605          IN A,(C)
0472     ED79          606          OUT (C),A
0474     ED7A          607          ADC HL,SP
0476     ED7B8405      608          LD SP, (NN)
047A     EDA0          609          LDI
047C     EDA1          610          CPI
047E     EDA2          611          INI
0480     EDA3          612          OUTI
0482     EDA8          613          LDD
0484     EDA9          614          CPD
0486     EDAA          615          IND
0488     EDAB          616          OUTD
048A     EDB0          617          LDIR
048C     EDB1          618          CPIR
048E     EDB2          619          INIR
0490     EDB3          620          OTIR
0492     EDB8          621          LDDR
0494     EDB9          622          CPDR
0496     EDBA          623          INDR
0498     EDBB          624          OTDR
049A     FD09          625          ADD IY,BC
049C     FD19          626          ADD IY,DE
049E     FD218405      627          LD IY,NN
04A2     FD228405      628          LD (NN),IY
04A6     FD23          629          INC IY
04A8     FD29          630          ADD IY,IY
04AA     FD2A8405      631          LD IY,(NN)
04AE     FD2B          632          DEC IY
04B0     FD3405        633          INC (IY+IND)
04B3     FD3505        634          DEC (IY+IND)
04B6     FD360520      635          LD (IY+IND),N
04BA     FD39          636          ADD IY,SP
04BC     FD4605        637          LD B,(IY+IND)
04BF     FD4E05        638          LD C,(IY+IND)
04C2     FD5605        639          LD D,(IY+IND)
04C5     FD5E05        640          LD E,(IY+IND)
04C8     FD6605        641          LD H,(IY+IND)
04CB     FD6E05        642          LD L,(IY+IND)
04CE     FD7005        643          LD (IY+IND),B
04D1     FD7105        644          LD (IY+IND),C
04D4     FD7205        645          LD(IY+IND),D
04D7     FD7305        646          LD (IY+IND),E
04DA     FD7405        647          LD (IY+IND),H
04DD     FD7505        648          LD (IY+IND),L
04E0     FD7705        649          LD (IY+IND),A
04E3     FD7E05        650          LD A,(IY+IND)
04E6     FD8605        651          ADD A,(IY+IND)
04E9     FD8E05        652          ADC A,(IY+IND)
04EC     FD9605        653          SUB (IY+IND)
04EF     FD9E05        654          SBC A,(IY+IND)
04F2     FDA605        655          AND (IY+IND)
04F5     FDAE05        656          XOR (IY+IND)
04F8     FDB605        657          OR (IY+IND)
04FB     FDBE05        658          CP (IY+IND)
04FE     FDE1          659          POP IY
0500     FDE3          660          EX (SP),IY
0502     FDE5          661          PUSH IY
0504     FDE9          662          JP (IY)
0506     FDF9          663          LD SP,IY
0508     FDCB0506      664          RLC (IY+IND)
050C     FDCB050E      665          RRC (IY+IND)
0510     FDCB0516      666          RL (IY+IND)
0514     FDCB051E      667          RR (IY+IND)
0518     FDCB0526      668          SLA (IY+IND)
051C     FDCB052E      669          SRA (IY+IND)

LOC      OBJ CODE      STMT SOURCE STATEMENT
0520     FDCB053E      670          SRL (IY+IND)
0524     FDCB0546      671          BIT 0,(IY+IND)
0528     FDCB054E      672          BIT 1,(IY+IND)
052C     FDCB0556      673          BIT 2,(IY+IND)
0530     FDCB055E      674          BIT 3,(IY+IND)
0534     FDCB0566      675          BIT 4,(IY+IND)
0538     FDCB056E      676          BIT 5,(IY+IND)
053C     FDCB0576      677          BIT 6,(IY+IND)
0540     FDCB057E      678          BIT 7,(IY+IND)
0544     FDCB0586      679          RES 0,(IY+IND)
0548     FDCB058E      680          RES 1,(IY+IND)
054C     FDCB0596      681          RES 2,(IY+IND)
0550     FDCB059E      682          RES 3,(IY+IND)
0554     FDCB05A6      683          RES 4,(IY+IND)
0558     FDCB05AE      684          RES 5,(IY+IND)
055C     FDCB05B6      685          RES 6,(IY+IND)
0560     FDCB05BE      686          RES 7,(IY+IND)
0564     FDCB05C6      687          SET 0,(IY+IND)
0568     FDCB05CE      688          SET 1,(IY+IND)
056C     FDCB05D6      689          SET 2,(IY+IND)
0570     FDCB05DE      690          SET 3,(IY+IND)
0574     FDCB05E6      691          SET 4,(IY+IND)
0578     FDCB05EE      692          SET 5,(IY+IND)
057C     FDCB05F6      693          SET 6,(IY+IND)
0580     FDCB05FE      694          SET 7,(IY+IND)
0584                   695 NN       DEFS 2
                       696 IND      EQU 5
                       697 M        EQU 10H
                       698 N        EQU 20H
                       699 DIS      EQU 30H
                       700          END
```



Z80-CPU REGISTER CONFIGURATION

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
0123		4567		0123		4567		0123		4567	
BYTE				BYTE				BYTE			

ASCII CHARACTER SET (7-BIT CODE)

LSD \ MSD		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENG	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	
C	1100	FF	FS	'	<	L		l	|
D	1101	CR	GS	–	=	M	]	m	
E	1110	SO	RS	●	>	N	↑	n	~
F	1111	SI	VS	/	?	_	←	o	DEL

POWERS OF 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$

POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

ALPHABETIC LIST OF INSTRUCTION SET

```
                              Z-80 CROSS ASSEMBLER    VERSION 1.06 OF 06/18/76
07/09/76  10:22:47                          OPCODE LISTING
LOC     OBJ CODE      STMT SOURCE STATEMENT
0000    8E               1        ADC     A, (HL)
0001    DD8E05           2        ADC     A, (IX+IND)
0004    FD8E05           3        ADC     A, (IY+IND)
0007    8F               4        ADC     A, A
0008    88               5        ADC     A, B
0009    89               6        ADC     A, C
000A    8A               7        ADC     A, D
000B    8B               8        ADC     A, E
000C    8C               9        ADC     A, H
000D    8D              10        ADC     A, L
000E    CE20            11        ADC     A, N
0010    ED4A            12        ADC     HL, BC
0012    ED5A            13        ADC     HL, DE
0014    ED6A            14        ADC     HL, HL
0016    ED7A            15        ADC     HL, SP
0018    86              16        ADD     A, (HL)
0019    DD8605          17        ADD     A, (IX+IND)
001C    FD8605          18        ADD     A, (IY+IND)
001F    87              19        ADD     A, A
0020    80              20        ADD     A, B
0021    81              21        ADD     A, C
0022    82              22        ADD     A, D
0023    83              23        ADD     A, E
0024    84              24        ADD     A, H
0025    85              25        ADD     A, L
0026    C620            26        ADD     A, N
0028    09              27        ADD     HL, BC
0029    19              28        ADD     HL, DE
002A    29              29        ADD     HL, HL
002B    39              30        ADD     HL, SP
002C    DD09            31        ADD     IX, BC
002E    DD19            32        ADD     IX, DE
0030    DD29            33        ADD     IX, IX
0032    DD39            34        ADD     IX, SP
0034    FD09            35        ADD     IY, BC
0036    FD19            36        ADD     IY, DE
0038    FD29            37        ADD     IY, IY
003A    FD39            38        ADD     IY, SP
003C    A6              39        AND     (HL)
003D    DDA605          40        AND     (IX+IND)
0040    FDA605          41        AND     (IY+IND)
0043    A7              42        AND     A
0044    A0              43        AND     B
0045    A1              44        AND     C
0046    A2              45        AND     D
0047    A3              46        AND     E
0048    A4              47        AND     H
0049    A5              48        AND     L
004A    E620            49        AND     N
004C    CB46            50        BIT     O, (HL)
004E    DDCB0546        51        BIT     O, (IX+IND)
0052    FDBC0546        52        BIT     O, (IY+IND)
0056    CB47            53        BIT     O, A
0058    CB40            54        BIT     O, B
005A    CB41            55        BIT     O, C
005C    CB42            56        BIT     O, D
005E    CB43            57        BIT     O, E
0060    CB44            58        BIT     O, H
0062    CB45            59        BIT     O, L
0064    CB4E            60        BIT     1, (HL)
0066    DDCB054E        61        BIT     1, (IX+IND)
006A    FDCB054E        62        BIT     1, (IY+IND)
006E    CB4F            63        BIT     1, A
0070    CB48            64        BIT     1, B
0072    CB49            65        BIT     1, C
0074    CB4A            66        BIT     1, D
0076    CB4B            67        BIT     1, E
0078    CB4C            68        BIT     1, H
007A    CB4D            69        BIT     1, L
007C    CB56            70        BIT     2, (HL)
007E    DDCB0556        71        BIT     2, (IX+IND)
0082    FDCB0556        72        BIT     2, (IY+IND)
0086    CB57            73        BIT     2, A
```

```
LOC     OBJ CODE      STMT SOURCE STATEMENT
0088    CB50            74        BIT     2, B
008A    CB51            75        BIT     2, C
008C    CB52            76        BIT     2, D
008E    CB53            77        BIT     2, E
0090    CB54            78        BIT     2, H
0092    CB55            79        BIT     2, L
0094    CB5E            80        BIT     3, (HL)
0096    DDCB055E        81        BIT     3, (IX+IND)
009A    FDCB055E        82        BIT     3, (IY+IND)
009E    CB5F            83        BIT     3, A
00A0    CB58            84        BIT     3, B
00A2    CB59            85        BIT     3, C
00A4    CB5A            86        BIT     3, D
00A6    CB5B            87        BIT     3, E
00A8    CB5C            88        BIT     3, H
00AA    CB5D            89        BIT     3, L
00AC    CB66            90        BIT     4, (HL)
00AE    DDCB0566        91        BIT     4, (IX+IND)
00B2    FDCB0566        92        BIT     4, (IY+IND)
00B6    CB67            93        BIT     4, A
00B8    CB60            94        BIT     4, B
00BA    CB61            95        BIT     4, C
00BC    CB62            96        BIT     4, D
00BE    CB63            97        BIT     4, E
00C0    CB64            98        BIT     4, H
00C2    CB65            99        BIT     4, L
00C4    CB6E           100        BIT     5, (HL)
00C6    DDCB056E       101        BIT     5, (IX+IND)
00CA    FDCB056E       102        BIT     5, (IY+IND)
00CE    CB6F           103        BIT     5, A
00D0    CB68           104        BIT     5, B
00D2    CB69           105        BIT     5, C
00D4    CB6A           106        BIT     5, D
00D6    CB6B           107        BIT     5, E
00D8    CB6C           108        BIT     5, H
00DA    CB6D           109        BIT     5, L
00DC    CB76           110        BIT     6, (HL)
00DE    DDCB0576       111        BIT     6, (IX+IND)
00E2    FDCB0576       112        BIT     6, (IY+IND)
00E6    CB77           113        BIT     6, A
00E8    CB70           114        BIT     6, B
00EA    CB71           115        BIT     6, C
00EC    CB72           116        BIT     6, D
00EE    CB73           117        BIT     6, E
00F0    CB74           118        BIT     6, H
00F2    CB75           119        BIT     6, L
00F4    CB7E           120        BIT     7, (HL)
00F6    DDCB057E       121        BIT     7, (IX+IND)
00FA    FDCB057E       122        BIT     7, (IY+IND)
00FE    CB7F           123        BIT     7, A
0100    CB78           124        BIT     7, B
0102    CB79           125        BIT     7, C
0104    CB7A           126        BIT     7, D
0106    CB7B           127        BIT     7, E
0108    CB7C           128        BIT     7, H
010A    CB7D           129        BIT     7, L
010C    DC8405         130        CALL    C, NN
010F    FC8405         131        CALL    M, NN
0112    D48405         132        CALL    NC, NN
0115    CD8405         133        CALL    NN
0118    C48405         134        CALL    NZ, NN
011B    F48405         135        CALL    P, NN
011E    EC8405         136        CALL    PE, NN
0121    E48405         137        CALL    PO, NN
0124    CC8405         138        CALL    Z, NN
0127    3F             139        CCF
0128    BE             140        CP      (HL)
0129    DDBE05         141        CP      (IX+IND)
012C    FDBE05         142        CP      (IY+IND)
012F    BF             143        CP      A
0130    B8             144        CP      B
0131    B9             145        CP      C
0132    BA             146        CP      D
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
0133  BB         147       CP     E
0134  BC         148       CP     H
0135  BD         149       CP     L
0136  FE20       150       CP     N
0138  EDA9       151       CPD
013A  EDB9       152       CPDR
013C  EDA1       153       CPI
013E  EDB1       154       CPIR
0140  2F         155       CPL
0141  27         156       DAA
0142  35         157       DEC    (HL)
0143  DD3505     158       DEC    (IX+IND)
0146  FD3505     159       DEC    (IY+IND)
0149  3D         160       DEC    A
014A  05         161       DEC    B
014B  0B         162       DEC    BC
014C  0D         163       DEC    C
014D  15         164       DEC    D
014E  1B         165       DEC    DE
014F  1D         166       DEC    E
0150  25         167       DEC    H
0151  2B         168       DEC    HL
0152  DD2B       169       DEC    IX
0154  FD2B       170       DEC    IY
0156  2D         171       DEC    L
0157  3B         172       DEC    SP
0158  F3         173       DI
0159  102E       174       DJNZ   DIS
015B  FB         175       EI
015C  E3         176       EX     (SP), HL
015D  DDE3       177       EX     (SP), IX
015F  FDE3       178       EX     (SP), IY
0161  08         179       EX     AF, AF'
0162  EB         180       EX     DE, HL
0163  D9         181       EXX
0164  76         182       HALT
0165  ED46       183       IM     0
0167  ED56       184       IM     1
0169  ED5E       185       IM     2
016B  ED78       186       IN     A, (C)
016D  DB20       187       IN     A, N
016F  ED40       188       IN     B, (C)
0171  ED48       189       IN     C, (C)
0173  ED50       190       IN     D, (C)
0175  ED58       191       IN     E, (C)
0177  ED60       192       IN     H, (C)
0179  ED68       193       IN     L, (C)
017B  34         194       INC    (HL)
017C  DD3405     195       INC    (IX+IND)
017F  FD3405     196       INC    (IY+IND)
0182  3C         197       INC    A
0183  04         198       INC    B
0184  03         199       INC    BC
0185  0C         200       INC    C
0186  14         201       INC    D
0187  13         202       INC    DE
0188  1C         203       INC    E
0189  24         204       INC    H
018A  23         205       INC    HL
018B  DD23       206       INC    IX
018D  FD23       207       INC    IY
018F  2C         208       INC    L
0190  33         209       INC    SP
0191  EDAA       210       IND
0193  EDBA       211       INDR
0195  EDA2       212       INI
0197  EDB2       213       INIR
0199  E9         214       JP     (HL)
019A  DDE9       215       JP     (IX)
019C  FDE9       216       JP     (IY)
019E  DA8405     217       JP     C, NN
01A1  FA8405     218       JP     M, NN
01A4  D28405     219       JP     NC, NN
01A7  C38405     220       JP     NN
01AA  C28405     221       JP     NZ, NN

LOC   OBJ CODE   STMT SOURCE STATEMENT
01AD  F28405     222       JP     P, NN
01B0  EA8405     223       JP     PE, NN
01B3  E28405     224       JP     PO, NN
01B6  CA8405     225       JP     Z, NN
01B9  382E       226       JR     C, DIS
01BB  182E       227       JR     DIS
01BD  302E       228       JR     NC, DIS
01BF  202E       229       JR     NZ, DIS
01C1  282E       230       JR     Z, DIS
01C3  02         231       LD     (BC), A
01C4  12         232       LD     (DE), A
01C5  77         233       LD     (HL), A
01C6  70         234       LD     (HL), B
01C7  71         235       LD     (HL), C
01C8  72         236       LD     (HL), D
01C9  73         237       LD     (HL), E
01CA  74         238       LD     (HL), H
01CB  75         239       LD     (HL), L
01CC  3620       240       LD     (HL), N
01CE  DD7705     241       LD     (IX+IND), A
01D1  DD7005     242       LD     (IX+IND), B
01D4  DD7105     243       LD     (IX+IND), C
01D7  DD7205     244       LD     (IX+IND), D
01DA  DD7305     245       LD     (IX+IND), E
01DD  DD7405     246       LD     (IX+IND), H
01E0  DD7505     247       LD     (IX+IND), L
01E3  DD360520   248       LD     (IX+IND), N
01E7  FD7705     249       LD     (IY+IND), A
01EA  FD7005     250       LD     (IY+IND), B
01ED  FD7105     251       LD     (IY+IND), C
01F0  FD7205     252       LD     (IY+IND), D
01F3  FD7305     253       LD     (IY+IND), E
01F6  FD7405     254       LD     (IY+IND), H
01F9  FD7505     255       LD     (IY+IND), L
01FC  FD360520   256       LD     (IY+IND), N
0200  328405     257       LD     (NN), A
0203  ED438405   258       LD     (NN), BC
0207  ED538405   259       LD     (NN), DE
020B  228405     260       LD     (NN), HL
020E  DD228405   261       LD     (NN), IX
0212  FD228405   262       LD     (NN), IY
0216  ED738405   263       LD     (NN), SP
021A  0A         264       LD     A, (BC)
021B  1A         265       LD     A, (DE)
021C  7E         266       LD     A, (HL)
021D  DD7E05     267       LD     A, (IX+IND)
0220  FD7E05     268       LD     A, (IY+IND)
0223  3A8405     269       LD     A, (NN)
0226  7F         270       LD     A, A
0227  78         271       LD     A, B
0228  79         272       LD     A, C
0229  7A         273       LD     A, D
022A  7B         274       LD     A, E
022B  7C         275       LD     A, H
022C  ED57       276       LD     A, I
022E  7D         277       LD     A, L
022F  3E20       278       LD     A, N
0231  46         279       LD     B, (HL)
0232  DD4605     280       LD     B, (IX+IND)
0235  FD4605     281       LD     B, (IY+IND)
0238  47         282       LD     B, A
0239  40         283       LD     B, B
023A  41         284       LD     B, C
023B  42         285       LD     B, D
023C  43         286       LD     B, E
023D  44         287       LD     B, H, NN
023E  45         288       LD     B. L
023F  0620       289       LD     B, N
0241  ED4B8405   290       LD     BC, (NN)
0245  018405     291       LD     BC, NN
0248  4E         292       LD     C, (HL)
0249  DD4E05     293       LD     C, (IX+IND)
024C  FD4E05     294       LD     C, (IY+IND)
024F  4F         295       LD     C, A
0250  48         296       LD     C, B
```

```
LOC    OBJ CODE    STMT SOURCE STATEMENT
0251   49          297    LD     C, C
0252   4A          298    LD     C, D
0253   4B          299    LD     C, E
0254   4C          300    LD     C, H
0255   4D          301    LD     C, L
0256   0E20        302    LD     C, N
0258   56          303    LD     D, (HL)
0259   DD5605      304    LD     D, (IX+IND)
025C   FD5605      305    LD     D, (IY+IND)
025F   57          306    LD     D, A
0260   50          307    LD     D, B
0261   51          308    LD     D, C
0262   52          309    LD     D, D
0263   53          310    LD     D, E
0264   54          311    LD     D, H
0265   55          312    LD     D, L
0266   1620        313    LD     D, N
0268   ED5B8405    314    LD     DE, (NN)
026C   118405      315    LD     DE, NN
026F   5E          316    LD     E, (HL)
0270   DD5E05      317    LD     E, (IX+IND)
0273   FD5E05      318    LD     E, (IY+IND)
0276   5F          319    LD     E, A
0277   58          320    LD     E, B
0278   59          321    LD     E, C
0279   5A          322    LD     E, D
027A   5B          323    LD     E, E
027B   5C          324    LD     E, H
027C   5D          325    LD     E, L
027D   1E20        326    LD     E, N
027F   66          327    LD     H, (HL)
0280   DD6605      328    LD     H, (IX+IND)
0283   FD6605      329    LD     H, (IY+IND)
0286   67          330    LD     H, A
0287   60          331    LD     H, B
0288   61          332    LD     H, C
0289   62          333    LD     H, D
028A   63          334    LD     H, E
028B   64          335    LD     H, H
028C   65          336    LD     H, L
028D   2620        337    LD     H, N
028F   2A8405      338    LD     HL, (NN)
0292   218405      339    LD     HL, NN
0295   ED47        340    LD     I, A
0297   DD2A8405    341    LD     IX, (NN)
029B   DD218405    342    LD     IX, NN
029F   FD2A8405    343    LD     IY, (NN)
02A3   FD218405    344    LD     IY, NN
02A7   6E          345    LD     L, (HL)
02A8   DD6E05      346    LD     L,(IX+IND)
02AB   FD6E05      347    LD     L,(IY+IND)
02AE   6F          348    LD     L,A
02AF   68          349    LD     L,B
02B0   69          350    LD     L,C
02B1   6A          351    LD     L,D
02B2   6B          352    LD     L,E
02B3   6C          353    LD     L,H
02B4   6D          354    LD     L,L
02B5   2E20        355    LD     L,N
02B7   ED7B8405    356    LD     SP,(NN)
02BB   F9          357    LD     SP,HL
02BC   DDF9        358    LD     SP,IX
02BE   FDF9        359    LD     SP,IY
02C0   318405      360    LD     SP,NN
02C3   EDA8        361    LDD
02C5   EDB8        362    LDDR
02C7   EDA0        363    LDI
02C9   EDB0        364    LDIR
02CB   ED44        365    NEG
02CD   00          366    NOP
02CE   B6          367    OR     (HL)
02CF   DDB605      368    OR     (IX+IND)
02D2   FDB605      369    OR     (IY+IND)
02D5   B7          370    OR     A
02D6   B0          371    OR     B
02D7   B1          372    OR     C
```

```
LOC    OBJ CODE    STMT SOURCE STATEMENT
02D8   B2          373    OR     D
02D9   B3          374    OR     E
02DA   B4          375    OR     H
02DB   B5          376    OR     L
02DC   F620        377    OR     N
02DE   EDBB        378    OTDR
02E0   EDB3        379    OTIR
02E2   ED79        380    OUT    (C),A
02E4   ED41        381    OUT    (C),B
02E6   ED49        382    OUT    (C),C
02E8   ED51        383    OUT    (C),D
02EA   ED59        384    OUT    (C),E
02EC   ED61        385    OUT    (C),H
02EE   ED69        386    OUT    (C),L
02F0   D320        387    OUT    N,A
02F2   EDAB        388    OUTD
02F4   EDA3        389    OUTI
02F6   F1          390    POP    AF
02F7   C1          391    POP    BC
02F8   D1          392    POP    DE
02F9   E1          393    POP    HL
02FA   DDE1        394    POP    IX
02FC   FDE1        395    POP    IY
02FE   F5          396    PUSH   AF
02FF   C5          397    PUSH   BC
0300   D5          398    PUSH   DE
0301   E5          399    PUSH   HL
0302   DDE5        400    PUSH   IX
0304   FDE5        401    PUSH   IY
0306   CB86        402    RES    0,(HL)
0308   DDCB0586    403    RES    0,(IX+IND)
030C   FDCB0586    404    RES    0,(IY+IND)
0310   CB87        405    RES    0,A
0312   CB80        406    RES    0,B
0314   CB81        407    RES    0,C
0316   CB82        408    RES    0,D
0318   CB83        409    RES    0,E
031A   CB84        410    RES    0,H
031C   CB85        411    RES    0,L
031E   CB8E        412    RES    1,(HL)
0320   DDCB058E    413    RES    1,(IX+IND)
0324   FDCB058E    414    RES    1,(IY+IND)
0328   CB8F        415    RES    1,A
032A   CB88        416    RES    1,B
032C   CB89        417    RES    1,C
032E   CB8A        418    RES    1,D
0330   CB8B        419    RES    1,E
0332   CB8C        420    RES    1,H
0334   CB8D        421    RES    1,L
0336   CB96        422    RES    2,(HL)
0338   DDCB0596    423    RES    2,(IX+IND)
033C   FDCB0596    424    RES    2,(IY+IND)
0340   CB97        425    RES    2,A
0342   CB90        426    RES    2,B
0344   CB91        427    RES    2,C
0346   CB92        428    RES    2,D
0348   CB93        429    RES    2,E
034A   CB94        430    RES    2,H
034C   CB95        431    RES    2,L
034E   CB9E        432    RES    3,(HL)
0350   DDCB059E    433    RES    3,(IX+IND)
0354   FDCB059E    434    RES    3,(IY+IND)
0358   CB9F        435    RES    3,A
035A   CB98        436    RES    3,B
035C   CB99        437    RES    3,C
035E   CB9A        438    RES    3,D
0360   CB9B        439    RES    3,E
0362   CB9C        440    RES    3,H
0364   CB9D        441    RES    3,L
0366   CBA6        442    RES    4,(HL)
0368   DDCB05A6    443    RES    4,(IX+IND)
036C   FDCB05A6    444    RES    4,(IY+IND)
0370   CBA7        445    RES    4,A
0372   CBA0        446    RES    4,B
0374   CBA1        447    RES    4,C
0376   CBA2        448    RES    4,D
```

```
LOC    OBJ CODE     STMT SOURCE STATEMENT
0378   CBA3         449        RES     4,E
037A   CBA4         450        RES     4,H
037C   CBA5         451        RES     4,L
037E   CBAE         452        RES     5,(HL)
0380   DDCB05AE     453        RES     5,(IX+IND)
0384   FDCB05AE     454        RES     5,(IY+IND)
0388   CBAF         455        RES     5,A
038A   CBA8         456        RES     5,B
038C   CBA9         457        RES     5,C
038E   CBAA         458        RES     5,D
0390   CBAB         459        RES     5,E
0392   CBAC         460        RES     5,H
0394   CBAD         461        RES     5,L
0396   CBB6         462        RES     6,(HL)
0398   DDCB05B6     463        RES     6,(IX+IND)
039C   FDCB05B6     464        RES     6,(IY+IND)
03A0   CBB7         465        RES     6,A
03A2   CBB0         466        RES     6,B
03A4   CBB1         467        RES     6,C
03A6   CBB2         468        RES     6,D
03A8   CBB3         469        RES     6,E
03AA   CBB4         470        RES     6,H
03AC   CBB5         471        RES     6,L
03AE   CBBE         472        RES     7,(HL)
03B0   DDCB05BE     473        RES     7,(IX+IND)
03B4   FDCB05BE     474        RES     7,(IY+IND)
03B8   CBBF         475        RES     7,A
03BA   CBB8         476        RES     7,B
03BC   CBB9         477        RES     7,C
03BE   CBBA         478        RES     7,D
03C0   CBBB         479        RES     7,E
03C2   CBBC         480        RES     7,H
03C4   CBBD         481        RES     7,L
03C6   C9           482        RET
03C7   D8           483        RET     C
03C8   F8           484        RET     M
03C9   D0           485        RET     NC
03CA   C0           486        RET     NZ
03CB   F0           487        RET     P
03CC   E8           488        RET     PE
03CD   E0           489        RET     PO
03CE   C8           490        RET     Z
03CF   ED4D         491        RETI
03D1   ED45         492        RETN
03D3   CB16         493        RL      (HL)
03D5   DDCB0516     494        RL      (IX+IND)
03D9   FDCB0516     495        RL      (IY+IND)
03DD   CB17         496        RL      A
03DF   CB10         497        RL      B
03E1   CB11         498        RL      C
03E3   CB12         499        RL      D
03E5   C813         500        RL      E
03E7   CB14         501        RL      H
03E9   CB15         502        RL      L
03EB   17           503        RLA
03EC   CB06         504        RLC     (HL)
03EE   DDCB0506     505        RLC     (IX+IND)
03F2   FDCB0506     506        RLC     (IY+IND)
03F6   CB07         507        RLC     A
03F8   CB00         508        RLC     B
03FA   CB01         509        RLC     C
03FC   CB02         510        RLC     D
03FE   CB03         511        RLC     E
0400   CB04         512        RLC     H
0402   CB05         513        RLC     L
0404   07           514        RLCA
0405   ED6F         515        RLD
0407   CB1E         516        RR      (HL)
0409   DDCB051E     517        RR      (IX+IND)
040D   FDCB051E     518        RR      (IY+IND)
0411   CB1F         519        RR      A
0413   CB18         520        RR      B
0415   CB19         521        RR      C
0417   CB1A         522        RR      D
0419   CB1B         523        RR      E

LOC    OBJ CODE     STMT SOURCE STATEMENT
041B   CB1C         524        RR      H
041D   CB1D         525        RR      L
041F   1F           526        RRA
0420   CB0E         527        RRC     (HL)
0422   DDCB050E     528        RRC     (IX+IND)
0426   FDCB050E     529        RRC     (IY+IND)
042A   CB0F         530        RRC     A
042C   CB08         531        RRC     B
042E   CB09         532        RRC     C
0430   CB0A         533        RRC     D
0432   CB0B         534        RRC     E
0434   CB0C         535        RRC     H
0436   CB0D         536        RRC     L
0438   0F           537        RRCA
0439   ED67         538        RRD
043B   C7           539        RST     0
043C   D7           540        RST     10H
043D   DF           541        RST     18H
043E   E7           542        RST     20H
043F   EF           543        RST     28H
0440   F7           544        RST     30H
0441   FF           545        RST     38H
0442   CF           546        RST     8
0443   9E           547        SBC     A,(HL)
0444   DD9E05       548        SBC     A,(IX+IND)
0447   FD9E05       549        SBC     A,(IY+IND)
044A   9F           550        SBC     A,A
044B   98           551        SBC     A,B
044C   99           552        SBC     A,C
044D   9A           553        SBC     A,D
044E   9B           554        SBC     A,E
044F   9C           555        SBC     A,H
0450   9D           556        SBC     A,L
0451   DE20         557        SBC     A,N
0453   ED42         558        SBC     HL,BC
0455   ED52         559        SBC     HL,DE
0457   ED62         560        SBC     HL,HL
0459   ED72         561        SBC     HL,SP
045B   37           562        SCF
045C   CBC6         563        SET     0,(HL)
045E   DDCB05C6     564        SET     0,(IX+IND)
0462   FDCB05C6     565        SET     0,(IY+IND)
0466   CBC7         566        SET     0,A
0468   CBC0         567        SET     0,B
046A   CBC1         568        SET     0,C
046C   CBC2         569        SET     0,D
046E   CBC3         570        SET     0,E
0470   CBC4         571        SET     0,H
0472   CBC5         572        SET     0,L
0474   CBCE         573        SET     1,(HL)
0476   DDCB05CE     574        SET     1,(IX+IND)
047A   FDCB05CE     575        SET     1,(IY+IND)
047E   CBCF         576        SET     1,A
0480   CBC8         577        SET     1,B
0482   CBC9         578        SET     1,C
0484   CBCA         579        SET     1,D
0486   CBCB         580        SET     1,E
0488   CBCC         581        SET     1,H
048A   CBCD         582        SET     1,L
048C   CBD6         583        SET     2,(HL)
048E   DDCB05D6     584        SET     2,(IX+IND)
0492   FDCB05D6     585        SET     2,(IY+IND)
0496   CBD7         586        SET     2,A
0498   CBD0         587        SET     2,B
049A   CBD1         588        SET     2,C
049C   CBD2         589        SET     2,D
049E   CBD3         590        SET     2,E
04A0   CBD4         591        SET     2,H
04A2   CBD5         592        SET     2,L
04A4   CBD8         593        SET     3,B
04A6   CBDE         594        SET     3,(HL)
04A8   DDCB05DE     595        SET     3,(IX+IND)
04AC   FDCB05DE     596        SET     3,(IY+IND)
04B0   CBDF         597        SET     3,A
04B2   CBD9         598        SET     3,C
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
04B4  CBDA       599       SET   3,D
04B6  CBDB       600       SET   3,E
04B8  CBDC       601       SET   3,H
04BA  CBDD       602       SET   3,L
04BC  CBE6       603       SET   4,(HL)
04BE  DDCB05E6   604       SET   4,(IX+IND)
04C2  FDCB05E6   605       SET   4,(IY+IND)
04C6  CBE7       606       SET   4,A
04C8  CBE0       607       SET   4,B
04CA  CBE1       608       SET   4,C
04CC  CBE2       609       SET   4,D
04CE  CBE3       610       SET   4,E
04D0  CBE4       611       SET   4,H
04D2  CBE5       612       SET   4,L
04D4  CBEE       613       SET   5,(HL)
04D6  DDCB05EE   614       SET   5,(IX+IND)
04DA  FDCB05EE   615       SET   5,(IY+IND)
04DE  CBEF       616       SET   5,A
04E0  CBE8       617       SET   5,B
04E2  CBE9       618       SET   5,C
04E4  CBEA       619       SET   5,D
04E6  CBEB       620       SET   5,E
04E8  CBEC       621       SET   5,H
04EA  CBED       622       SET   5,L
04EC  CBF6       623       SET   6,(HL)
04EE  DDCB05F6   624       SET   6,(IX+IND)
04F2  FDCB05F6   625       SET   6,(IY+IND)
04F6  CBF7       626       SET   6,A
04F8  CBF0       627       SET   6,B
04FA  CBF1       628       SET   6,C
04FC  CBF2       629       SET   6,D
04FE  CBF3       630       SET   6,E
0500  CBF4       631       SET   6,H
0502  CBF5       632       SET   6,L
0504  CBFE       633       SET   7,(HL)
0506  DDCB05FE   634       SET   7,(IX+IND)
050A  FDCB05FE   635       SET   7,(IY+IND)
050E  CBFF       636       SET   7,A
0510  CBF8       637       SET   7,B
0512  CBF9       638       SET   7,C
0514  CBFA       639       SET   7,D
0516  CBFB       640       SET   7,E
0518  CBFC       641       SET   7,H
051A  CBFD       642       SET   7,L
051C  CB26       643       SLA   (HL)
051E  DDCB0526   644       SLA   (IX+IND)
0522  FDCB0526   645       SLA   (IY+IND)
0526  CB27       646       SLA   A
0528  CB20       647       SLA   B
052A  CB21       648       SLA   C
052C  CB22       649       SLA   D
052E  CB23       650       SLA   E
0530  CB24       651       SLA   H
0532  CB25       652       SLA   L
0534  CB2E       653       SRA   (HL)
0536  DDCB052E   654       SRA   (IX+IND)
053A  FDCB052E   655       SRA   (IY+IND)
053E  CB2F       656       SRA   A
0540  CB28       657       SRA   B
0542  CB29       658       SRA   C
0544  CB2A       659       SRA   D
0546  CB2B       660       SRA   E
0548  CB2C       661       SRA   H
054A  CB2D       662       SRA   L
054C  CB3E       663       SRL   (HL)
054E  DDCB053E   664       SRL   (IX+IND)
0552  FDCB053E   665       SRL   (IY+IND)
0556  CB3F       666       SRL   A
0558  CB38       667       SRL   B
055A  CB39       668       SRL   C
055C  CB3A       669       SRL   D
055E  CB3B       670       SRL   E
0560  CB3C       671       SRL   H
0562  CB3D       672       SRL   L
0564  96         673       SUB   (HL)
0565  DD9605     674       SUB   (IX+IND)
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
0568  FD9605     675       SUB   (IY+IND)
056B  97         676       SUB   A
056C  90         677       SUB   B
056D  91         678       SUB   C
056E  92         679       SUB   D
056F  93         680       SUB   E
0570  94         681       SUB   H
0571  95         682       SUB   L
0572  D620       683       SUB   N
0574  AE         684       XOR   (HL)
0575  DDAE05     685       XOR   (IX+IND)
0578  FDAE05     686       XOR   (IY+IND)
057B  AF         687       XOR   A
057C  A8         688       XOR   B
057D  A9         689       XOR   C
057E  AA         690       XOR   D
057F  AB         691       XOR   E
0580  AC         692       XOR   H
0581  AD         693       XOR   L
0582  EE20       694       XOR   N
0584             695 NN    DEFS  2
                 696 IND   EQU   5
                 697 M     EQU   10H
                 698 N     EQU   20H
                 699 DIS   EQU   30H
                 700       END
```

Error Messages

The TRS-80 Assembler/Editor recognizes two types of errors:

1) **Command errors** – The error message is printed and control is transferred to command level.
2) **Assembler errors** – These three types of errors may occur while executing an Assemble command.
   a) **Terminal** – Assembly is terminated and control is returned to command level.
   b) **Fatal** – The line containing the error is not further processed and no object code is generated for that line. Assembly proceeds with next source line.
   c) **Warning** – The error message is printed and assembly of the line containing the warning continues. The resulting object code may not be what the programmer intended.

Following is a list of all errors and an explanation of each.

COMMAND ERRORS

1) BAD PARAMETER(S)

Causes –

Increment specified as zero.

```
    I1ØØ,Ø
```

Parameter(s) not properly separated or terminated.

```
    P 1ØØØ,2ØØØ    (comma should be colon)
    P1Ø:2ØL        (garbage at end of command)
```

Specified line number or increment is greater than 65529.

```
    E66ØØØ
```

Line specification is not a number or one of the special characters #, ., or *

```
    P @:2ØØ
```

Second line number of range is less than first line number of range.

```
    P 2ØØ:1ØØ
```

Specified cassette filename:
i) is longer than 6 characters
ii) does not begin with an alphabetic character
iii) contains characters which are not alphanumeric

```
    W 1 TEST
    L TESTFILE
```

An unsupported assembly switch was specified or the slashes were misplaced or omitted.

```
    A/NO/NL
    A NO
    A ZZ
```

An attempt was made to load a cassette which was not written by the Editor or for some other reason cannot be properly read.

BUFFER FULL

There is no room in the edit buffer for adding text.

ILLEGAL COMMAND

The first character of the command line does not specify a valid Editor/Assembler command.

```
    *Z1ØØØ:12ØØ
```

LINE NUMBER TOO LARGE

Causes

Renumbering (using the N command with the specified starting line number and increment would cause line(s) to be assigned numbers greater than 65529. The renumbering is not performed.

```
    N6ØØØØ,1ØØØ  (if there are more than 6 lines of text
                 in the edit buffer)
```

The next line number to be generated by Insert or Replace would exceed 65529.

```
    *I 64ØØØ,16ØØ
    64ØØØ HELLO
    LINE NUMBER TOO LARGE
    *                           (next number would be 656ØØ)
```

NO ROOM BETWEEN LINES

The next line number to be generated by Insert or Replace would be greater than or equal to the line number of the next line of text in the edit buffer. The increment must be decreased or the lines in the buffer renumbered.

```
    *P 1ØØ:115
    ØØ1ØØ HEY
    ØØ114 YOU
    *I 112,2
    ØØ112 TEST
    NO ROOM BETWEEN LINES
    *                        (next number would be 114
                             which already exists)
```

NO SUCH LINE

A line specified by a command does not exist.

```
    *P1ØØ:115
    ØØ1ØØ HEY
    ØØ114 YOU
    *E112
    NO SUCH LINE                (there is no line 112)
```

NO TEXT IN BUFFER

A command requiring text in the buffer was issued when the edit buffer was empty.
The commands Load, Insert, Basic, and System can be executed when the buffer is empty. All other commands require at least one line of text to be in the buffer.

```
*D#:*                              (empty the buffer)
*P
NO TEXT IN BUFFER
```

STRING NOT FOUND

The string being searched for by the Find command could not be found between the current line and the end of the buffer.

TERMINAL ERRORS

1) SYMBOL TABLE OVERFLOW

There is not enough memory for the assembler's symbol table.

FATAL ERRORS

BAD LABEL

The character string found in the label field of the source statement
a) begins with a non alphabetic character
b) is no longer than 6 characters
c) contains characters which are not alphanumeric

EXPRESSION ERROR

The operand field contains an ill-formed expression.

ILLEGAL ADDRESSING MODE

The operand field does not specify an addressing mode which is illegal with the specified opcode.

ILLEGAL OPCODE

The character string found in the opcode field of the source statement is not a recognized instruction mnemonic or assembler pseudo-op.

MISSING INFORMATION

Information vital to the correct assembly of the source line was not provided. The opcode is missing or the operands are not completely specified.

WARNINGS

BRANCH OUT OF RANGE

The destination (D) of a relative jump instruction (JR, DJNZ) is not within the range (LC-128 $\leq$D $\leq$(LC 127) where LC is the address assigned to the first byte of the jump instruction. The instruction is assembled as a branch to itself by forcing the offset to hex FE.

FIELD OVERFLOW

A number or expression result specified in the operand field is too large for the specified instruction operand. The result is truncated to the largest allowable number of bits. For example, BIT 9, A would cause such an error.

MULTIPLY DEFINED SYMBOL

The operand field contains a reference to the symbol which has been multiply defined. The first definition of the symbol is used to assemble the line.

MULTIPLE DEFINITION

The source line is attempting to illegally redefine a symbol. The original definition of the symbol is retained. Symbols may only be redefined by the DEFL pseudo-op and only if they were originally defined by DEFL.

NO END STATEMENT

The program end statement is missing.

UNDEFINED SYMBOL

The operand field contains a reference to a symbol which has not been defined. A value of zero is used for the undefined symbol.

LEVEL I BASIC Addresses

CURSOR
LOCATION

4068H
Contains a 3C00H to 3FFFH which is the current cursor position on screen.

KEYBOARD SCAN

```
WAIT        CALL        0B40H       ;SCAN

            JR          Z,WAIT      ;Z=1 IF KB CLEAR
```

(A-register contains input byte, Input byte is displayed at current cursor).

DISPLAY BYTE
AT CURSOR

```
PUSH        DE                      ;MUST SAVE
PUSH        IY                      ;     DE & IY

LD          A,20H                   ;BYTE TO DISPLAY

RST         10H                     ;DISPLAY BYTE

POP         IY                      ;RESTORE

POP         DE                      ;     DE & IY
```

TURN ON
CASSETTE

```
CALL        0FE9H
```

(On board cassette is turned on via remote plug)

SAVE MEMORY
TO CASSETTE

```
CALL        0FE9H                   ;TURN ON CASSETTE
LD          HL,7000H                ;START ADDRESS
LD          DE,7100H                ;LAST+1 ADDRESS
CALL        0F4BH                   ;SAVE IT
```

(Cassette is turned off)

LOAD MEMORY
FROM CASSETTE

```
CALL        0EF4H                   ;TURN ON & READ
```

(On return HL = last + 1 address
Z = 0 if checksum error
Z = 1 if checksum OK)

(Cassette is turned off)

RETURN TO
LEVEL I BASIC

```
Press       RESET
JP          0                       ;POWER UP
JP          01C9H                   ;RE-ENTRY
```

(Re-entry gives a READY)

RETURN TO TBUG
(UNDER LEVEL I
BASIC)

Set a Breakpoint to next
opcode address.

```
JP          40B1H                   ;RE-ENTER TBUG
```

LEVEL II BASIC Addresses

CURSOR
LOCATION

4020H
(Contains 3C00H to 3FFF which is the current cursor position on screen)

TURN ON CURSOR
CHARACTER

```
        PUSH    DE      ;MUST SAVE
        PUSH    IY      ;    DE & IY
        LD      A,0EH   ;0EH IS CURSOR BYTE
        CALL    33H     ;DISPLAY ROUTINE
        POP     IY      ;RESTORE
        POP     DE      ;    DE & IY
```

KEYBOARD SCAN

```
        PUSH    DE      ;MUST SAVE
        PUSH    IY      ; DE & IY
AGN     CALL    2BH     ;SCAN ROUTINE
        OR      A       ;A=0 IF KB CLEAR
        JR      Z,AGN   ;BRANCH IF NO BYTE
        POP     IY      ;RESTORE
        POP     DE      ; DE & IY
```

(A register contains byte when loop falls through.)
(Byte is NOT displayed on screen!)

DISPLAY BYTE
AT CURSOR

```
        PUSH    DE      ;MUST SAVE
        PUSH    IY      ;    DE & IY
        LD      A,20H   ;BYTE TO DISPLAY
        CALL    33H     ;DISPLAY
        POP     IY      ;RESTORE
        POP     DE      ;    DE & IY
```

DEFINE DRIVE

```
;A-REGISTER SPECIFIES CASSETTE
        LD      A,0     ;ON BOARD CASSETTE
        CALL    0212H   ;DEFINE DRIVE
```

WRITE LEADER
AND SYNC BYTE

```
        CALL    0287H
```

TURN OFF
CASSETTE

```
        CALL    01F8H
```

SAVE MEMORY
TO CASSETTE

```
        LD      A,0     ;ON BOARD CASSETTE
        CALL    0212H   ;DEFINE DRIVE
        CALL    0287H   ;WRITE LEADER
        LD      A,20H   ;BYTE TO RECORD
        CALL    0264H   ;OUTPUT BYTE
```

(USER must CALL 264H often enough to keep up with 500 baud. Timing is automatic.)

```
        CALL    01F8H   ;CASSETTE OFF
```

LOOK FOR LEADER
AND SYNC BYTE

```
        CALL    0296H
```

LOAD MEMORY
FROM CASSETTE

```
        LD      A,0
        CALL    0212H   ;DEFINE DRIVE
        CALL    0296H   ;FIND SYNC BYTE
        CALL    0235H   ;READ ONE BYTE
```

(User must CALL 0235H often enough to keep up with 500 baud. User must do own checksum if desired. A-register contains byte read.) The user must turn off the Cassette (CALL 01F8H) when all bytes have been read.

RETURN TO LEVEL II BASIC	Press	RESET	
	JP	0	;LIKE POWER UP
	JP	1A19H	;RE-ENTRY
	(RE-ENTRY gives a READY 〉)		
RETURN TO TBUG (UNDER LEVEL II BASIC)	Set a Breakpoint to next opcode address.		
	JP	43A0H	;RE-ENTER TBUG

LEVEL II BASIC MEMORY MAP

ADDRESS		
DECIMAL	HEXIDECIMAL	
0	0000	
		LEVEL II BASIC ROM
12288	3000	
		RESERVED
14302	37DE	COMMUNICATION STATUS ADDRESS
14303	37DF	COMMUNICATION DATA ADDRESS
14304	37E0	INTERRUPT LATCH ADDRESS
14305	37E1	DISK DRIVE SELECT LATCH ADDRESS
14308	37E4	CASSETTE SELECT LATCH ADDRESS
14312	37E8	LINE PRINTER ADDRESS
14316	37EC	FLOPPY DISK CONTROLLER ADDRESS
14336	3800	
		TRS-80 KEYBOARD MEMORY
15360	3000	
		TRS-80 CRT VIDEO MEMORY
16383	3FFF	
16384	4000	
		LEVEL II BASIC FIXED RAM VECTORS (RST'S 1 THROUGH 7)
16402	4012	
16405	4015	KEYBOARD DEVICE CONTROL BLOCK
		DCB + 0 = DCB TYPE
		+ 1 = DRIVER ADDRESS
		+ 2 = DRIVER ADDRESS
		+ 3 = ∅
		+ 4 = ∅
		+ 5 = ∅
		+ 6 = 'K'
		+ 7 = 'I'
16413	401D	VIDEO DISPLAY CONTROL BLOCK
		DCB + 0 = DCB TYPE
		+ 1 = DRIVER ADDRESS (LSB)
		+ 2 = DRIVER ADDRESS (MSB)
		+ 3 = CURSOR POS N (LSB)
		+ 4 = CURSOR POS N (MSB)
		+ 5 = CURSOR CHARACTER
		+ 6 = 'D'
16421	4025	+ 7 = '0'
		LINE PRINTER CONTROL BLOCK
		DCB + 0 = DCB TYPE
		+ 1 = DRIVER ADDRESS (LSB)
		+ 2 = DRIVER ADDRESS (MSB)
		+ 3 = LINES/PAGE
		+ 4 = LINE COUNTER
		+ 5 = ∅
		+ 6 = 'P'
		+ 7 = 'R'

16429	402D	
		RESERVED
16463	404F	
16464	4050	FDC INTERRUPT VECTOR
16466	4052	COMMUNICATIONS INTERRUPT VECTOR
16468	4054	RESERVED
16476	405C	
16478	405E	25 MSEC HEARTBEAT INTERRUPT
16512	4080	RESERVED
		LEVEL II BASIC FREE RAM
		RESERVED
16870	41E6	
		I/O BUFFER
17127	42E7	
17128	42E8	ALWAYS ZERO
17129	42E9	
		↓ PROGRAM TEXT
		↓ SIMPLE VARIABLES
		↓ ARRAYS
		↓ STRING VARIABLE NAMES AND OVERHEAD
		FREE MEMORY
		↑ STACK
		↑ STRING SPACE
		SPACE RESERVED FOR MACHINE LANGUAGE ROUTINES MIXED WITH BASIC – IF MEMORY SIZE SET
20479 (4K) 32767 (16K)	4FFF (4K) 7FFF (16K)	END OF ACTUAL MEMORY

Editor/Assembler Command List

Assemble	*A [[ƀfilename] [/switch[/switch] . . .]]
Basic	*B
Delete	*D [line1[:line2]]
EDIT	*E [line]
Find	*F [string]
Insert	*I line[,inc]
Hardcopy	*H [line1[:line2]]
Load	*L [ƀfilename]
Number	*N [line[,inc]]
Print	*P [line1[:line2]]
Replace	* R [line[,inc]]
Type	*T[line1[:line2]]
Write	*W[ƀfilename]

Important Information for Cassette Users

Using Your Cassette Deck

Many factors affect the performance of a cassette system. The most significant one is volume. Too low a volume may cause some of the information to be missed. Too high a volume may cause distortion and result in the transfer of background noise as valid information.

Three different cassette models have been supplied with the TRS-80 system – the CTR-40, CTR-41 and CTR-80. Each model has its own loading characteristics. The table below gives suggested volume ranges for each of the CTR models.

Notice that volume ranges for LEVEL I and LEVEL II are different. This is because the LEVEL II data transfer rate is faster (500 baud vs. 250 baud). Also, notice that pre-recorded Radio Shack programs need a slightly **higher** volume setting than that required by your own CSAVEd tapes. The prerecorded tapes are produced with high-speed audio equipment at a slightly lower level than the CSAVE process provides.

RECORDER	USER-GENERATED TAPES		PRE-RECORDED RADIO SHACK TAPES	
MODEL	LEVEL I	LEVEL II	LEVEL I	LEVEL II
CTR-40	YELLOW LINE	RED LINE	YELLOW LINE	RED LINE
CTR-41	6—8	4—6	6½—8½	5—7
CTR-80	4½—6½	3—5	5½—7½	2½—5

Recommended Volume Settings for RADIO SHACK Cassette Decks

(With CTR-40 and CTR-80, to increase volume, turn the control to the left. With CTR-41, turn control to the right.)

When information is being loaded from the cassette tape, two asterisks will appear on the screen. The one on the right will flash on or off each time a new line of program is read in. If the asterisks do not appear, or the one on the right does not flash, then the volume setting is probably too low. If the asterisks appear but one is not flashing, try increasing the volume setting. Use the reset button to stop the cassette and return control to you if loading problems occur. (Asterisks do not blink when loading data from cassette.)

Radio Shack programs are recorded at least twice on each tape (usually once on each side). You should do the same when you record programs on tape. This will give you a back-up if one does not load properly or if it becomes damaged.

Important Note: The CTR-41 requires that you keep the supplied "dummy plug" in the **MIC** jack at all times. However, the CTR-40 and the CTR-80 should never be used with the "dummy plug."

LEVEL I

Sometimes you will get an error message during an attempted CLOAD. This means that some information was lost or garbled. Adjust the volume level slightly and try again.

LEVEL II

In case of an error message, proceed as above. In LEVEL II, there is also a rare case in which the program has not loaded correctly *even though no error is generated.* So, after CLOADing a program, be sure to LIST it. If some data was garbled, then at some point in the listing, the display will be filled with meaningless words and characters. Adjust the volume and try again.

Hints and Tips

Computer tapes should be stored in a relatively dust-free area (a cassette case is recommended) and protected from high temperatures. Magnetic and electrical fields may alter recorded information, so avoid them (i.e. household appliances, power sources such as transformers and television sets, etc.).

The cassette deck supplied with TRS-80 is very compatible with the system and will perform its duties with great success. To keep the cassette deck in top condition and thus minimize your problems, you should periodically perform some routine maintenance on it. Dirty heads can cause as much as a 50% loss in volume. Also, heads become magnetized with use and may cause distortion. We recommend that you clean the head, capstan and pinch roller after every four hours of operation. Heads on new recorders should always be cleaned before use.

Note: Cassette cleaning and demagnetizing accessories are available from your local Radio Shack store.

All Radio Shack computer programs are distributed on an "AS IS" basis without warranty.

Radio Shack shall have no liability or responsibility to customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by computer equipment or programs sold by Radio Shack, including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer or computer programs.

NOTE: Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or program are satisfactory.

Refer to User's Manual for warranties. Failure to adhere to procedures set forth in User's Manual may result in the loss of warranties.

RADIO SHACK TC A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA
280-316 VICTORIA ROAD
RYDALMERE N S W 2116

BELGIUM
PARC INDUSTRIEL DE NANINNE
5140 NANINNE

U K
BILSTON ROAD WEDNESBURY
WEST MIDLANDS WS10 7JN

578-PERKCO—298-0072 REV B

PRINTED IN U.S.A.