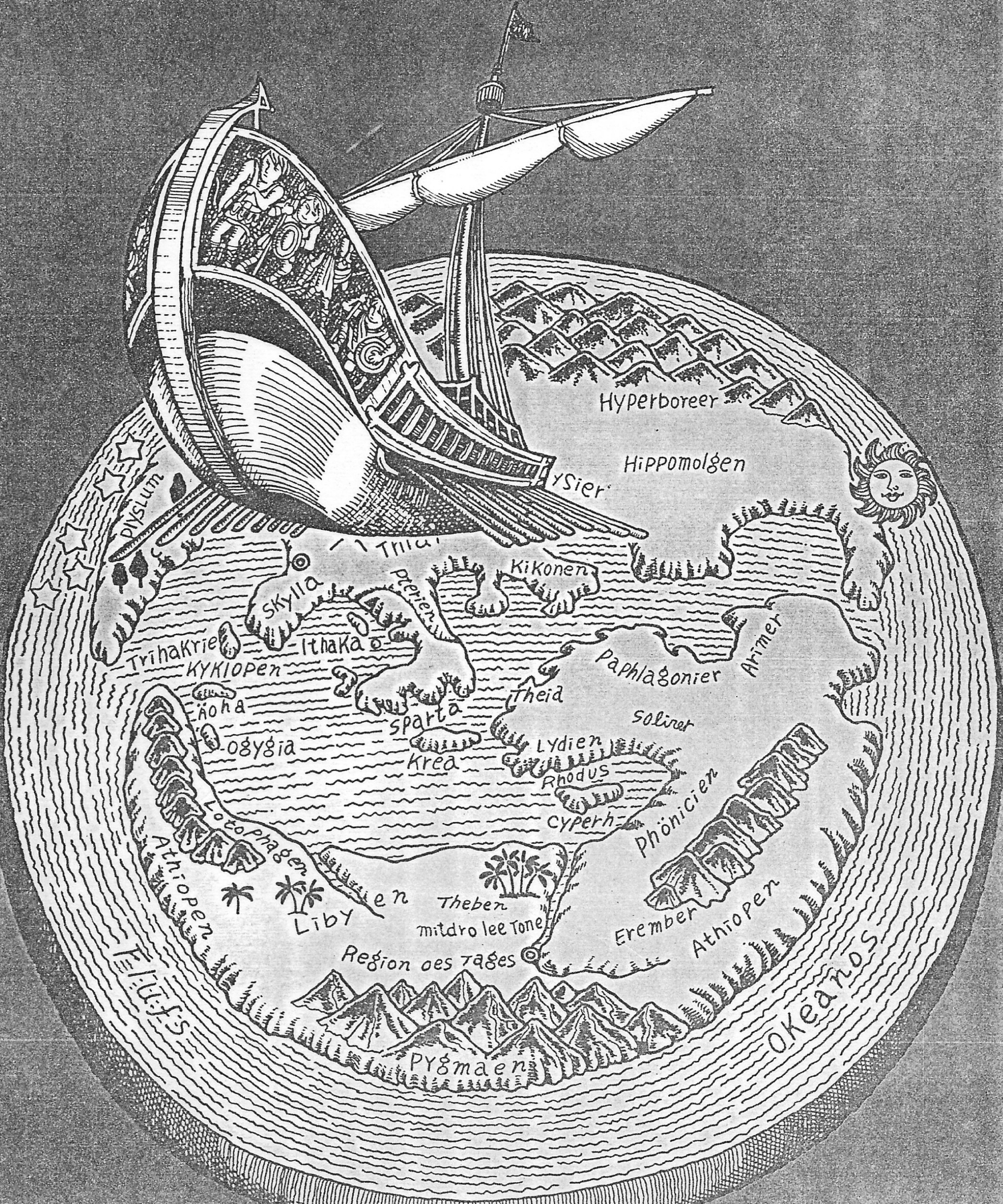
MACHINE LANGUAGE

MZ-80K

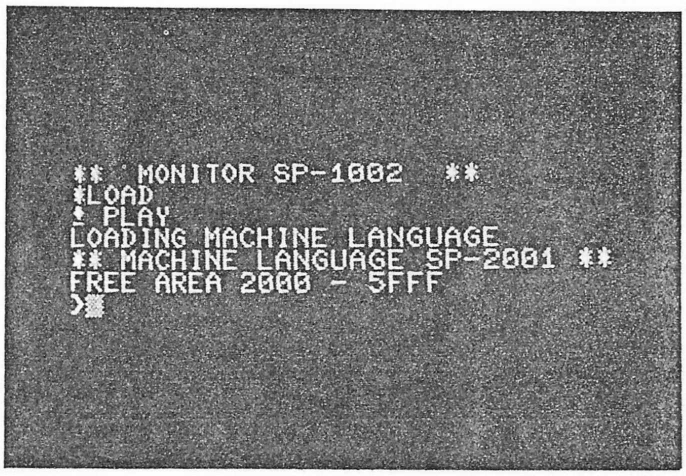


SHARP CORPORATION

Table of contents

What is Machine Language?

Machine Language is provided with a simple debugging function for the microprocessor of the Z80 series. This enables high-speed processing and programming of particular input/output functions. This program is supplied being recorded on a cassette tape, just like BASIC, thus requiring the same loading procedure as that of BASIC. (For loading procedure of BASIC, refer to Manual "BASIC MZ-80K".)
On completion of program loading, the range of Free Area accessible by the user is outputted and a command will be waited for at the symbol ">".
The photo at right shows the steps to this point.



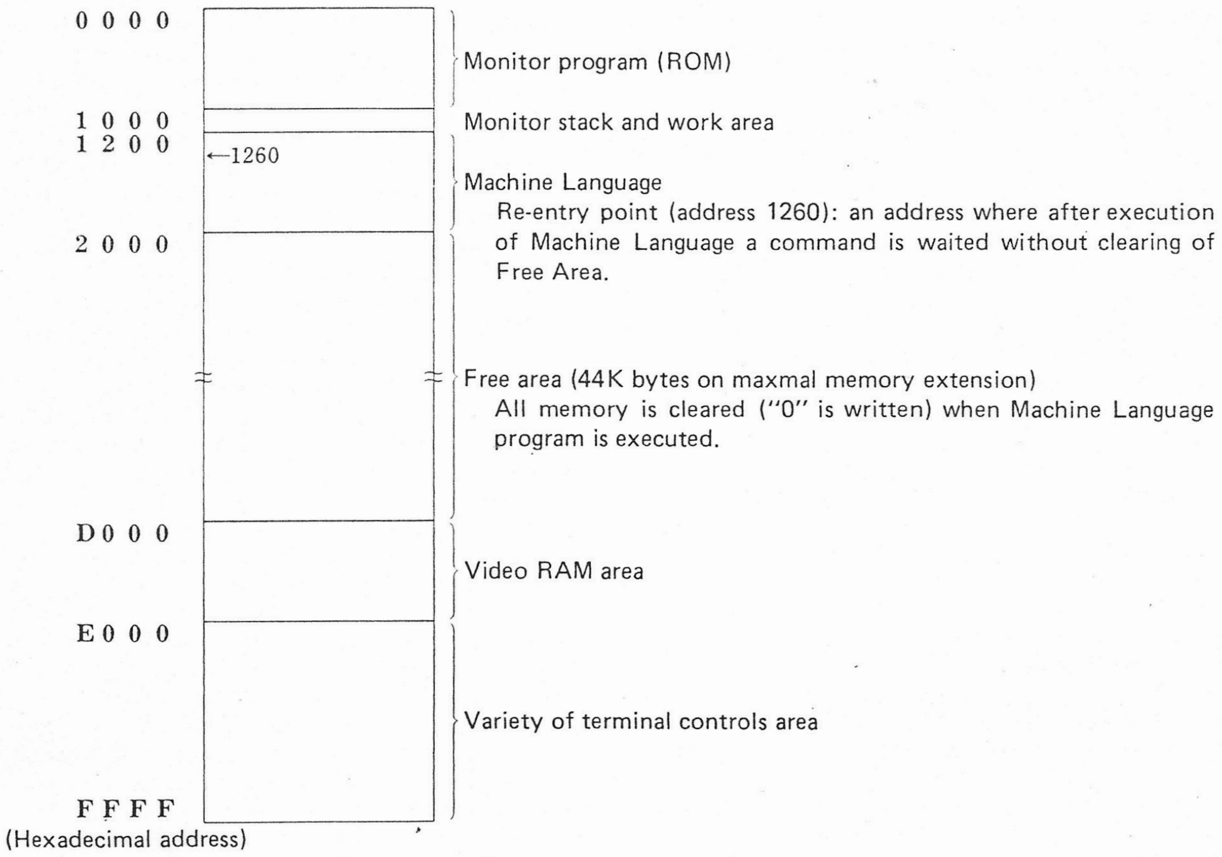
```
** MONITOR SP-1002  **
*LOAD
* PLAY
LOADING MACHINE LANGUAGE
** MACHINE LANGUAGE SP-2001 **
FREE AREA 2000 - 5FFF
>
```

Command	Contents
W (memory Write)	Writes hexadecimal data in succession from specified memory address.
M (Memory dump)	Displays or modifies data of specified memory block in hexadecimal notation.
B (Break point)	Sets the specified number of break points in specified memory address.
&	Clears all the break points (max. 9) set by B command.
G (Go)	Shifts CPU control to specified memory address.
A (Accumulator)	Displays or modifies the contents of F, A, B, C, D, E, H and L in hexadecimal notation.
C (Complementary)	Displays or modifies the contents of F', A', B', C', D', E', H' and H' in hexadecimal notation.
P (Program counter)	Displays or modifies the contents of PC, SP, IX, IY and I in hexadecimal notation.
R (Register)	Displays all the registers of A, C and P commands simultaneously in hexadecimal notation (modification impossible).
X (TRANSfer)	Transfers specified memory block to specified address.
S (Save)	Records and stores specified memory block with its file name.
V (Verify)	Verifies the cassette file and memory block specified by the file name.
Y (Yank)	Reads the cassette file specified by the file name into memory block.
#	Prints the contents displayed on the CRT screen on the printer at the same time.
!	Shifts CPU control to monitor.

- Command error is invalidated to wait for the next one.
- In addressing by M, W, G, S or X command, data in any notation other than hexadecimal are invalidated. Also CPU is in waiting condition until hexadecimal data corresponding to the command are given.
- Data in M, B, A, C or P command can be modified by means of cursor, but the data must be corresponding to one of the command.
- Areas other than Free Area are not accessible with M, W, B, X, S, V or Y command.
- M, S, V or Y command can be executed intermittently by means of BREAK key.
- When the break point set by B command becomes valid, R command is automatically executed.
- The max. number of break points B command can set is 9. The number of appointments lies from 0 (same as clear of break point) up to E (until the 14th execution is broken).

Memory map

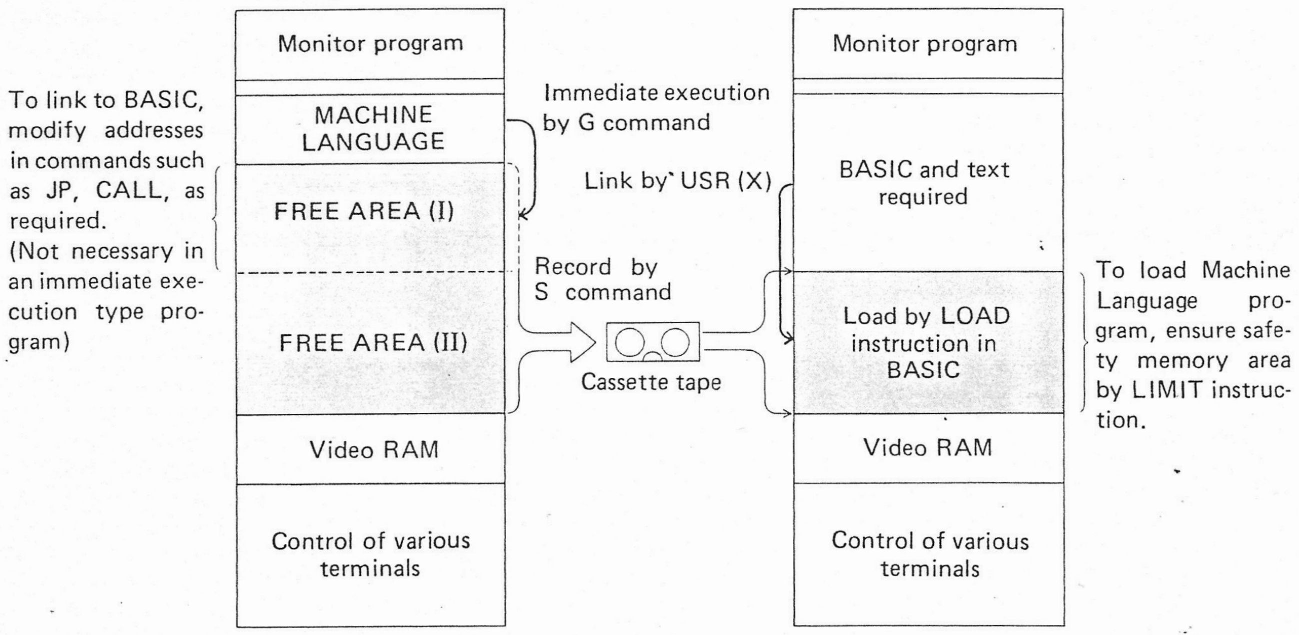
The memory map composed when loading Machine Language is as follows.



0000
1000
1200
←1260
2000
D000
E000
FFFF
(Hexadecimal address)
Monitor program (ROM)
Monitor stack and work area
Machine Language
Re-entry point (address 1260): an address where after execution of Machine Language a command is waited without clearing of Free Area.
Free area (44K bytes on maxmal memory extension)
All memory is cleared ("0" is written) when Machine Language program is executed.
Video RAM area
Variety of terminal controls area

Machine Language programming

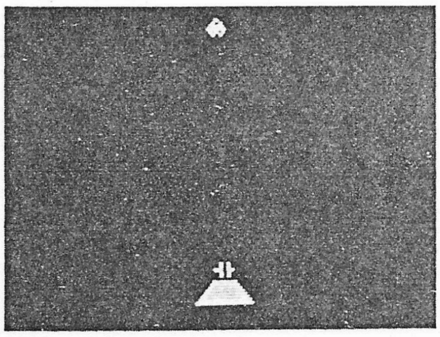
Programs prepared using Machine Language are divided into the following: Immediate execution type (closed program) and BASIC link type (subroutine program).



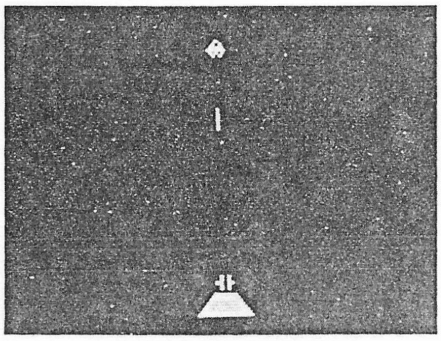
To link to BASIC, modify addresses in commands such as JP, CALL, as required. (Not necessary in an immediate execution type program)
Monitor program
MACHINE LANGUAGE
FREE AREA (I)
FREE AREA (II)
Video RAM
Control of various terminals
Immediate execution by G command
Link by USR (X)
Record by S command
Cassette tape
Monitor program
BASIC and text required
Load by LOAD instruction in BASIC
Video RAM
Control of various terminals
To load Machine Language program, ensure safety memory area by LIMIT instruction.

Machine Language training program

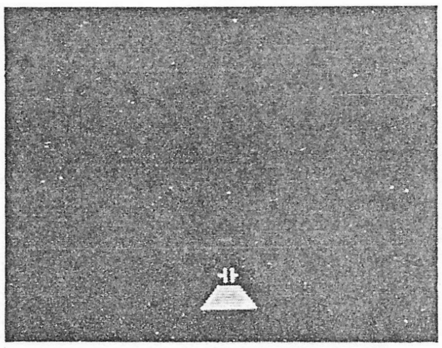
Let us take a brief program in order to explain how to use commands in Machine Language. Byte size is 100-odd. On the screen a UFO and a missile launching ramp are to be displayed. The UFO is supposed to be hit by a missile launched by key-in operation. Below are shown how to prepare subroutines and data area, as well as how to use monitor subroutines.



To display on the CRT screen, store character code in video RAM address corresponding to the position of the element to be shown on the screen. The address are numbered from D000 (Hex) (top left of the screen) to D3E7 (bottom right).

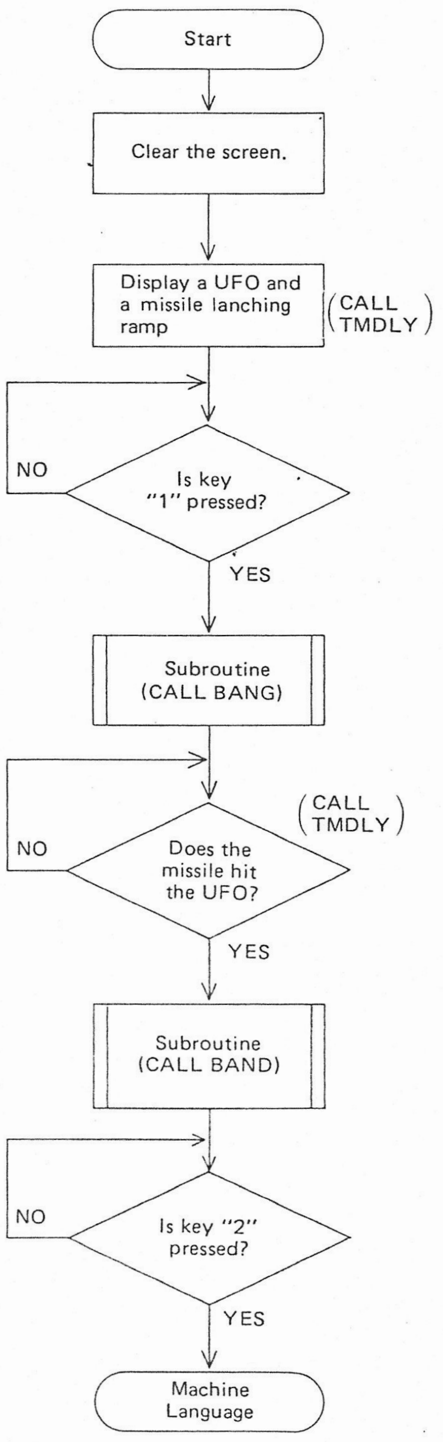


By keying-in "1", the missile is launched with firing sound.

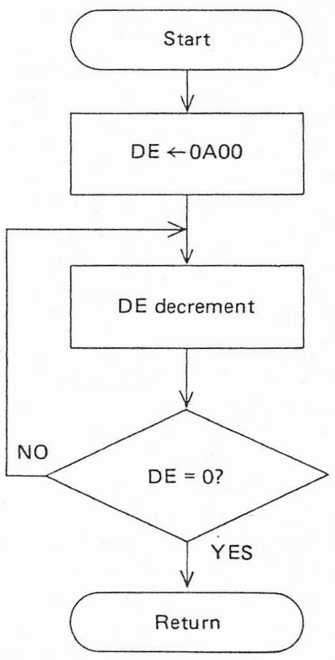


Pressing key "2" after the UFO has been shot down provides wait condition of Machine language command.

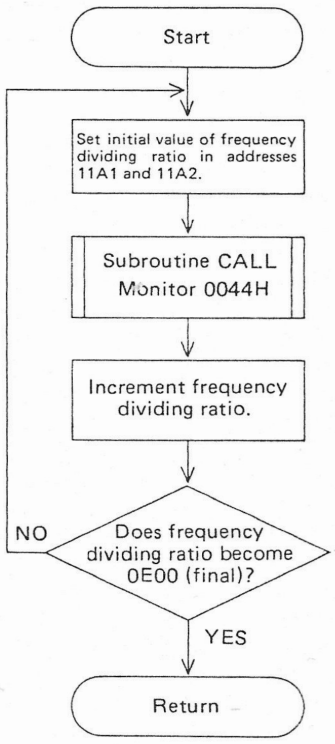
Flowchart of training program



Subroutine TMDLY



Subroutine BANG


Start
Set initial value of frequency dividing ratio in addresses 11A1 and 11A2.
Subroutine CALL Monitor 0044H
Increment frequency dividing ratio.
NO
Does frequency dividing ratio become 0E00 (final)?
YES
Return

In the main program, two subroutines are ready and monitor subroutine is also being called. For video RAM address, refer to Manual "MZ-80 BASIC."

Below is shown a training program assembled according to the Z80 statements. The object codes in the table are in absolute form, 5E00 being the initial address of this program. For these codes to be used unchanged, it is necessary to designate the start address to 5E00 in order to write the program by W command.

```
                    SHARP  Z80  ASSEMBLER
5E00  3E16                     LD      A, 16H
5E02  CD1200                   CALL    0012H
```

Place clear code 16 (Hex) to Acc. and clear the screen by **CALL 0012H.**

```
5E05  060A                     LD      B, 0AH
5E07  21615E                   LD      HL, DATA
5E0A  5E                       LD      E, (HL)
5E0B  23                       INC     HL
5E0C  56                       LD      D, (HL)
5E0D  23                       INC     HL
5E0E  EDA0                     LDI
5E10  CD425E                   CALL    TMDLY
5E13  10F5                     DJNZ    -9
```

Read data and transfer display code to video RAM.

```
5E15  CD1B00                   CALL    001BH
5E18  FE31                     CP      31H
5E1A  20F9                     JR      NZ, -5
```

Key-in waited. By pressing key "1" (code: 31 Hex) the missile is launched.

```
5E1C  CD4B5E                   CALL    BANG
5E1F  216AD2                   LD      HL, D26AH
5E22  3635                     LD      (HL), 35H
5E24  CD425E                   CALL    TMDLY
5E27  3600                     LD      (HL), 00H
5E29  112800                   LD      DE, 0028H
5E2C  ED52                     SBC     HL, DE
5E2E  7E                       LD      A, (HL)
5E2F  FEC7                     CP      C7H
5E31  20EF                     JR      NZ, -15
```

Emitting firing sound, the missile goes up to the UFO.

```
5E33  CD4B5E                   CALL    BANG
5E36  3600                     LD      (HL), 00H
```

The missile destroys the UFO with destroying sound.

```
5E38  CD1B00                   CALL    001BH
5E3B  FE32                     CP      32H
5E3D  20F9                     JR      NZ, -5
5E3F  C36012                   JP      1260H
```

Key-in waited. By pressing key "2" (code: 32 Hex), the following command of Machine Language is waited for.

```
5E42  11000A       TMDLY:      LD      DE, 0A00H
5E45  1B                       DEC     DE
5E46  7A                       LD      A, D
5E47  B3                       OR      E
5E48  20FB                     JR      NZ, -3
5E4A  C9                       RET
```

Time delay subroutine Input **0A00 (Hex)** in DE register and consume time until repeated decrement results in 0.

```
5E4B  E5           BANG:       PUSH    HL
5E4C  016400                   LD      BC, 0064H
5E4F  ED43A111                 LD      (11A1H), BC
5E53  CD4400                   CALL    0044H
5E56  03                       INC     BC
5E57  78                       LD      A, B
5E58  FE0E                     CP      0EH
5E5A  20F3                     JR      NZ, -11
5E5C  CD4700                   CALL    0047H
5E5F  E1                       POP     HL
5E60  C9                       RET
```

Firing and destroying sounds subroutine. Store frequency dividing ratio **0064 (Hex)** = 100 in monitor work areas **11A1** and **11A2 (Hex)**. At **CALL 0044H** a sound of 2MHz/100 = 20 kHz is given. Then increment repeatedly frequency dividing ratio. Sounds which are changed from treble to bass are produced. The sound stops by **CALL 0047H.** Furthermore, **PUSH HL** and **POP HL** are arranged for protection of HL register.

```
5E61  02D1         DATA:       DEFW    D102H
5E63  C7                       DEFB    C7H
5E64  92D2                     DEFW    D292H
5E66  EC                       DEFB    ECH
5E67  B9D2                     DEFW    D2B9H
5E69  4E                       DEFB    4EH
5E6A  BAD2                     DEFW    D2BAH
5E6C  43                       DEFB    43H
5E6D  BBD2                     DEFW    D2BBH
5E6F  4D                       DEFB    4DH
5E70                           END
```

These data are for displaying five characters as to the UFO and missile launching ramp on the CRT screen. Each of them is composed of video RAM address and character codes.

Application of commands

W command

Function

- To write hexadecimal data to specified memory address.

Usage

```
>W  2000
  2000  01  23  45  67  89  AB  CD  EF
  2008  FE [CR]
>
```

(1) Give W (memory Write) command after the command wait ">".
(2) The system displays one space and is in waiting condition for the next key-in.
(3) Hexadecimal address given is displayed. Key-in of data is waited.
(4) After 8 pieces of hexadecimal data are given, the system automatically displays the next address and the following key-in of data is waited.
(5) After inputting necessary data, the wait condition for the next command is returned by means of [CR] key.

Error

- It is impossible to write in any area other than free area.

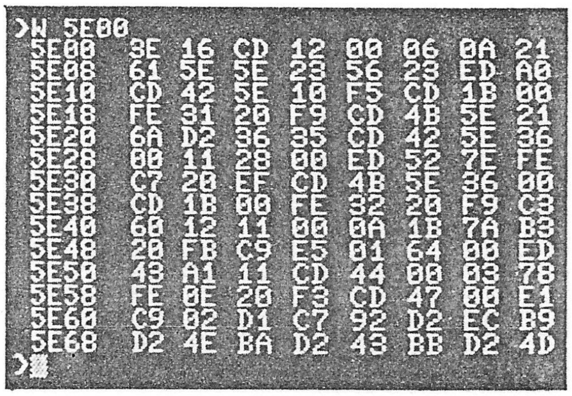
```
>W  1000
  1000
  ? ? ?     <== Address 1000 is not in free area.
```

- Command format is not correct (or when input of the command is desired to be suspended).

```
>W  20 [CR]   <== Give [CR] while inputting the command.
  INVALID
```

Example 1

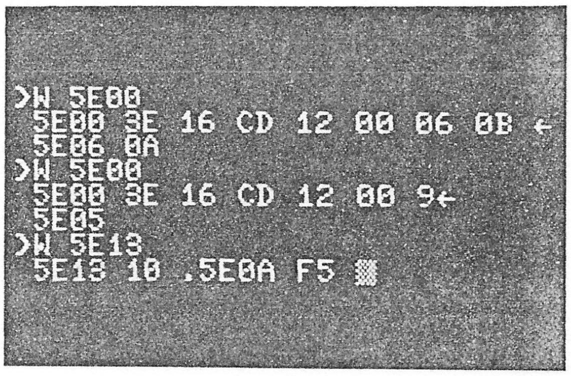
In the photo at right is shown a written training program. (For further explanation of other commands, write as shown in the photo.)



```
>W 5E00
 5E00  3E 16 CD 12 00 06 0A 21
 5E08  61 5E 5E 23 56 23 ED A0
 5E10  CD 42 5E 10 F5 CD 1B 00
 5E18  FE 31 20 F9 CD 4B 5E 21
 5E20  6A D2 36 35 CD 42 5E 36
 5E28  00 11 28 00 ED 52 7E FE
 5E30  C7 20 EF CD 4B 5E 36 00
 5E38  CD 1B 00 FE 32 20 F9 C3
 5E40  60 12 11 00 0A 1B 7A B3
 5E48  20 FB C9 E5 01 64 00 ED
 5E50  43 A1 11 CD 44 00 03 78
 5E58  FE 0E 20 F3 CD 47 00 E1
 5E60  C9 02 D1 C7 92 D2 EC B9
 5E68  D2 4E BA D2 43 BB D2 4D
>▒
```

Example 2

By utilizing cursor key ⇐ as photographed at left, one byte is shifted down, facilitating correction of mistyping. When displacement e such as JR, DJNZ commands, etc. is to be specified, the system is in waiting condition for hexadecimal 4-digit key-in by inputting a period ".". Then a branch address may be directly specified instead of e. The system calculates displacement e automatically from the address and stores it into a required address.



```
>W 5E00
 5E00 3E 16 CD 12 00 06 0B ←
 5E06 0A
>W 5E00
 5E00 3E 16 CD 12 00 9←
 5E05
>W 5E13
 5E13 10 .5E0A F5 ▒
```

M command

Function

- To display data in a specified memory block in hexadecimal notation.
- To modify displayed data by cursor operation.

Usage

```
>M  2000  200F
 2000    01  23  45  67  89  AB  CD  EF
 2008    FE  00  00  00  00  00  00  00
 ■             ↑
```

(1) Give M (Memory dump) command after ">".
(2) The system displays one space and is in waiting condition for start address key-in.
(3) By giving start address on hexadecimal 4-digit basis, an end address is waited.
(4) After hexadecimal 4-digit end address is given, the system displays data of the specified block hexadecimally, and stands allowing cursor operation.
(5) To modify "00" indicated with an arrow to "DC", shift the cursor up to the arrow position and input "DC". By keying-in of [CR], thereafter, the cursor is returned to ■ position.
(6) When the cursor is in ■ position, key-in of [CR] provides wait state of the next command.

Error

- Any data in any area other than free area cannot be displayed.

```
>M  1100  1107
 1100
 ???     <== Address 1100 is not in free area.
```

- The end address to be displayed must be equivalent to or larger than the start address.

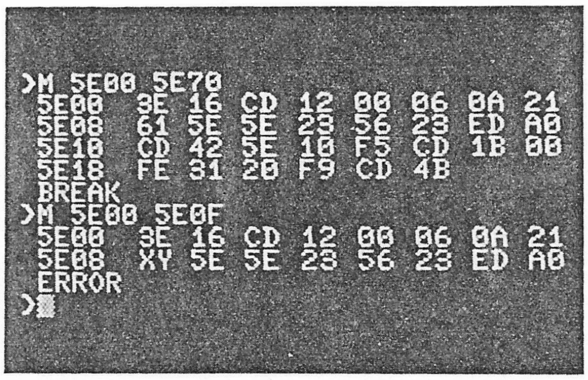
```
>M  2100  2000
 ?
```

- Command format is not correct (or when command input is disired to be suspended).

```
>M  200 [CR]  <== Give [CR] while inputting command.
 INVALID
```

Example 1

Provide "BREAK" in displayed memory block by means of [SHIFT] + [BREAK], and command wait state is resumed. Or input address or data other than hexadecimal ones by cursor operation, "ERROR" is displayed and the system is returned to command wait state.



```
>M 5E00 5E70
 5E00  3E 16 CD 12 00 06 0A 21
 5E08  61 5E 5E 23 56 23 ED A0
 5E10  CD 42 5E 10 F5 CD 1B 00
 5E18  FE 31 20 F9 CD 4B
 BREAK
>M 5E00 5E0F
 5E00  3E 16 CD 12 00 06 0A 21
 5E08  XY 5E 5E 23 56 23 ED A0
 ERROR
>■
```

B command

Function

- To execute instructions the number of times specified in the break counter up to that preceding the address set by the break address.
- The maximum number of break points is 9, and the greatest break counter is "E" (14 times).
- The break points displayed can be modified by cursor operation.

Usage

```
>B
 ADDR  COUNT       ⌴is a space symbol. Input space and distinguish two numbers.
 2004⌴1            Be sure to specify operation code address as break address.
 ■
```

(1) Give B (Break point) command in wait state after ">".
(2) The system displays the break point presently being set and is in wait state for key-in of break condition.
(3) Break address: hexadecimal 4-digit. Give 1 thru E in break counter and key in $\boxed{\text{CR}}$.
(4) Start a new line and wait for an input. Up to 9 points can be set as required.
(5) When the cursor is in ■ position, key-in of $\boxed{\text{CR}}$ provides wait state of the next command.

Error

- Non-existing break point is attempted to be cleared.

```
>B
 ADDR  COUNT
 3000⌴0
 ???             Break count "0" implies that break point is cleared.
```

- No break point can be set in DJNZ instruction.

```
>B
 ADDR  COUNT
 2100⌴1
 DJNZ?
```

- No break point can be set in RST7 instruction.

```
>B
 ADDR  COUNT
 210F⌴A
 RST  7?
```

- Over 9 break points are attempted to be set.

```
>B
 ADDR  COUNT
 2004⌴1
 :
 210D⌴1          Not received because the number of break points exceeds 9.
 OVER
```

- No break point can be set in program counter stack instruction such as CALL instruction. (If check of CALL instruction is desired, a break point is supposed to be set at the destination address of CALL.)

```
>B
 ADDR  COUNT
 5E02⌴1
 CALL?
```

& command

Function

- To clear all the break points being set.

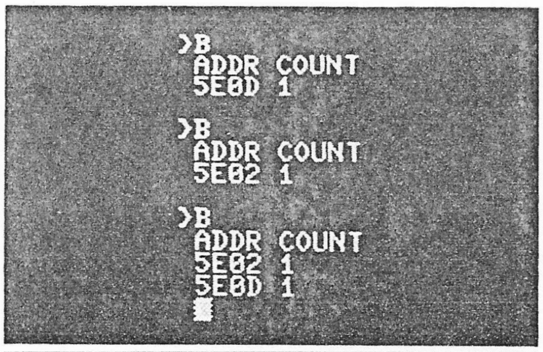
Usage

>&

(1) Give & command after " > ".
(2) Return to wait state of the following command.

Example 1

The right photo shows a case where the break point of address 5E0D is to be changed to address 5E02. ("5E02" displayed in central B command has been changed from "5E0D" by cursor operation.) In this case, a new break point has been set in address 5E02, in addition to 5E0D. (Notice the arrangement in the photo.)



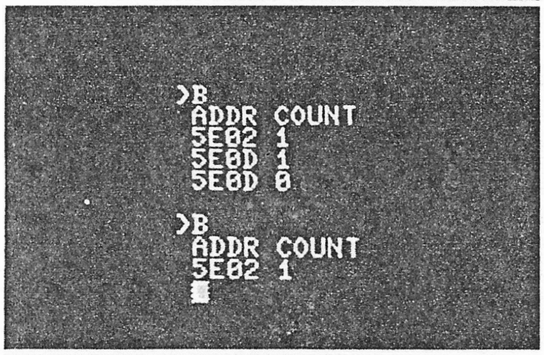
```
>B
 ADDR COUNT
 5E0D 1

>B
 ADDR COUNT
 5E02 1

>B
 ADDR COUNT
 5E02 1
 5E0D 1
 █
```

Example 2

At right is shown a case where the break point of address 5E0D has been cleared.

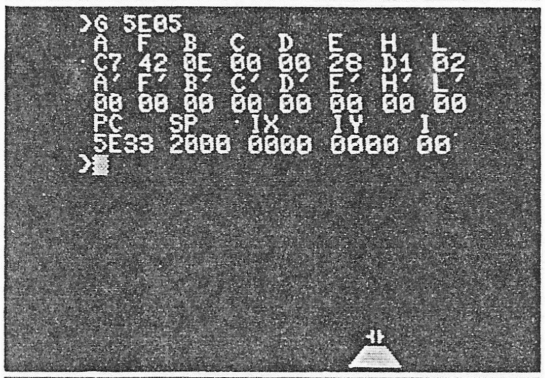


```
>B
 ADDR COUNT
 5E02 1
 5E0D 1
 5E0D 0

>B
 ADDR COUNT
 5E02 1
 █
```

Example 3 **(write down training program in memory.)**

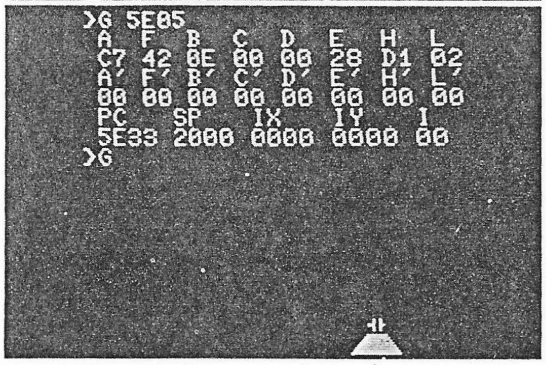
The right photo gives a case where execution starts by G command (see next page) and is broken at address 5E33. The system autoamtically displays the contents of each register and counter at this point, and returns to command wait state. (Set break points in other places so that change of each register and counter be grasped from the conditions of break point and training program flowchart.)



```
>G 5E05
 A  F  B  C  D  E  H  L
 C7 42 0E 00 00 28 D1 02
 A' F' B' C' D' E' H' L'
 00 00 00 00 00 00 00 00
 PC   SP   IX   IY   I
 5E33 2000 0000 0000 00
>█
```

Example 4

At right is given how to restart from the address once interrupted by a break point.
(For explanation of G command, see next page.)
By restarting by G command after clearing all break points by & command the system is brought free from control of Machine Language, and completely depends on its machine language program.



```
>G 5E05
 A  F  B  C  D  E  H  L
 C7 42 0E 00 00 28 D1 02
 A' F' B' C' D' E' H' L'
 00 00 00 00 00 00 00 00
 PC   SP   IX   IY   I
 5E33 2000 0000 0000 00
>G
```

·G command

Function

- To execute a program from a specified start address. Used also to restart from a break point.

Usage

```
>G   2000
```

(1) Give G (Go) command after ">".
(2) The system is in waiting condition for key-in of a start address for executing a program.
(3) By keying-in the start address on hexadecimal 4-digit basis, the system control is transferred to the address.
(4) Stopping of G command execution may be enabled either by the program or a break point.
(5) To restart from a break point, press CR following G command. In this case the break point being set remains.

Error

- Start address must be given on hexadecimal 4-digit basis.

```
>G   200  CR
 INVALID
```

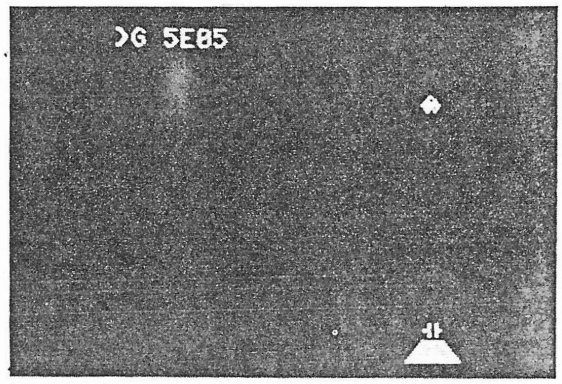
Example

The start address of the training program has been set to be 5E00. Thus input 5E00 by G command. On the CRT screen shown at right, the program is executed from 5E05 and the screen has not yet been cleared with

```
>G   5E05
```

being displayed.

(Note) When the program overruns, turn off the power switch and resume from the beginning. In preparing a lengthy program, therefore, it is recommended to put S command prior to G command. This enables reading from the tape by Y command, setting of break points and debugging.


>G 5E05

A command

Function

- To display the contents of main register set (2 sets of general purpose registers are provided in the Z80-CPU).
- Also to modify the contents of the register by means of cursor.

Usage

```
>A
 A    F    B    C    D    E    H    L
 01   23   45   67   89   AB   CD   EF
 ■
```

(1) Give A (Accumulator) command after ">".
(2) Register name and its contents are lined up as shown above and the cursor appears.
(3) If required, do cursor control to modify the register contents.
(4) By pressing CR , the next command is waited for.

Error

- To correct, 2-digit number is to be put to the specified position. The following shows a data error.

```
>A
 A    F    B    C    D    E    H    L
 01   23   45   671 89   AB   CD   EF
 ERROR
```

2-digit number required

C command

Function

- To display and modify the contents of the complementary register set, which is one of the above general purpose register set.

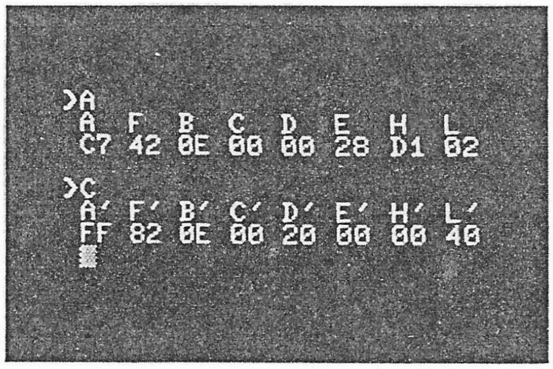
Usage

```
>C
 A'   F'   B'   C'   D'   E'   H'   L'
 01   23   45   67   89   AB   CD   EF
 ■
```

(1) Give C (Complementary) command after ">".
(2) Follow the same steps as A command.

Example

The right photo shows displayed contents of the general purpose registers by A and C commands.



```
>A
A  F  B  C  D  E  H  L
C7 42 0E 00 00 28 D1 02

>C
A' F' B' C' D' E' H' L'
FF 82 0E 00 20 00 00 40
▒
```

P command

Function

- To display the contents of the special purpose register set for the Z80-CPU. Also to modify those contents by cursor control.

Usage

```
>P
 PC    SP    IX    IY    I
 0123  4567  89AB  CDEF  01
 ■
```

(1) By giving P (Program counter) command are displayed the contents of the following special purpose registers; program counter (PC), stack pointer (SP), index registers (IX and IY), and interrupt page address register (I). The cursor is also displayed.

(2) Modification, if required, may be added by cursor control.

Error

- Like error in A command and C command, modification in which the number of digits of the newly given contents is not equal to that of the previous contents causes data error.

R command

Function

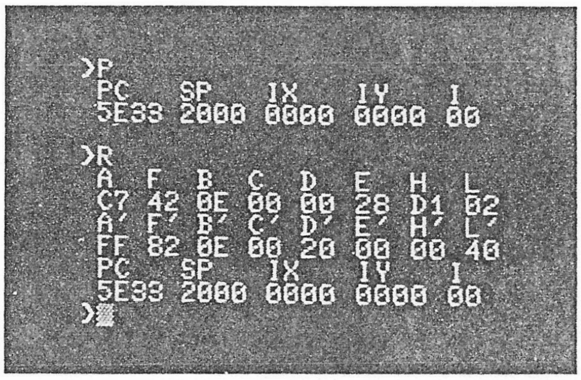
- To display the contents of all Z80-CPU registers. However, modification of the contents is impossible.

```
>R
 A   F   B   C   D   E   H   L
 01  23  45  67  89  AB  CD  EF
 A'  F'  B'  C'  D'  E'  H'  L'
 01  23  45  67  89  AB  CD  EF
 PC    SP    IX    IY    I
 0123  4567  89AB  CDEF  01
>
```

By giving R (Register) command, the register contents as shown above are displayed, and the next command is waited.

Example

The right photo shows the contents of special purpose registers and those of all registers displayed on the CRT screen by giving P command and R command, respectively.



```
>P
 PC   SP   IX   IY   I
 5E33 2000 0000 0000 00

>R
 A  F  B  C  D  E  H  L
 C7 42 0E 00 00 28 D1 02
 A' F' B' C' D' E' H' L'
 FF 82 0E 00 20 00 00 40
 PC   SP   IX   IY   I
 5E33 2000 0000 0000 00
>▒
```

X command

Function

- To transfer a specified memory block to other memory area.

Usage

```
>X
 FROM?  2000  TO?  21FF  TOP?  3000
```

(1) Give X (memory TRANSfer) command after ">".
(2) After displaying of "FROM?", the system is in waiting condition for an initial address of transfer memory block.
(3) Being given the initial address on hexadecimal 4-digit basis, the system displays "TO?" and is in waiting condition for an end address of transfer memory block.
(4) Being given the end address on hexadecimal 4-digit basis, the system displays "TOP?" and is in waiting condition for an initial address of the counter-area.
(5) After the address of counter-area given on hexadecimal 4-digit basis, the specified memory block is transferred and the next command is waited.

Error

- Transfer is performed only in memory area of free area.

```
>X
 FROM?  2000  TO?  2FFF  TOP?  D000
 ???
```

Address D000 is not in free area.

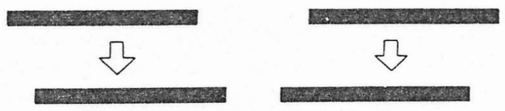
- Command format is not appropriate (or when inputting of command is desired to be suspended).

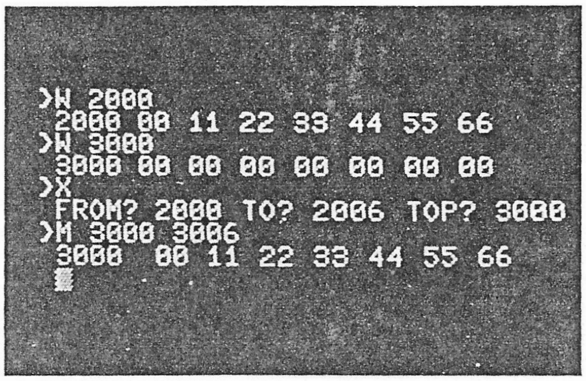
```
>X
 FROM?  2000  TO?  2.[CR]
 INVALID
```

Give $\boxed{CR}$ while inputting command.

Example 1

At right is shown an execution in which data is written from address 2000 to address 2006, address 3000 thru 3006 are cleared, and then X command is given. Note that the transfer as shown below is also accessible.

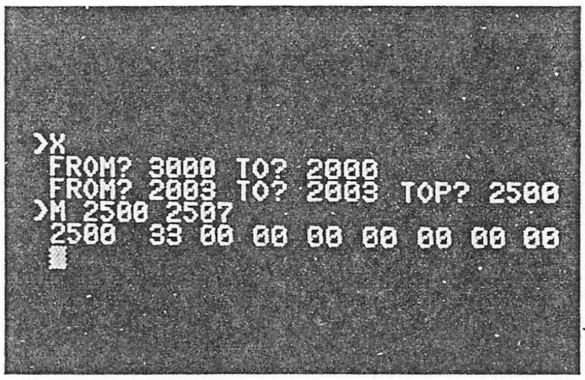




```
>W 2000
 2000 00 11 22 33 44 55 66
>W 3000
 3000 00 00 00 00 00 00 00
>X
 FROM? 2000 TO? 2006 TOP? 3000
>M 3000 3006
 3000  00 11 22 33 44 55 66
```

Example 2

When the end address of memory block to be transferred is smaller than the initial address, the system is returned to waiting condition for key-in of the initial address. In case the end address is the same as initial address, only one byte is transferred.



```
>X
 FROM? 3000 TO? 2000
 FROM? 2003 TO? 2003 TOP? 2500
>M 2500 2507
 2500  33 00 00 00 00 00 00 00
```

S command

Function

- To record and store a specified block in the in-memory machine language program in a cassette tape together with its file name.

Usage

```
>S
  FILENAME?ABC [CR]
  FROM?  2000  TO?  3000
↓ RECORD·PLAY
  WRITING  ABC
  OK        (or ERROR)
```

(1) Give S (Save) command after " > ".
(2) The system displays "FILENAME?" and is in waiting condition for key-in of the file name.
(3) Input the name and press [CR].
(4) Then the system displays "FROM?" and is in waiting condition for key-in of the initial address of memory block to be recorded.
(5) Being given the initial address on hexadecimal 4-digit basis, the system displays "TO?" and is in waiting condition for key-in of the end address of memory block to be recorded.
(6) After the end address given on hexadecimal 4-digit basis, an instruction is given to press RECORD and PLAY buttons of cassette tape recorder.
(7) Pushing of these buttons gives on-record indication. After complete recording, "OK" is displayed. If an error occurs halfway, "ERROR" appears.
(8) To suspend execution of S command, press [BREAK].

Error

- Memory block only in free area is executed by S command.

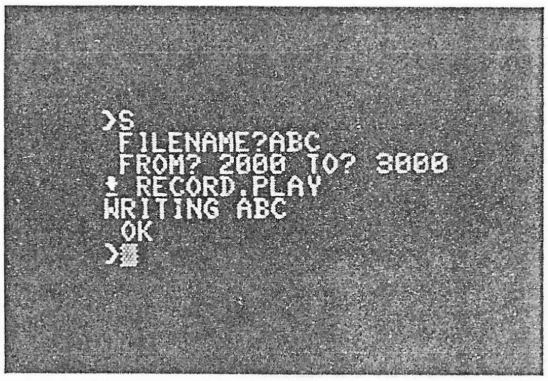
```
>S
  FILENAME?ABC [CR]
  FROM?  1000
  ???  <== Error display when specified outside free area.
>
```

- The initial address must be smaller than the end address.

```
>S
  FILENAME?ABC [CR]
  FROM?  3000  TO?  2000
  FROM? <== The initial and end addresses are inquired again.
```

Example

The right photo shows procedure to store the contents of memory block from address 2000 to address 3000 into a cassette tape together with the file name "ABC".



```
>S
 FILENAME?ABC
 FROM? 2000 TO? 3000
 ↓ RECORD.PLAY
WRITING ABC
 OK
>▒
```

V command

Function

- To verify whether or not the file contents stored in the cassette tape are equal to those of the memory block corresponding to the address.

Usage

```
>V
 FILENAME?ABC [CR]
↓ PLAY
 FOUND  ABC  FROM  XXXX  TO  YYYY
 VERIFYING  ABC
 OK      (or ERROR)
>
```

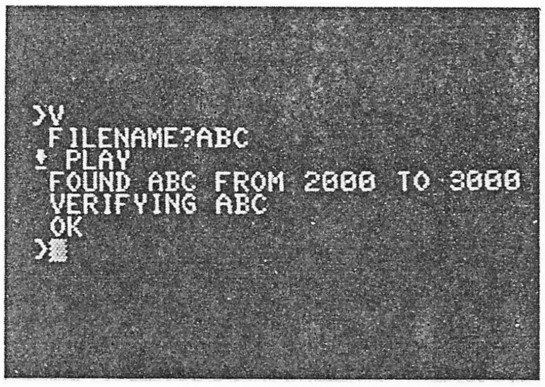
(1) Give V (Verify) command after ">".
(2) The system displays "FILENAME?" and is in waiting condition for the file name to be verified.
(3) After the file name is inputted and [CR] is pressed, an instruction is given to push PLAY button.
(4) Verification indication appears. When verified to be equal, "OK" is indicated, while any error causes to display "ERROR".
(5) To suspend execution of V command, press [BREAK].

Error

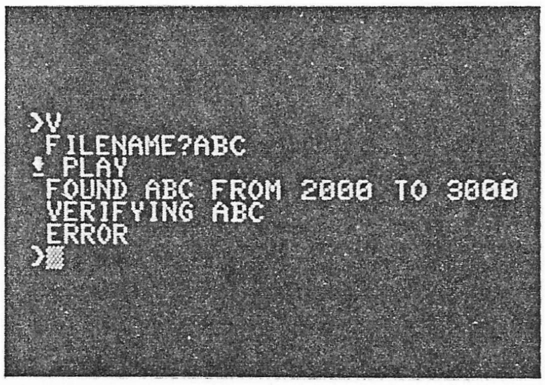
- When the contents of cassette tape and those of memory block corresponding to the address are different, "ERROR" appears.

Example

At right are shown indications when V command is executed to cassette tape file with its name of "ABC" and to memory block. When both are verified to be equal, "OK" is displayed as described above and if any difference found, "ERROR" is displayed.



```
>V
 FILENAME?ABC
↓ PLAY
 FOUND ABC FROM 2000 TO 3000
 VERIFYING ABC
 OK
>▒
```



```
>V
 FILENAME?ABC
↓ PLAY
 FOUND ABC FROM 2000 TO 3000
 VERIFYING ABC
 ERROR
>▒
```

Y command

Function

- To read a file stored on the cassette tape and load it into a memory.
  By specifying the predetermined file name, the particular file can be loaded.

Usage

```
>Y
 FILENAME?ABC [CR]
⬇  PLAY
   FOUND ABC FROM XXXX TO YYYY
   LOADING ABC
   OK       (or ERROR)
>
```

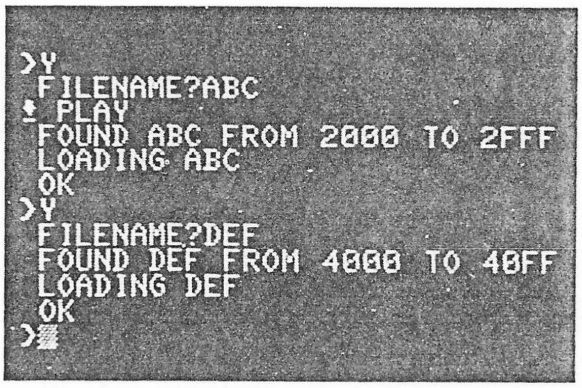
(1) Give Y (Yank) command after ">".
(2) The system displays "FILENAME?" and is in waiting condition for the file name to be read.
(3) After the file name is inputted and [CR] is pressed, an instruction is given to push PLAY button.
(4) By pushing PLAY button, the specified file is found and read. When the file name is not specified, the file first found is read. The file is loaded in memory and it is stored in the memory block from the initial address to the end address specified at saving.
(5) To suspend Y command, press [BREAK] .

Error

- When any error occurs during reading, "ERROR" appears.

Example

At right is shown an example in which after one Y command for file name "ABC", another Y command for file name "DEF" is executed.
File "ABC" is loaded to addresses 2000 thru 2FFF and file "DEF" to 4000 thru 40FF. (The address to load is determined at the time of saving.)



```
>Y
 FILENAME?ABC
⬇ PLAY
 FOUND ABC FROM 2000 TO 2FFF
 LOADING ABC
 OK
>Y
 FILENAME?DEF
 FOUND DEF FROM 4000 TO 40FF
 LOADING DEF
 OK
>▒
```

# command

Function

- To print on a option printer at the same time as displaying on the CRT screen.

Usage

> #

Every setting of # command reverses printer mode.
When starting Machine Language, the printer mode is reset. A # command then enables change-over to printer mode, and the subsequent outputs are represented both on the CRT screen and the printer. Another # command resets printer mode allowing to display on the CRT screen only.

Messages

- NO POWER OR NO CONNECTION (PRINTER)
  Indicates that printer power source is turned OFF or the printer is disconnected from the system.
- ALARM (PRINTER)
  Indicates that abnormality such as paper jamming, etc. has occurred in printer mechanism.
- PAPER EMPTY (PRINTER)
  Indicates that the printer paper needs replacing.

Example

Only in case of execution of M command, CRT screen display and printer printing form are different. On the CRT screen 8 bytes of data are displayed in one line, while 16 bytes are printed in one line of the printer.

```
>M 5E00 5E6F
 5E00   3E 16 CD 12 00 06 0A 21 61 5E 5E 23 56 23 ED A0
 5E10   CD 42 5E 10 F5 CD 1B 00 FE 31 20 F9 CD 4B 5E 21
 5E20   6A D2 36 35 CD 42 5E 36 00 11 28 00 ED 52 7E FE
 5E30   C7 20 EF CD 4B 5E 36 00 CD 1B 00 FE 32 20 F9 C3
 5E40   60 12 11 00 0A 1B 7A B3 20 FB C9 E5 01 64 00 ED
 5E50   43 A1 11 CD 44 00 03 78 FE 0E 20 F3 CD 47 00 E1
 5E60   C9 02 D1 C7 92 D2 EC B9 D2 4E BA D2 43 BB D2 4D
```

## ! command

Function

- To shift the control to the monitor.

Usage

> !

By giving ! command after ">", the control is immediately returned to the monitor. To return to Machine Language, there are the following two manners.

● ＊G O T O $ 1 2 0 0 [CR]

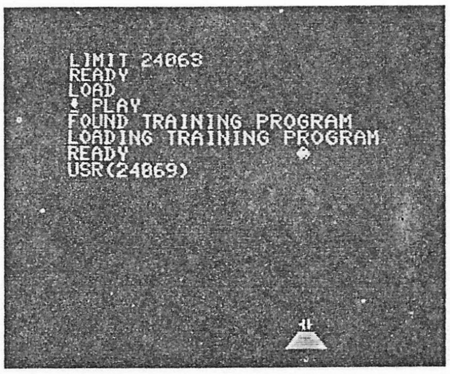
Clear free area and return the stack to its initial state.

● ＊G O T O $ 1 2 6 0 [CR]

Keep free area and return to command wait state.

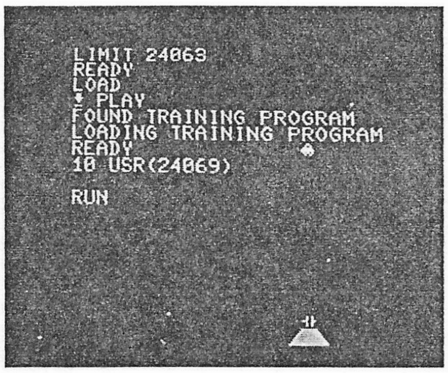
Linkage to BASIC program

A machine language program by this system can be linked to a BASIC program. The version numbers of BASIC to be linked, however, are limited to SP-5010 or after.
The photos below explain a typical method of linking to BASIC program.



```
LIMIT 24063
READY
LOAD
↓ PLAY
FOUND TRAINING PROGRAM
LOADING TRAINING PROGRAM
READY
USR(24069)
```

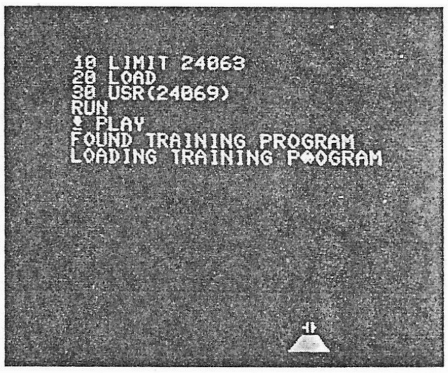
Photo 1



```
LIMIT 24063
READY
LOAD
↓ PLAY
FOUND TRAINING PROGRAM
LOADING TRAINING PROGRAM
READY
10 USR(24069)

RUN
```

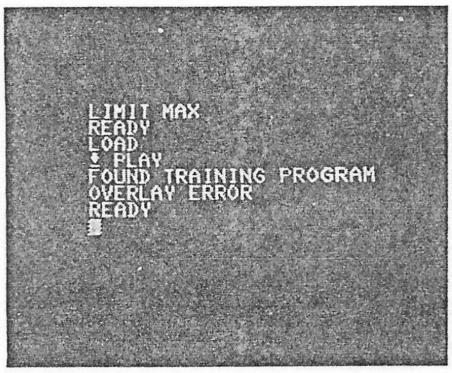
Photo 2



```
10 LIMIT 24063
20 LOAD
30 USR(24069)
RUN
↓ PLAY
FOUND TRAINING PROGRAM
LOADING TRAINING PROGRAM
```

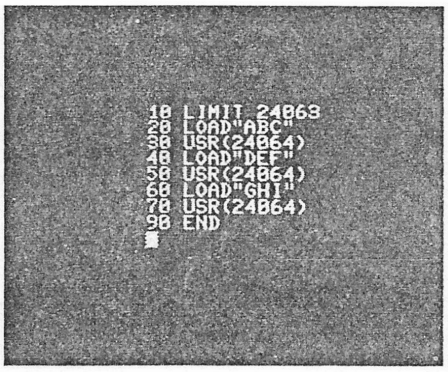
Photo 3

"LIMIT 24063" is placed in order to ensure safety memory area where the training program is executed starting with address 5E00 (24064 in decimal number system). "LIMIT 24063" also limits the greatest memory address used in BASIC to 24063 (address 5DEF in hexadecimal number system). Photo 4 shows those loaded after tentatively maximizing BASIC usable area by means of "LIMIT MAX." OVERLAY displayed here means that the addresses to be loaded are in BASIC area. Therefore error occur.
It should be noticed from Photo 3 that even if machine language program is loaded, BASIC text is never cleared. This is a distinguishable point as compared to loading of BASIC text.
Taking this advantage, another method may be brought forth so as to consecutively link a BASIC program to a few machine language programs. As shown in Photos 5 and 6, the cassette tape can be automatically wound and stopped by pressing PLAY button only once at the initial load. It is noted that the loading addresses of the three machine language programs are same in the former, and not same in the latter.
It may become thus possible that more complicated programming of BASIC and machine language program could provide linkage of BASIC conversational language and calculating functions with Machine Language high-speed processings and input/output functions.



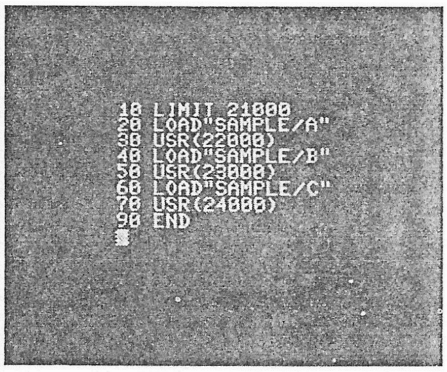
```
LIMIT MAX
READY
LOAD
↓ PLAY
FOUND TRAINING PROGRAM
OVERLAY ERROR
READY
```

Photo 4



```
10 LIMIT 24063
20 LOAD"ABC"
30 USR(24064)
40 LOAD"DEF"
50 USR(24064)
60 LOAD"GHI"
70 USR(24064)
90 END
```

Photo 5



```
10 LIMIT 21000
20 LOAD"SAMPLE/A"
30 USR(22000)
40 LOAD"SAMPLE/B"
50 USR(23000)
60 LOAD"SAMPLE/C"
70 USR(24000)
90 END
```

Photo 6

Monitor command

The MZ-80 series has the following monitor commands; LOAD command to load saved object program, GOTO$ command to allow jumping to the initial address of program execution, SG command to emit sound as keying-in and SS command to stop sound.
The cassette file maked by Machine Language system program can be loaded and executed by the monitor without Machine Language.

＊LOAD [CR]

Instructs to load saved object program. The address to be loaded is related to an area predetermined in the file. After complete loading of system programs such as BASIC, Machine Language, the system transfers the control to the loaded program. It is, however, necessary to transfer the control as to machine language program by GOTO command mentioned below.

＊GOTO $ HHHH [CR]

Instructs to transfer the system control to hexadecimal address HHHH. Following the GOTO$ command, input the address represented in hexadecimal 4-digit numerals with keys.
Since load address of BASIC or Machine Language is 1200 (in dexadecimal number system), to execute "BYE" for BASIC or "!" for Mahcine Language and return the control to BASIC or Machine Language, key in GOTO$ 1200 [CR] . In this case text area of BASIC and free area of Machine Language are cleared.

＊SG [CR]

Instructs to emit a beeptone each time keying-in for inputting.

＊SS [CR]

Instructs, contrary to SS command, to interrupt a beeptone when keying-in.

How to use monitor subroutines

Subroutine (hexadecimal address)	Function	Register storage	Number of stacks
CALL LETNL (0006)	Starts a new line and set the cursor at the head of the line.	Other than AF	8
CALL PRINTS (000C)	Displays one space at cursor position on the CRT screen.	Other than AF	13
CALL PRNT (0012)	Displays character corresponding to ACC data (regarded ASCII code) in the cursor position on the CRT screen.	Other than AF	13
CALL MSG (0015)	Displays messages from the cursor position on the CRT screen. The initial address of the message should be designated by DE register and its end mark must be carriage return (0D in hexadecimal notation). Carriage return is, however, not executed.	All registers	13
CALL GETKY (001B)	Takes ASCII code of one character to ACC using the key-board. Without keying-in, 0 is set in ACC. Chattering by keying-in is, however, not prevented. Echo back is not done either.	Other than AF	9
CALL BRKEY (001E)	Checks whether $\boxed{\text{SHIFT}}$ and $\boxed{\text{BREAK}}$ keys have been pressed. If pressed, Z flag is set, and if not pressed, it is reset.	Other than AF	1
CALL GETL (0003)	Inputs one line using the keyboard (end mark is put by carriage return). ● Input data store address is set to DE register and called. ● The number of input letters and numbers (incl. carriage return) is 80 max. ● In keying-in, echo back is done and cursor operation is accessible. ● By pressing $\boxed{\text{SHIFT}}$ and $\boxed{\text{BREAK}}$ keys, BREAK code and carriage return code are set in the address given by DE register and return to main routine.	All registers	15
CALL MELDY (0030)	Plays music data given by DE register. The end mark of music data is carriage return (0D in hexadecimal notation) or ▩ (C8 in hexadecimal notation). It is, however, noted that if C flag is 0 when returning, the performance has been completed, and if it is 1, $\boxed{\text{BREAK}}$ key has been pressed in course of playing.	Other than AF	7
CALL BELL (003E)	Gives a mid-range tone "la" (approx. 440 Hz).	Other than AF	5

CALL XTEMP (0041)	Changes the tempo. Tempo data (1 thru 7) is set to ACC and called.
	ACC ← 1 the slowest
	ACC ← 4 moderate
	ACC ← 7 the fastest
		All registers	4

Subroutine (hexadecimal address)	Function	Register storage	Number of stacks
CALL MSTA (0044)	Emits continuous sounds of specified frequency dividing ratio. Of the ratio nn' (binary), n' and n are stored in addresses 11A1 and 11A2, respectively, and they are called. The relation of frequency dividing ratio to generated frequency is 2 MHz/nn'.	Only BC and DE	3
CALL MSTP (0047)	Stops emitting sounds.	Other than AF	1
CALL TIMST (0033)	Sets the built-in clock. (The clock starts to function by this call.) The call conditions are; ACC ← 0 (AM), ACC ← 1 (PM), DE ← number of seconds represented in binary notation.	Other than AF	6
CALL TIMRD (003B)	Reads indication of the built-in clock. The conditions when returning are; ACC ← 0 (AM), ACC ← 1 (PM), DE ← number of seconds represented in binary notation.	Other than AF and DE	3

Special display codes and ASCII codes

Key	Display code	ASCII code
CURSOR ⬇	C1	11
CURSOR ⬆	C2	12
CURSOR ➡	C3	13
CURSOR ⬅	C4	14
HOME	C5	15
CLR	C6	16
DEL	C7	60
INST	C8	61
CAP	C9	62
SML	CA	63
BREAK	CB	64
CR	CD	66

Note

For music data, like those in BASIC Manual, ASCII codes shall be assigned in the order of musical interval and tone.

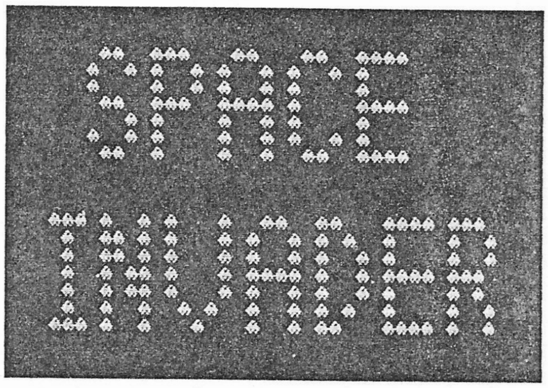
Example C 3 D E F G A B R□C.2 D E F G A B $\boxed{\text{CR}}$

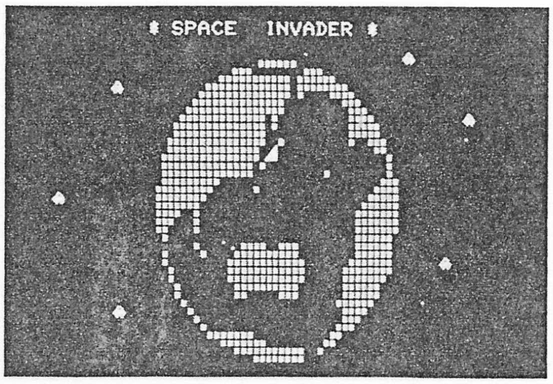
↑ Address given by DE register

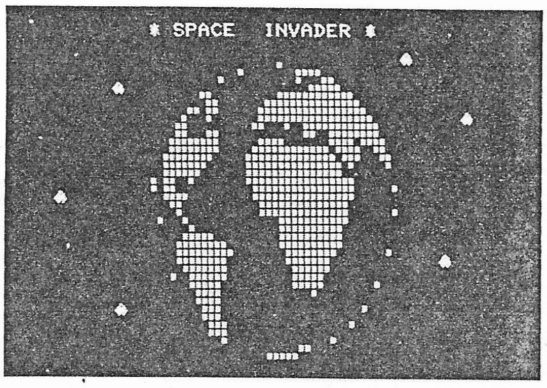
Also tempo data is represented in binary codes of 1 thru 7.

List of sample program

Below is shown a somewhat complicated sample program using Machine Language. This program data having about 4600 bytes are stored beginning at address 2000, and the program is executed from this address. The program is interrupted with SHIFT and BREAK keys, and Machine Language is taken back. However, this procedure is valid only just before repeating of the program.







```
>M 2000 317A
 2000   31 00 20 C3 97 20 3E 16 CD 5A 23 11 00 00 3E 00
 2010   CD 63 23 CD 5D 23 11 74 23 CD 60 23 CD 5D 23 06
 2020   14 21 9C 23 11 79 D0 C5 01 28 00 CD E8 20 CD 02
 2030   21 C1 05 C2 27 20 C3 39 20 CD 69 23 CD BC 21 CD
 2040   72 20 CD E2 21 CD 72 20 CD 00 22 CD 72 20 CD 22
 2050   22 CD 72 20 CD 40 22 CD 72 20 CD 4A 22 CD 72 20
 2060   CD 6C 23 CA 60 12 CD 6F 23 7B FE 40 D2 97 20 C3
 2070   3C 20 CD 02 23 CD 8A 21 CD 27 23 CD 8A 21 CD 2E
 2080   23 CD 8A 21 CD 35 23 CD 8A 21 CD 3C 23 CD 8A 21
 2090   CD 43 23 CD 8A 21 C9 3E 16 CD 5A 23 11 79 D0 21
 20A0   BC 26 01 18 01 CD 02 21 11 09 D2 21 D4 27 01 18
 20B0   01 CD 02 21 21 00 D0 11 EC 28 0E 13 E5 D5 C5 CD
 20C0   23 21 3E C7 CD 6C 22 C1 0D CA DA 20 D1 62 6B 11
 20D0   0A 00 19 54 5D E1 23 C3 BC 20 D1 E1 06 0A CD 7A
 20E0   21 05 C2 DE 20 C3 06 20 E5 D5 C5 01 01 00 2A 72
 20F0   23 71 23 70 CD 66 23 03 78 FE 20 C2 EE 20 C1 D1
 2100   E1 C9 ED A0 79 FE 00 C2 02 21 78 FE 00 C2 02 21
 2110   C9 F5 E5 21 02 E0 7E E6 80 FE 00 C2 16 21 E1 F1
 2120   C3 02 21 E5 D5 C5 01 64 00 2A 72 23 71 23 70 CD
 2130   66 23 03 78 FE 10 C2 29 21 CD 69 23 C1 D1 E1 C9
 2140   47 0E 0A 3E 6B CD AB 21 CD 92 21 CD 5D 21 78 CD
 2150   AB 21 CD 92 21 CD 5D 21 0D C2 43 21 C9 F5 D5 11
 2160   00 02 1B 7A FE 00 C2 62 21 7B FE 00 C2 62 21 D1
 2170   F1 C9 F5 D5 11 00 01 C3 62 21 F5 D5 11 00 A0 C3
 2180   62 21 F5 D5 11 00 03 C3 62 21 F5 D5 11 00 40 C3
 2190   62 21 C5 D5 E5 01 00 0B 2A 72 23 71 23 70 CD 66
 21A0   23 CD 72 21 CD 69 23 E1 D1 C1 C9 F5 E5 21 02 E0
 21B0   7E E6 80 FE 00 C2 B0 21 E1 F1 77 C9 E5 CD 23 21
 21C0   21 97 D0 11 27 00 19 7E FE 00 C2 DD 21 47 3E 2D
```

```
21D0   CD AB 21 CD 72 21 78 CD AB 21 C3 C3 21 CD 40 21
21E0   E1 C9 E5 CD 23 21 21 3C D1 2B 7E FE 00 C2 DD 21
21F0   47 3E 78 CD AB 21 CD 72 21 78 CD AB 21 C3 E9 21
2200   E5 CD 23 21 21 A2 D2 11 29 00 ED 52 7E FE 00 C2
2210   DD 21 47 3E 59 CD AB 21 CD 72 21 78 CD AB 21 C3
2220   07 22 E5 CD 23 21 21 D0 D0 23 7E FE 00 C2 DD 21
2230   47 3E 78 CD AB 21 CD 72 21 78 CD AB 21 C3 29 22
2240   E5 CD 23 21 21 E3 D1 C3 29 22 E5 CD 23 21 21 00
2250   D3 11 27 00 ED 52 7E FE 00 C2 DD 21 47 3E 2D CD
2260   AB 21 CD 72 21 78 CD AB 21 C3 51 22 C5 F5 06 00
2270   1A E6 F0 FE F0 CA 8C 22 FE 00 CA 8F 22 FE 10 CA
2280   B1 22 FE 20 CA CB 22 FE 30 CA E8 22 F1 C1 C9 1A
2290   E6 0F 70 01 28 00 ED 42 4F CD 82 21 46 F1 CD AB
22A0   21 F5 0D CA AD 22 CD 82 21 79 C3 92 22 13 C3 70
22B0   22 1A E6 0F 70 23 4F CD 82 21 46 F1 CD AB 21 F5
22C0   0D CA AD 22 CD 82 21 79 C3 B4 22 1A E6 0F 70 01
22D0   28 00 09 4F CD 82 21 46 F1 CD AB 21 F5 0D CA AD
22E0   22 CD 82 21 79 C3 CE 22 1A E6 0F 70 2B 4F CD 82
22F0   21 46 F1 CD AB 21 F5 0D CA AD 22 CD 82 21 79 C3
2300   EB 22 E5 11 AA 29 21 83 D0 06 14 C5 E5 01 14 00
2310   EB CD 11 21 EB E1 D5 11 28 00 19 D1 C1 05 CA 25
2320   23 C5 C3 0C 23 E1 C9 E5 11 3A 2B C3 06 23 E5 11
2330   CA 2C C3 06 23 E5 11 5A 2E C3 06 23 E5 11 EA 2F
2340   C3 06 23 E5 06 14 21 9C 23 11 79 D0 C5 01 28 00
2350   CD 11 21 C1 05 C2 4C 23 E1 C9 C3 12 00 C3 06 00
2360   C3 15 00 C3 33 00 C3 44 00 C3 47 00 C3 1E 00 C3
2370   3B 00 A1 11 20 20 20 20 20 20 20 20 20 20 20 2A
2380   20 53 50 41 43 45 20 20 49 4E 56 41 44 45 52 20
2390   2A 20 20 20 20 20 20 20 20 20 20 0D 00 00 00 00
23A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 F8 FC
23B0   FC F4 00 00 00 00 00 00 00 00 C7 00 00 00 00 00
23C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
23D0   00 00 00 F8 FE FF FC FC FC F8 FF FD F4 00 00 00
23E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
23F0   00 00 00 C7 00 00 00 00 00 00 FE FF FF FF FF FF
2400   FF FA F3 F3 FF FD 00 00 00 00 00 00 00 00 00 00
2410   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2420   00 FE FF FF FF FF FF FF F5 00 00 00 00 FB FD 00
2430   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2440   00 00 00 00 00 00 00 00 FE FF FF FF FF FF FF FA
2450   00 00 00 00 00 FA FF FD 00 00 00 00 00 00 00 C7
2460   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F8
2470   FF FF FF FF FF FF FF 00 F4 00 00 00 00 FA F3 FF
2480   F4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2490   00 00 00 00 00 00 00 FE FF FF FF FF FF FF F5 4E
24A0   00 00 00 00 00 00 00 F2 F9 00 00 00 00 00 00 00
24B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF
24C0   FF FF FF FF FF FF F2 00 00 00 00 F8 00 00 00 00
24D0   FA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
24E0   00 00 00 00 00 00 F8 FF FF FF FF F1 00 00 F4 00
24F0   00 00 00 00 00 00 00 F8 FF F5 00 00 00 00 00 00
2500   00 00 00 00 00 00 C7 00 00 00 00 00 00 00 FA FF
2510   FF FF F1 00 00 00 00 00 00 00 00 00 00 00 F8 FF
2520   FF F5 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2530   00 00 00 00 00 00 FA FF F1 FA 00 00 00 00 00 00
2540   00 00 00 00 00 00 FF FF FF F5 00 00 00 00 00 00
```

```
2550   00 00 00 00 00 00 00 00 00 00 00 00 00 00 F2 F5
2560   00 F2 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
2570   FF F1 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2580   00 00 00 00 00 00 00 F4 00 F2 F4 00 F8 FF FF FC
2590   FF F5 00 00 00 00 FF FF FF 00 00 00 00 00 00 00
25A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F9
25B0   00 00 00 00 FF FF FF FF FF FF 00 00 00 FE FF FF
25C0   FF 00 00 00 00 C7 00 00 00 00 00 00 00 00 00 00
25D0   00 00 00 00 00 00 00 F2 F4 00 00 00 FB FF FF FF
25E0   FF F7 00 00 00 FF FF FF F1 00 00 00 00 00 00 00
25F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2600   F9 00 00 00 F2 F1 00 00 F3 F1 00 00 F8 FF FF F7
2610   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2620   00 00 00 C7 00 00 00 00 00 F9 00 00 00 00 00 00
2630   00 00 00 00 00 FF F7 00 00 00 00 00 00 00 00 00
2640   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2650   00 00 F9 FC FC FC 00 00 00 00 00 00 FE F3 00 00
2660   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2670   00 00 00 00 00 00 00 00 00 00 00 F2 FB FF FF FC
2680   FC FC 00 F6 F1 00 00 00 00 00 00 00 00 00 00 00
2690   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26A0   00 00 00 00 00 00 F3 F3 F3 F3 00 00 00 00 00 00
26B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26C0   00 00 00 00 00 00 C7 C7 C7 00 00 00 C7 C7 00 00
26D0   00 C7 C7 00 00 C7 C7 C7 C7 00 00 00 00 00 00 00
26E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 C7 00
26F0   00 C7 00 00 00 00 C7 00 C7 00 00 C7 00 C7 00 00
2700   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2710   00 00 00 00 00 00 C7 00 00 C7 00 C7 00 00 C7 00
2720   C7 00 00 00 00 C7 00 00 00 00 00 00 00 00 00 00
2730   00 00 00 00 00 00 00 00 00 00 00 00 00 00 C7 C7
2740   C7 00 00 C7 C7 C7 C7 00 C7 00 00 00 00 C7 C7 C7
2750   C7 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2760   00 00 00 00 C7 00 C7 00 00 00 00 C7 00 00 C7 00
2770   C7 00 00 00 00 C7 00 00 00 00 00 00 00 00 00 00
2780   00 00 00 00 00 00 00 00 00 C7 00 00 C7 00 C7 00
2790   00 00 00 C7 00 00 C7 00 C7 00 00 C7 00 C7 00 00
27A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
27B0   00 00 C7 C7 00 00 C7 00 00 00 00 00 00 00 00 00
27C0   00 C7 C7 00 00 C7 C7 C7 C7 00 00 00 00 00 00 00
27D0   00 00 00 00 00 00 C7 C7 C7 00 C7 00 00 00 00 C7
27E0   00 00 00 C7 00 00 C7 C7 00 00 00 C7 00 00 00 C7
27F0   00 C7 C7 00 00 C7 C7 00 00 00 00 00 00 00 00 C7
2800   00 00 C7 C7 00 C7 00 C7 00 00 00 C7 00 C7 00 00
2810   C7 00 C7 00 C7 00 00 C7 00 00 00 00 C7 00 00 C7
2820   00 00 00 00 00 00 00 C7 00 00 C7 C7 00 00 00 00
2830   00 00 00 C7 00 C7 00 00 C7 00 C7 00 00 00 00 C7
2840   00 00 00 00 C7 00 00 C7 00 00 00 00 00 00 00 C7
2850   00 00 C7 00 C7 C7 00 C7 00 00 00 C7 00 C7 C7 C7
2860   C7 00 C7 00 00 C7 00 C7 C7 C7 C7 00 C7 C7 C7 00
2870   00 00 00 00 00 00 00 C7 00 00 C7 00 C7 C7 00 C7
2880   00 00 00 00 00 C7 00 00 C7 00 C7 00 00 C7 00 C7
2890   00 00 00 00 C7 00 C7 00 00 00 00 00 00 00 00 C7
28A0   00 00 C7 00 00 C7 00 00 C7 00 C7 00 00 C7 00 00
28B0   C7 00 C7 00 C7 00 00 C7 00 00 00 00 C7 00 00 C7
28C0   00 00 00 00 00 00 C7 C7 C7 00 C7 00 00 C7 00 00
```

```
28D0   00 C7 00 00 00 C7 00 00 C7 00 C7 C7 00 00 00 C7
28E0   C7 C7 C7 00 C7 00 00 C7 00 00 00 00 17 23 F0 00
28F0   00 00 00 00 00 00 17 23 F0 00 00 00 00 00 00 00
2900   24 17 F0 00 00 00 00 00 00 00 13 24 F0 00 00 00
2910   00 00 00 00 1F 29 11 3E 04 F0 00 00 00 00 23 33
2920   23 15 F0 00 00 00 00 00 22 13 23 31 21 F0 00 00
2930   00 00 1F 11 11 1F 11 2F 3E F0 00 00 2D 12 F0 00
2940   00 00 00 00 00 00 17 24 F0 00 00 00 00 00 00 00
2950   2F F0 F0 00 00 00 00 00 00 00 2F 11 F0 00 00 00
2960   00 00 00 00 2D 1B F0 00 00 00 00 00 00 00 27 34
2970   F0 00 00 00 00 00 00 00 2F 06 12 F0 00 00 00 00
2980   00 00 22 3E 2F 1F F0 00 00 00 00 00 22 14 27 31
2990   F0 00 00 00 00 00 1F 2D 11 F0 00 00 00 00 00 00
29A0   22 12 29 19 22 F0 00 00 00 00 00 00 00 00 00 00
29B0   00 00 F8 00 00 00 00 00 00 00 00 00 00 00 00 00
29C0   00 00 00 00 F8 FC FF FF FF FF FF FD 00 00 00 00
29D0   00 00 00 00 00 00 FE FA FF FF FF FF FF FF FF FF
29E0   FA F1 00 00 00 00 00 00 00 FE FF FE FF FF FF FF
29F0   FF FF FF F5 00 00 F1 00 00 00 00 00 FE FF FF FF
2A00   FF FF FF FF FF FF FA 00 00 00 00 F1 00 00 00 F8
2A10   FF FF FF FF FF FF FF FF FF FF 00 F4 00 00 00 00
2A20   F4 00 00 F6 FF FF FF FF FF FF FF FF FF F5 4E 00
2A30   00 00 00 00 00 00 00 FD FF FF FF FF FF FF FF FF
2A40   FF F2 00 00 00 00 00 00 F2 00 F2 FB FF FF FF FF
2A50   FF FF F1 00 00 00 00 00 00 00 00 00 00 F4 F8 F4
2A60   FA FF FF FF FF F1 00 00 00 00 00 00 00 00 00 00
2A70   00 00 FA F1 00 FF FF F1 FA 00 00 00 00 00 00 00
2A80   00 00 00 00 00 F5 F2 00 00 F2 F5 00 F2 00 00 00
2A90   00 00 00 00 00 00 00 00 FA F1 00 F4 00 00 00 00
2AA0   F2 F4 00 F8 FF FD FE F5 00 00 00 00 FA 00 00 F8
2AB0   00 00 00 00 00 00 00 FF FF FF FF FF 00 00 00 F8
2AC0   F7 00 00 00 F4 00 00 00 00 00 00 FB FF FF FF FF
2AD0   00 00 00 FA F1 00 00 00 F8 00 00 00 00 00 00 F2
2AE0   F1 00 F2 F1 00 00 00 F3 00 00 00 00 00 F8 00 00
2AF0   00 00 00 00 00 00 00 00 00 00 F2 00 00 00 00 00
2B00   00 00 F8 00 00 00 00 00 00 00 00 00 00 F2 00 00
2B10   00 00 00 00 00 00 00 F2 00 FB FF FF FF FC FC 00
2B20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 F2 F3
2B30   F3 F3 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2B40   00 00 00 F4 F8 00 00 00 00 00 00 00 00 00 00 00
2B50   00 00 00 00 FC F4 00 00 00 FC FD FC 00 00 00 00
2B60   00 00 00 00 00 00 FC F8 FF FF FF FF FF FF FF FF
2B70   FF FC 00 00 00 00 00 00 00 FE F7 FA FF FF FF FF
2B80   FF FF FF FF FF FF FC 00 00 00 00 00 FE F1 FF FF
2B90   FF FF FF FF FF FF FF FF FF FF FA 00 00 00 00 F8
2BA0   00 FC FF FF FF FF FF FF FF FF FF FF FF FF 00 F4
2BB0   00 00 00 FE FF FF FF FF FF FF FF FF FF FF FF FF
2BC0   FF F5 4E 00 00 00 00 FF FF FF FF FF FF FF FF FF
2BD0   FF FF FF FF FF F2 00 00 F2 00 F8 F5 FA FF FF FF
2BE0   FA FF FF FF FF FF FF FF F5 00 00 00 00 F4 FA FB
2BF0   FF FF FF FF FF FF FF FF FF F7 00 00 00 00 00 00
2C00   00 00 00 00 FA FF FF FB FF FF FF FF F7 00 00 00
2C10   00 00 00 00 00 F1 F2 FF FF FB FF FD FA FF F7 00
2C20   F5 00 00 00 00 00 00 00 00 00 00 FF FF FD FB F7
2C30   00 FB 00 00 F1 00 00 00 00 00 00 00 F8 00 00 FB
2C40   FF FF FC F4 00 00 00 00 F9 00 00 F8 F4 F8 00 00
```

```
2C50   00 00 00 F2 FF FF F7 00 00 00 00 00 00 00 00 FF
2C60   FF FF 00 00 F1 00 00 00 FA FF F1 00 00 00 00 00
2C70   00 00 00 F2 F1 F2 00 F4 00 00 00 00 00 FB F1 00
2C80   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2C90   00 00 F1 00 00 00 00 00 00 00 00 00 F8 F2 00 00
2CA0   00 00 00 00 00 00 00 00 F8 00 00 00 00 00 F8 F2
2CB0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F3
2CC0   F3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2CD0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2CE0   00 00 00 00 FC FD FC FC FE FD FC FC F4 00 00 00
2CF0   00 00 00 00 00 00 00 FA F7 F1 FF FF FF FF FF FF
2D00   FF FD 00 00 00 00 00 00 00 00 00 F2 F1 FC FF FF
2D10   FF FF FF FF FF FF FD 00 00 00 00 00 F2 F8 FE FF
2D20   FF FF FF FF FF FF FF FF FF FF FF F5 00 00 00 00
2D30   00 FF FF FF FF F5 FA FF FA FF FF FF FF FF FF F5
2D40   00 00 00 00 F2 F1 00 F1 FA F7 FB FF FB FF FF FF
2D50   FF FF 00 00 F8 00 00 F1 F8 FF FF FC FC FC F6 FF
2D60   FD FA FF FF F7 F5 00 00 00 00 00 00 FF FF FF FF
2D70   FF FF FF F6 F7 00 FB FF 00 F1 00 00 00 F4 F8 00
2D80   FF FF FF FF FF FF FF FD F4 00 00 F1 00 F9 00 00
2D90   00 F4 F8 00 FA FF FF FF FF FF FF FF F5 00 00 00
2DA0   00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF
2DB0   00 00 00 00 00 00 00 00 00 F1 00 F4 00 00 FA FF
2DC0   FF FF FF F7 00 00 00 00 00 00 00 00 FC 00 00 F8
2DD0   00 00 FA FF FF FF FF F5 00 00 00 00 00 00 00 00
2DE0   F7 00 00 00 F4 00 00 FF FF FF F7 00 00 00 00 00
2DF0   00 00 00 00 00 00 00 00 F8 00 00 FA FF FF F1 00
2E00   00 00 00 00 00 00 00 F4 00 00 00 00 00 F8 00 F2
2E10   FF F7 00 00 00 00 00 00 00 00 F4 00 00 00 00 00
2E20   00 00 00 00 00 F1 00 00 00 00 00 00 F8 00 00 00
2E30   00 00 00 00 00 00 00 F2 00 F8 F4 00 00 F8 F4 00
2E40   00 00 00 00 00 00 00 00 00 00 00 00 00 00 F3 F3
2E50   F3 F1 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2E60   00 00 00 00 F4 00 00 00 00 00 00 00 00 00 00 00
2E70   00 00 00 F8 00 F1 00 00 00 FA F3 F9 F4 00 00 00
2E80   00 00 00 00 00 00 FC 00 00 00 00 00 FC FC FE FF
2E90   FF FC 00 00 00 00 00 00 00 FC FA F5 00 00 F8 FF
2EA0   FF FF FF FF FF FF FD 00 00 00 00 00 FE 00 F2 00
2EB0   00 00 FE F3 FB F3 FF FF FF FF FF FD 00 00 00 F8
2EC0   FF FC FE F5 00 00 F8 FC F4 00 F3 00 FF FF FB FF
2ED0   F4 00 00 FE FF FF FF F1 00 F8 FF FF FF FF FF FF
2EE0   F6 FF F5 F2 FD 00 00 FF FF FF F1 00 00 FA FF FF
2EF0   FF FF FF FF FF F2 00 00 F2 00 F5 FF 00 F1 00 00
2F00   00 F2 FF FF FF FF FF FF FF FF 00 00 00 F4 00 FB
2F10   F7 00 00 00 00 00 FB FF FF FF FF FF FF F7 00 00
2F20   00 00 F8 00 F9 00 00 00 00 00 F2 F3 FF FF FF FF
2F30   FF F5 00 00 00 F1 00 00 F8 FD FC 00 00 00 00 00
2F40   FF FF FF FF FF 00 00 00 00 00 00 00 FF FF FF FF
2F50   F4 00 00 00 FB FF FF FF F7 00 00 00 F8 00 00 00
2F60   FB FF FF FF 00 00 00 00 F2 FF FF FF 00 00 00 00
2F70   00 00 00 F2 00 FF FF F7 00 00 00 00 00 FF FF F5
2F80   00 00 00 F8 00 00 00 00 00 FA FF F5 00 00 00 00
2F90   00 00 F2 00 00 00 00 00 00 00 00 00 00 00 FF F5
2FA0   00 00 00 00 00 00 00 00 00 00 F2 00 00 00 00 00
2FB0   00 00 FA FF 00 00 00 00 00 00 00 00 00 F2 00 00
2FC0   00 00 00 00 00 00 00 F2 00 00 00 00 00 F8 F4 00
```

```
2FD0   F1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F3
2FE0   F3 F1 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2FF0   00 00 00 F8 00 F8 00 00 00 00 00 00 00 00 00 00
3000   00 00 00 00 F2 00 00 00 00 00 00 F8 00 00 00 00
3010   00 00 00 00 00 00 FE FF F4 00 00 F8 FC F4 00 00
3020   00 00 00 00 00 00 00 00 00 F2 F7 F7 00 F8 FF FF
3030   FF FF FD FE F5 00 F1 00 00 00 00 00 00 F4 00 00
3040   00 F3 00 FF FF FF FF FF F1 00 00 00 00 00 00 00
3050   4E 00 00 00 00 00 00 F2 FF F7 FB F1 00 00 00 00
3060   00 00 00 F2 00 00 00 00 00 00 00 00 FF FD F4 00
3070   00 00 00 00 F8 00 00 F1 00 00 00 00 00 00 00 00
3080   F7 FF 00 00 00 00 00 00 F2 00 F8 00 00 00 00 00
3090   00 00 00 00 00 F2 F4 00 00 00 00 00 FA F5 F8 00
30A0   00 00 00 00 00 00 00 00 00 00 FA FC F4 00 00 00
30B0   00 F5 00 00 00 00 00 00 00 00 00 00 00 F8 FF FF
30C0   FF FF F4 00 00 F5 00 00 00 00 00 00 00 00 00 00
30D0   FA FF FF FF FF FF FC 00 00 F5 00 F1 00 00 00 00
30E0   00 00 00 00 F2 FB FF FF FF FF F1 00 00 00 00 FB
30F0   F1 00 00 00 00 00 00 00 00 00 FB FF FF FF 00 00
3100   00 00 00 00 F5 00 00 00 00 00 00 00 00 00 F2 FF
3110   FF F5 00 00 F1 00 00 00 00 00 00 00 00 00 00 00
3120   00 00 00 FB FF F5 00 00 00 00 00 00 00 F8 00 00
3130   00 00 00 00 00 00 00 F2 FF F1 F2 00 00 00 00 00
3140   00 00 00 F4 00 00 00 00 00 00 00 00 F2 00 00 00
3150   00 00 00 00 00 00 00 00 00 F4 00 00 00 FC 00 F4
3160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 F2 F3
3170   F3 F3 00 00 00 00 00 00 00 00 28
```

Explanation of Z80 instruction set

Z80 CPU flags

The flag register (F and F') are provided so that the programmer be able to check CPU status at any time.
The bit arrangement of each flag is:

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

C:	Carry flag	Z:	Zero flag
N:	Add/subtract flag	S:	Sign flag
P/V:	Parity/overflow flag	X:	Unused
H:	Half-carry flag		

These flags are set or reset by CPU operations.
C, P/V, Z, and S can be verified by the programmer using instructions such as conditional jump, call, etc., while H and N employed in BCD arithmetic operation are not able to be directly checked.

Carry flag C

Setting and resetting of the carry flag depends on arithmetic operations to be executed.
The carry flag is set when carry or borrow takes place in ADD instruction or SUB instruction, respectively. Without any carry or borrow, the carry flag is reset.
When the conditions for decimal adjustment are met, DAA instruction sets the carry flag.
In PLA, RRA, RL, and RR instructions, the carry flag is included as a bit in the link.
In RLCA, RLC, and SLA instructions, a contents of bit 7 of register and memory location are shifted to the carry flag and remains there.
In RRCA, RRC, SRA, and SRL instructions, 0-bit contents of any register or memory location are shifted to the carry flag.
The carry flag is reset by AND, OR, or XOR command.
It is also set by SCF instruction and reversed by CCF instruction.

Add/subtract flag N

This flag is used for execution of DAA instruction.
It is set to "0" by ADD instruction and to "1" by SUB instruction.

Parity/overflow flag P/V

This flag is set due to an overflow caused when result of an arithmetic operation to be stored in the accumulator is less than −128 or more than +127.
Described here are conditions for this flag to be set or reset.

1) When adding numbers of different signs: Reset.
2) When addition of numbers of the same sign results in a number of the opposite sign: Set.

```
Example        Decimal           Binary
               + 1 2 0   =   0 1 1 1   1 0 0 0
          +)   + 1 0 5   =   0 1 1 0   1 0 0 1
        ------------------------------------------
               −   9 5   =   1 1 1 0   0 0 0 1     (Overflow)
```

3) When subtracting numbers of the same sign: Reset.

4) When subtracting numbers of different signs: Set or reset depending on how large each number is.
In the following case, the flag is reset.

```
Example        Decimal              Binary
               +127    =    0111   1111
      -)       − 64    =    1100   0000
      ------------------------------------------------
               − 65    =    1011   1111      (Overflow)
```

This flag is also used in checking the parity (the number of "1" bits in a byte) calculated in logical operation or rotate command. When the sum of "1" bits is odd, P = 0 (odd parity), and if it is even, P = 1 (even parity).
While executing search instructions (CPI, CPD, etc.) or block transfer instructions (LDI, LDD, etc.), this P/V flag monitors the status of the byte counter (BC). When the byte counter is not "0", the flag is "1", and when "0", it is also "0".
In executing LD A, I and LD A, R instructions, the contents of IFF2 (interrupt enable flip-flop 2) are transferred to this P/V flag, by which the contents of IFF2 can be saved or tested. When reading bytes one by one from the I/O device by IN r, (C) instruction, this P/V flag is set or reset according to data parity.

Half-carry flag H

The half-carry flag is set or reset depending on the carry or borrow status between bits 3 and 4 in an 8-bit arithmetic operation.
This flag is utilized for correction of results of packed BCD addition or subtraction by means of DAA command. If carry or borrow exists, the flag is set ot "1", and if not, it is reset to "0".

Zero flag Z

The zero flag is set or reset depending on whether result of an execution by a instruction is 0 or not. When the contents of the accumulator is 0 in a 8-bit arithmetic or logical operation, the flag is set to "1". Otherwise, it is reset to "0". In case value of the accumulator and that of the memory location specified by register pair HL are equal to each other, the zero flag is set to "1" in search instructions..
In bit test instruction, the complement of a specified bit is placed in this zero flag.
In input/output instruction (INI, IND, OUTI and OUTD), when the number obtained by subtracting 1 from byte counter B is 0 (B − 1 = 0), the zero flag is set, otherwise it is reset. Also, in IN r, (C) instruction, this flag is set when input data is 0.

Sign flag S

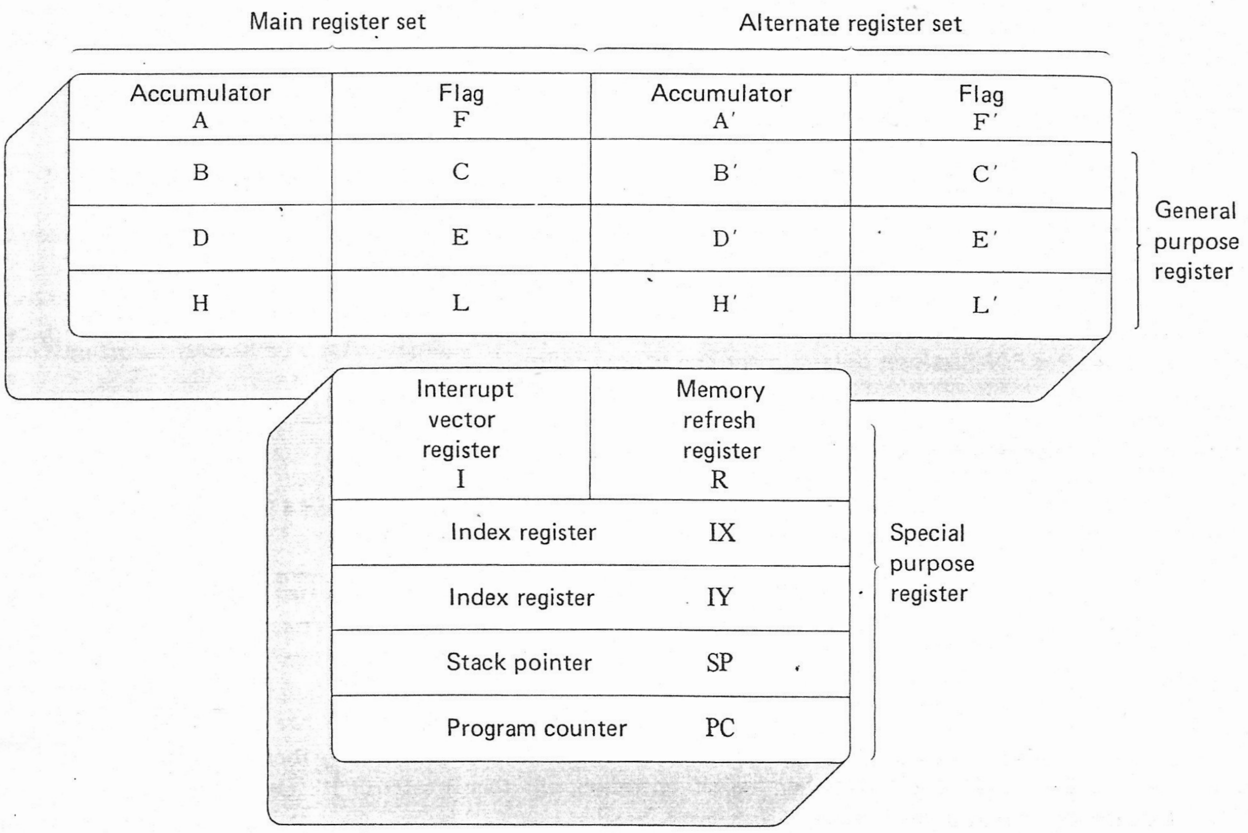
The sign flag storeing the state of the most significant bit (bit 7) of the accumulator is employed in arithmetic operations. For operations with signs, binary two's complement notation is used, when the bit 7 is "0", it is considered as positive, and when "1", negative.
Both positive and negative numbers are given in 7 bits (0 ~127 or −1 ~ −128).
In reading data by input instruction (IN r, (C)), positiveness and negativeness of the data are set to the sign flag with "0" and "1", respectively.

Symbols concerning flags used in the Z80 instruction set

↕	:	Affected according to results.
●	:	No change
0	:	To be reset
1	:	To be set
X	:	To be destroyed
V	:	To be set when overflowing, otherwise to be reset.
P	:	To be set when parity is odd, otherwise to be reset.

Architecture of internal registers

The Z80 CPU internal registers are composed of 208-bit read/write memories. The architecture is as shown below.



The CPU register architecture consists of general purpose registers and special purpose ones. The former has two sets of registers; main and alternate. The contents of each set are interchangeable by swap instruction. Each of the two sets is composed of an 8-bit accumulator, 8-bit flag register and 6 general-purpose registers (8-bit each). The general purpose registers can be also used as 16-bit registers being paired (BC, DE and HL).
The interrupt vector register I (8 bits) of special purpose register group gives the upper 8 bits of interrupt service routine indirect address when an interrupt occurs, and the lower 8 bits thereof are given from the interrupt device. The memory refresh register R (7 bits) automatically generates an address for memory refresh when using a dynamic RAM as an external memory.

Symbols concerning registers used in the Z80 instruction set

r, r′	: Any one of the CPU internal registers A, B, C, D, E, H and L
dd, pp, qq, rr, ss	: Paired CPU internal registers
ii	: Any one of the two index registers IX and IY
R	: Refresh counter
d	: 8-bit displacement used when locating memory with index registers.
dd	: 16-bit memory location
e	: Complement of 2 (−126 to 129) with signs in relative address mode
n	: 8-bit data (0 to 255)
nn	: 16-bit memory location (0 to 65535)

8-bit load group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
LD r, r'	r←r'	●	●	●	●	●	●	01 r r'	1	1	4	r, r' — Register 000 — B 001 — C 010 — D 011 — E 100 — H 101 — L 111 — A
LD r, n	r←n	●	●	●	●	●	●	00 r 110 ← n →	2	2	7	
LD r, (HL)	r←(HL)	●	●	●	●	●	●	01 r 110	1	2	7	
LD r, (IX+d)	r←(IX+d)	●	●	●	●	●	●	11 011 101 01 r 110 ← d →	3	5	19	
LD r, (IY+d)	r←(IY+d)	●	●	●	●	●	●	11 111 101 01 r 110 ← d →	3	5	19	
LD (HL), r	(HL)←r	●	●	●	●	●	●	01 110 r	1	2	7	
LD (IX+d), r	(IX+d)←r	●	●	●	●	●	●	11 011 101 01 110 r ← d →	3	5	19	
LD (IY+d), r	(IY+d)←r	●	●	●	●	●	●	11 111 101 01 110 r ← d →	3	5	19	
LD (HL), n	(HL)←n	●	●	●	●	●	●	00 110 110 ← n →	2	3	10	
LD (IX+d), n	(IX+d)←n	●	●	●	●	●	●	11 011 101 00 110 110 ← d → ← n →	4	5	19	
LD (IY+d), n	(IY+d)←n	●	●	●	●	●	●	11 111 101 00 110 110 ← d → ← n →	4	5	19	
LD A, (BC)	A←(BC)	●	●	●	●	●	●	00 001 010	1	2	7	
LD A, (DE)	A←(DE)	●	●	●	●	●	●	00 011 010	1	2	7	
LD A, (nn)	A←(nn)	●	●	●	●	●	●	00 111 010 ← n → ← n →	3	4	13	
LD (BC), A	(BC)←A	●	●	●	●	●	●	00 000 010	1	2	7	
LD (DE), A	(DE)←A	●	●	●	●	●	●	00 010 010	1	2	7	
LD (nn), A	(nn)←A	●	●	●	●	●	●	00 110 010 ← n → ← n →	3	4	13	
LD A, I	A←I	●	↕	IFF2	↕	0	0	11 101 101 01 010 111	2	2	9	IFF2 : Contents of interrupt enable flip-flop 2
LD A, R	A←R	●	↕	IFF2	↕	0	0	11 101 101 01 011 111	2	2	9	
LD I, A	I←A	●	●	●	●	●	●	11 101 101 01 000 111	2	2	9	
LD R, A	R←A	●	●	●	●	●	●	11 101 101 01 001 111	2	2	9	

16-bit load group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
LD dd,nn	dd←nn	●	●	●	●	●	●	00 dd0 001	3	3	10	dd — Register Pair
								← n →				00 — BC
								← n →				
LD IX,nn	IX←nn	●	●	●	●	●	●	11 011 101	4	4	14	01 — DE
								00 100 001				10 — HL
								← n →				11 — SP
								← n →				
LD IY,nn	IY←nn	●	●	●	●	●	●	11 111 101	4	4	14	
								00 100 001				
								← n →				
								← n →				
LD HL,(nn)	H←(nn+1)	●	●	●	●	●	●	00 101 010	3	5	16	
	L←(nn)							← n →				
								← n →				
LD dd,(nn)	dd_H←(nn+1)	●	●	●	●	●	●	11 101 101	4	6	20	
	dd_L←(nn)							01 dd1 011				
								← n →				
								← n →				
LD IX,(nn)	IX_H←(nn+1)	●	●	●	●	●	●	11 011 101	4	6	20	
	IX_L←(nn)							00 101 010				
								← n →				
								← n →				
LD IY,(nn)	IY_H←(nn+1)	●	●	●	●	●	●	11 111 101	4	6	20	
	IY_L←(nn)							00 101 010				
								← n →				
								← n →				
LD (nn),HL	(nn+1)←H	●	●	●	●	●	●	00 100 010	3	5	16	
	(nn)←L							← n →				
								← n →				
LD (nn),dd	(nn+1)←dd_H	●	●	●	●	●	●	11 101 101	4	6	20	
	(nn)←dd_L							01 dd0 011				
								← n →				
								← n →				
LD (nn),IX	(nn+1)←IX_H	●	●	●	●	●	●	11 011 101	4	6	20	
	(nn)←IX_L							00 100 010				
								← n →				
								← n →				
LD (nn),IY	(nn+1)←IY_H	●	●	●	●	●	●	11 111 101	4	6	20	
	(nn)←IY_L							00 100 010				
								← n →				
								← n →				
LD SP,HL	SP←HL	●	●	●	●	●	●	11 111 001	1	1	6	
LD SP,IX	SP←IX	●	●	●	●	●	●	11 011 101	2	2	10	
								11 111 001				
LD SP,IY	SP←IY	●	●	●	●	●	●	11 111 101	2	2	10	
								11 111 001				

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
PUSH qq	$(SP-2) \leftarrow qq_L$	●	●	●	●	●	●	11 qq0 101	1	3	11	qq Register Pair 00 BC 01 DE 10 HL 11 AF
	$(SP-1) \leftarrow qq_H$											
PUSH IX	$(SP-2) \leftarrow IX_L$	●	●	●	●	●	●	11 011 101	2	4	15	
	$(SP-1) \leftarrow IX_H$							11 100 101				
PUSH IY	$(SP-2) \leftarrow IY_L$	●	●	●	●	●	●	11 111 101	2	4	15	
	$(SP-1) \leftarrow IY_H$							11 100 101				
POP qq	$qq_H \leftarrow (SP+1)$	●	●	●	●	●	●	11 qq0 001	1	3	10	
	$qq_L \leftarrow (SP)$											
POP IX	$IX_H \leftarrow (SP+1)$	●	●	●	●	●	●	11 011 101	2	4	14	
	$IX_L \leftarrow (SP)$							11 100 001				
POP IY	$IY_H \leftarrow (SP+1)$	●	●	●	●	●	●	11 111 101	2	4	14	
	$IY_L \leftarrow (SP)$							11 100 001				

Exchange group, block transfer and search group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
EX DE, HL	DE↔HL	●	●	●	●	●	●	11 101 011	1	1	4	
EX AF, AF′	AF↔AF′	●	●	●	●	●	●	00 001 000	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix} \leftrightarrow \begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	●	●	●	●	●	●	11 011 001	1	1	4	Exchanges the contents of a pair of registers and those of a pair of alternate registers.
EX (SP), HL	H↔(SP+1) L↔(SP)	●	●	●	●	●	●	11 100 011	1	5	19	
EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	●	●	●	●	●	●	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	●	●	●	●	●	●	11 111 101 11 100 011	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	●	●	① ↕	●	0	0	11 101 101 10 100 000	2	4	16	
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Repeat until BC=0	●	●	0	●	0	0	11 101 101 10 110 000	2 2	5 4	21 16	When BC≠0 When BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	●	●	① ↕	●	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 Repeat until BC=0	●	●	0	●	0	0	11 101 101 10 111 000	2 2	5 4	21 16	When BC≠0 When BC=0
CPI	A−(HL) HL←HL+1 BC←BC−1	●	② ↕	① ↕	↕	1	↕	11 101 101 10 100 001	2	4	16	

(Note) ① denotes that when BC − 1 = 0, the P/V flag is 0, and in other cases, it is 1.
② denotes that when A = (HL), the Z flag is 1, and in other cases, it is 0.

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
CPIR	A−(HL) HL←HL+1 BC←BC−1 Repeat until A=(HL) or BC=0	●	② ↕	① ↕	↕	1	↕	11 101 101 10 110 001	2 2	5 4	21 16	When BC≠0 and A≠(HL) When BC=0 or A=(HL)
CPD	A−(HL) HL←HL−1 BC←BC−1	●	② ↕	① ↕	↕	1	↕	11 101 101 10 101 001	2	4	16	
CPDR	A−(HL) HL←HL−1 BC←BC−1 Repeat until A=(HL) or BC=0	●	② ↕	① ↕	↕	1	↕	11 101 101 10 111 001	2 2	5 4	21 16	When BC≠0 and A≠(HL) When BC=0 or A=(HL)

(Note) ① denotes that when BC − 1 = 0, the P/V flag is 0, and in other cases, it is 1.
② denotes that when A = (HL), the Z flag is 1, and in other cases, it is 0.

Correction flag and CPU control group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
DAA	Converts Acc. content into packed BCD following add or subtract with packed BCD operands.	↕	↕	P	↕	●	↕	00 100 111	1	1	4	Decimal ajust accumutator.
CPL	A←$\bar{A}$	●	●	●	●	1	1	00 101 111	1	1	4	
NEG	A←$\bar{A}$+1	↕	↕	V	↕	1	↕	11 101 101 01 000 100	2	2	8	
CCF	CY←$\overline{CY}$	↕	●	●	●	0	X	00 111 111	1	1	4	Complement carry flag
SCF	CY←1	1	●	●	●	0	0	00 110 111	1	1	4	Set carry flag.
NOP	Nothing is executed, but PC←PC+1	●	●	●	●	●	●	00 000 000	1	1	4	
HALT	CPU halted	●	●	●	●	●	●	01 110 110	1	1	4	
DI	IFF←0	●	●	●	●	●	●	11 110 011	1	1	4	
EI	IFF←1	●	●	●	●	●	●	11 111 011	1	1	4	
IM 0	Set interrupt mode 0	●	●	●	●	●	●	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	●	●	●	●	●	●	11 101 101 01 010 110	2	2	8	
IM 2	Set interrupt mode 2	●	●	●	●	●	●	11 101 101 01 011 110	2	2	8	

8-bit arithmetic and logic group

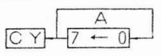
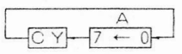
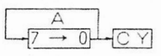
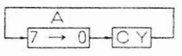
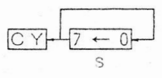
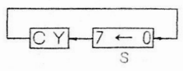
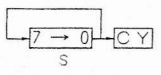
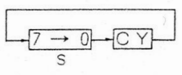
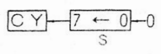
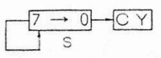
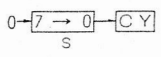
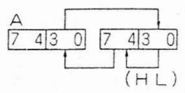
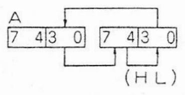
Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
ADD A,r	A←A+r	↕	↕	V	↕	0	↕	10 $\boxed{000}$ r	1	1	4	
ADD A,n	A←A+n	↕	↕	V	↕	0	↕	11 $\boxed{000}$ 110	2	2	7	
								← n →				
ADD A,(HL)	A←A+(HL)	↕	↕	V	↕	0	↕	10 $\boxed{000}$ 110	1	2	7	
ADD A,(IX+d)	A←A+(IX+d)	↕	↕	V	↕	0	↕	11 011 101	3	5	19	
								10 $\boxed{000}$ 110				
								← d →				
ADD A,(IY+d)	A←A+(IY+d)	↕	↕	V	↕	0	↕	11 111 101	3	5	19	
								10 $\boxed{000}$ 110				
								← d →				
ADC A,s	A←A+s+CY	↕	↕	V	↕	0	↕	$\boxed{001}$				s indicates any one of r, n, (HL), (IX+d) and (IY+d), like ADD instruction. Set framed bits in place of $\boxed{000}$ of ADD instruction.
SUB s	A←A−s	↕	↕	V	↕	1	↕	$\boxed{010}$				
SBC A,s	A←A−s−CY	↕	↕	V	↕	1	↕	$\boxed{011}$				
AND s A	A←A∧s	0	↕	P	↕	0	1	$\boxed{100}$				
OR s	A←A∨s	0	↕	P	↕	0	0	$\boxed{110}$				
XOR s	A←A∀s	0	↕	P	↕	0	0	$\boxed{101}$				
CP s	A−s	↕	↕	V	↕	1	↕	$\boxed{111}$				
INC r	r←r+1	●	↕	V	↕	0	↕	00 r $\boxed{100}$	1	1	4	
INC (HL)	(HL)←(HL)+1	●	↕	V	↕	0	↕	00 110 $\boxed{100}$	1	3	11	
INC (IX+d)	(IX+d)← (IX+d)+1	●	↕	V	↕	0	↕	11 011 101	3	6	23	
								00 110 $\boxed{100}$				
								← d →				
INC (IY+d)	(IY+d)← (IY+d)+1	●	↕	V	↕	0	↕	11 111 101	3	6	23	
								00 110 $\boxed{100}$				
								← d →				
DEC m	m←m−1	●	↕	V	↕	1	↕	$\boxed{101}$				m indicates any one of r, (HL), (IX+D) and (IY+d), like INC instruction.
												Operation code is the same as INC instruction changed from $\boxed{100}$ to $\boxed{101}$.

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

16-bit arithmetic operation group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments	
		C	Z	P/V	S	N	H	76 543 210					
ADD HL,ss	HL←HL+ss	↕	●	●	●	0	X	00 ss1 001	1	3	11	ss	Register
ADC HL,ss	HL←HL+ss+CY	↕	↕	V	↕	0	X	11 101 101	2	4	15	00	BC
								01 ss1 010				01	DE
SBC HL,ss	HL←HL−ss−CY	↕	↕	V	↕	1	X	11 101 101	2	4	15	10	HL
								01 ss0 010				11	SP
ADD IX,pp	IX←IX+pp	↕	●	●	●	0	X	11 011 101	2	4	15	pp	Register
								00 pp1 001				00	BC
												01	DE
												10	IX
												11	SP
ADD IY,rr	IY←IY+rr	↕	●	●	●	0	X	11 111 101	2	4	15	rr	Register
								00 rr1 001				00	BC
												01	DE
												10	IY
												11	SP
INC ss	ss←ss+1	●	●	●	●	●	●	00 ss0 011	1	1	6		
INC IX	IX←IX+1	●	●	●	●	●	●	11 011 101	2	2	10		
								00 100 011					
INC IY	IY←IY+I	●	●	●	●	●	●	11 111 101	2	2	10		
								00 100 011					
DEC ss	ss←ss−1	●	●	●	●	●	●	00 ss1 011	1	1	6		
DEC IX	IX←IX−1	●	●	●	●	●	●	11 011 101	2	2	10		
								00 101 011					
DEC IY	IY←IY−1	●	●	●	●	●	●	11 111 101	2	2	10		
								00 101 011					

Rotate and shift group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
RLC A		↕	●	●	●	0	0	00 000 111	1	1	4	Rotate the contents of accumulator to the left.
RL A		↕	●	●	●	0	0	00 010 111	1	1	4	
RRC A		↕	●	●	●	0	0	00 001 111	1	1	4	Rotate the contents of accumulator to the right.
RR A		↕	●	●	●	0	0	00 011 111	1	1	4	
RLC r		↕	↕	P	↕	0	0	11 001 011	2	2	8	Rotate the contents of resister r to the left.
								00 $\boxed{000}$ r				
RLC (HL)		↕	↕	P	↕	0	0	11 001 011	2	4	15	
								00 $\boxed{000}$ 110				
RLC (IX+d)		↕	↕	P	↕	0	0	11 011 101	4	6	23	
								11 001 011				
								← d →				
								00 $\boxed{000}$ 110				
RLC (IY+d)		↕	↕	P	↕	0	0	11 111 101	4	6	23	
								11 001 011				
								← d →				
								00 $\boxed{000}$ 110				
RL s		↕	↕	P	↕	0	0	$\boxed{010}$				r, (HL), (IX+d) or (IY+d) is employed as operand s.
RRC s		↕	↕	P	↕	0	0	$\boxed{001}$				
RR s		↕	↕	P	↕	0	0	$\boxed{011}$				
SLA s		↕	↕	P	↕	0	0	$\boxed{100}$				
SRA s		↕	↕	P	↕	0	0	$\boxed{101}$				
SRL s		↕	↕	P	↕	0	0	$\boxed{111}$				
RLD		●	↕	P	↕	0	0	11 101 101	2	5	18	
								01 101 111				
RRD		●	↕	P	↕	0	0	11 101 101	2	5	18	
								01 100 111				

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

Bit set, reset and test group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
BIT b,r	$Z \leftarrow \overline{r}_b$	●	↕	X	X	0	1	11 001 011 01 b r	2	2	8	
BIT b,(HL)	$Z \leftarrow \overline{(HL)}_b$	●	↕	X	X	0	1	11 001 011 01 b 110	2	3	12	
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	●	↕	X	X	0	1	11 011 101 11 001 011 ← d → 01 b 110	4	5	20	
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	●	↕	X	X	0	1	11 111 101 11 001 011 ← d → 01 b 110	4	5	20	
SET b,r	$r_b \leftarrow 1$	●	●	●	●	●	●	11 001 011 $\boxed{11}$ b r	2	2	8	
SET b,(HL)	$(HL)_b \leftarrow 1$	●	●	●	●	●	●	11 001 011 $\boxed{11}$ b 110	2	4	15	
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	●	●	●	●	●	●	11 011 101 11 001 011 ← d → $\boxed{11}$ b 110	4	6	23	
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	●	●	●	●	●	●	11 111 101 11 001 011 ← d → $\boxed{11}$ b 110	4	6	23	
RES b,s	$s_b \leftarrow 0$ $s \equiv r, (HL), (IX+d), (IY+d)$							$\boxed{10}$				Reset bit b of operand s.

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

b	Bit tested
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Jump group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
JP nn	PC←nn	●	●	●	●	●	●	11 000 011 ← n → ← n →	3	3	10	
JP cc,nn	If condition cc is true, PC←nn If condition cc is false, then next command	●	●	●	●	●	●	11 cc 010 ← n → ← n →	3	3	10	
JR e	PC←PC+e	●	●	●	●	●	●	00 011 000 ← e-2 →	2	3	12	
JR C,e	If C=0, then next command	●	●	●	●	●	●	00 111 000 ← e-2 →	2	2	7	
	If C=1, PC←PC+e								2	3	12	
JR NC,e	If C=1, then next command.	●	●	●	●	●	●	00 110 000 ← e-2 →	2	2	7	
	If C=0, PC←PC+e								2	3	12	
JR Z,e	If Z=0, then next command	●	●	●	●	●	●	00 101 000 ← e-2 →	2	2	7	
	If Z=1, PC←PC+e								2	3	12	
JR NZ,e	If Z=1, then next command	●	●	●	●	●	●	00 100 000 ← e-2 →	2	2	7	
	If Z=0, PC←PC+e								2	3	12	
JP (HL)	PC←HL	●	●	●	●	●	●	11 101 001	1	1	4	
JP (IX)	PC←IX	●	●	●	●	●	●	11 011 101 11 101 001	2	2	8	
JP (IY)	PC←IY	●	●	●	●	●	●	11 111 101 11 101 001	2	2	8	
DJNZ,e D	B←B−1 If B=0, then next command	●	●	●	●	●	●	00 010 000 ← e-2 →	2	2	8	If B=0
	If B≠0, PC←PC+e								2	3	13	If B≠0

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

(Note) The range in which displacement e is allowable is −126 to +129. A binary number equivalent to e − 2 must be placed in operation code.

Call and return group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
CALL nn	$(SP-1) \leftarrow PC_H$	●	●	●	●	●	●	11 001 101	3	5	17	
	$(SP-2) \leftarrow PC_L$							← n →				
	$PC \leftarrow nn$							← n →				
CALL cc,nn	If condition cc is true, same as CALL nn. If it is false, then next command.	●	●	●	●	●	●	11 cc 100	3	3	10	If condition cc is false,
								← n →				
								← n →	3	5	17	If condition cc is true,
RET	$PC_L \leftarrow (SP)$	●	●	●	●	●	●	11 001 001	1	3	10	
	$PC_H \leftarrow (SP+1)$											
RET cc	If condition cc is true, same as RET. If it is false, then next command.	●	●	●	●	●	●	11 cc 000	1	1	5	If condition cc is false,
									1	3	11	If condition cc is true,
RETI	Return from interrupt.	●	●	●	●	●	●	11 101 101	2	4	14	
								01 001 101				
RETN	Return from NMI (Non Maskable Interrupt).	●	●	●	●	●	●	11 101 101	2	4	14	
								01 000 101				
RST p	$(SP-1) \leftarrow PC_H$	●	●	●	●	●	●	11 t 111	1	3	11	
	$(SP-2) \leftarrow PC_L$											
	$PC_H \leftarrow 0$											
	$PC_L \leftarrow P$											

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Input and output group

Mnemonic	Operation	Flags						Operation code	No. of bytes	No. of M cycles	No. of T states	Comments
		C	Z	P/V	S	N	H	76 543 210				
IN A,(n)	A←(n)	●	●	●	●	●	●	11 011 011 ← n →	2	3	10	n to A_0~A_7 Acc to A_8~A_{15}
IN r,(C)	r←(C) If r=110, only the flag is affected.	●	↕	P	↕	0	0	11 101 101 01 r 000	2	3	11	C to A_0~A_7 B to A_8~A_{15}
INI	(HL)←(C) B←B−1 HL←HL+1	●	① ↕	X	X	1	X	11 101 101 10 100 010	2	4	15	C to A_0~A_7 B to A_8~A_{15}
INIR	(HL)←(C) B←B−1 HL←HL+1 Repeat until B=0.	●	1	X	X	1	X	11 101 101 10 110 010	2 2	5 (If $B \neq 0$) 4 (If $B = 0$)	20 15	C to A_0~A_7 B to A_8~A_{15}
IND	(HL)←(C) B←B−1 HL←HL−1	●	① ↕	X	X	1	X	11 101 101 10 101 010	2	4	15	C to A_0~A_7 B to A_8~A_{15}
INDR	(HL)←(C) B←B−1 HL←HL−1 Repeat until B=0.	●	1	X	X	1	X	11 101 101 10 111 010	2 2	5 (If $B \neq 0$) 4 (If $B = 0$)	20 15	C to A_0~A_7 B to A_8~A_{15}
OUT (n),A	(n)←A	●	●	●	●	●	●	11 010 011	2	3	11	n to A_0~A_7 Acc to A_8~A_{15}
OUT (C),r	(C)←r	●	●	●	●	●	●	11 101 101 01 r 001	2	3	12	C to A_0~A_7 B to A_8~A_{15}
OUTI	(C)←(HL) B←B−1 HL←HL+1	●	① ↕	X	X	1	X	11 101 101 10 100 011	2	4	15	C to A_0~A_7 B to A_8~A_{15}
OTIR	(C)←(HL) B←B−1 HL←HL+1 Repeat until B=0.	●	1	X	X	1	X	11 101 101 10 110 011	2 2	5 (If $B \neq 0$) 4 (If $B = 0$)	20 15	C to A_0~A_7 B to A_8~A_{15}
OUTD	(C)←(HL) B←B−1 HL←HL−1	●	① ↕	X	X	1	X	11 101 101 10 101 011	2	4	15	C to A_0~A_7 B to A_8~A_{15}
OTDR	(C)←(HL) B←B−1 HL←HL−1 Repeat until B=0.	●	1	X	X	1	X	11 101 101 10 111 011	2 2	5 (If $B \neq 0$) 4 (If $B = 0$)	20 15	C to A_0~A_7 B to A_8~A_{15}

(Note) ① indicates that if B − 1 = 0, the Z flag is set and in other cases, it is reset.

Statement: Machine Language Comparison Table (alphabetical order)

Hexadecimal notation	Statement
8E	ADC A, (HL)
DD8E*05*	ADC A, (IX+d)
FD8E*05*	ADC A, (IY+d)
8F	ADC A, A
88	ADC A, B
89	ADC A, C
8A	ADC A, D
8B	ADC A, E
8C	ADC A, H
8D	ADC A, L
CE*20*	ADC A, n
ED4A	ADC HL, BC
ED5A	ADC HL, DE
ED6A	ADC HL, HL
ED7A	ADC HL, SP
86	ADD A, (HL)
DD86*05*	ADD A, (IX+d)
FD86*05*	ADD A, (IY+d)
87	ADD A, A
80	ADD A, B
81	ADD A, C
82	ADD A, D
83	ADD A, E
84	ADD A, H
85	ADD A, L
C6*20*	ADD A, n
09	ADD HL, BC
19	ADD HL, DE
29	ADD HL, HL
39	ADD HL, SP
DD09	ADD IX, BC
DD19	ADD IX, DE
DD29	ADD IX, IX
DD39	ADD IX, SP
FD09	ADD IY, BC
FD19	ADD IY, DE
FD29	ADD IY, IY
FD39	ADD IY, SP

Hexadecimal notation	Statement
A6	AND (HL)
DDA6*05*	AND (IX+d)
FDA6*05*	AND (IY+d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E6*20*	AND n
CB46	BIT 0, (HL)
DDCB*05*46	BIT 0, (IX+d)
FDCB*05*46	BIT 0, (IY+d)
CB47	BIT 0, A
CB40	BIT 0, B
CB41	BIT 0, C
CB42	BIT 0, D
CB43	BIT 0, E
CB44	BIT 0, H
CB45	BIT 0, L
CB4E	BIT 1, (HL)
DDCB*05*4E	BIT 1, (IX+d)
FDCB*05*4E	BIT 1, (IY+d)
CB4F	BIT 1, A
CB48	BIT 1, B
CB49	BIT 1, C
CB4A	BIT 1, D
CB4B	BIT 1, E
CB4C	BIT 1, H
CB4D	BIT 1, L
CB56	BIT 2, (HL)
DDCB*05*56	BIT 2, (IX+d)
FDCB*05*56	BIT 2, (IY+d)
CB57	BIT 2, A
CB50	BIT 2, B
CB51	BIT 2, C
CB52	BIT 2, D
CB53	BIT 2, E

Hexadecimal notation	Statement
CB54	BIT 2, H
CB55	BIT 2, L
CB5E	BIT 3, (HL)
DDCB*05*5E	BIT 3, (IX+d)
FDCB*05*5E	BIT 3, (IY+d)
CB5F	BIT 3, A
CB58	BIT 3, B
CB59	BIT 3, C
CB5A	BIT 3, D
CB5B	BIT 3, E
CB5C	BIT 3, H
CB5D	BIT 3, L
CB66	BIT 4, (HL)
DDCB*05*66	BIT 4, (IX+d)
FDCB*05*66	BIT 4, (IY+d)
CB67	BIT 4, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB6E	BIT 5, (HL)
DDCB*05*6E	BIT 5, (IX+d)
FDCB*05*6E	BIT 5, (IY+d)
CB6F	BIT 5, A
CB68	BIT 5, B
CB69	BIT 5, C
CB6A	BIT 5, D
CB6B	BIT 5, E
CB6C	BIT 5, H
CB6D	BIT 5, L
CB76	BIT 6, (HL)
DDCB*05*76	BIT 6, (IX+d)
FDCB*05*76	BIT 6, (IY+d)
CB77	BIT 6, A
CB70	BIT 6, B
CB71	BIT 6, C
CB72	BIT 6, D
CB73	BIT 6, E

Hexadecimal notation	Statement
CB74	BIT 6, H
CB75	BIT 6, L
CB7E	BIT 7, (HL)
DDCB*05*7E	BIT 7, (IX+d)
FDCB*05*7E	BIT 7, (IY+d)
CB7F	BIT 7, A
CB78	BIT 7, B
CB79	BIT 7, C
CB7A	BIT 7, D
CB7B	BIT 7, E
CB7C	BIT 7, H
CB7D	BIT 7, L
DC*8405*	CALL C, nn
FC*8405*	CALL M, nn
D4*8405*	CALL NC, nn
CD*8405*	CALL nn
C4*8405*	CALL NZ, nn
F4*8405*	CALL P, nn
EC*8405*	CALL PE, nn
E4*8405*	CALL PO, nn
CC*8405*	CALL Z, nn
3F	CCF
BE	CP (HL)
DDBE*05*	CP (IX+d)
FDBE*05*	CP (IY+d)
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
FE*20*	CP n
EDA9	CPD
EDB9	CPDR
EDA1	CPI

Hexadecimal notation	Statement
EDB1	CPIR
2F	CPL
27	DAA
35	DEC (HL)
DD35*05*	DEC (IX+d)
FD35*05*	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
10*2E*	DJNZ e
FB	EI
E3	EX (SP),HL
DDE3	EX (SP),IX
FDE3	EX (SP),IY
08	EX AF,AF'
EB	EX DE,HL
D9	EXX
76	HALT
ED46	IM 0
ED56	IM 1

Hexadecimal notation	Statement
ED5E	IM 2
ED78	IN A,(C)
DB*20*	IN A,(n)
ED40	IN B,(C)
ED48	IN C,(C)
ED50	IN D,(C)
ED58	IN E,(C)
ED60	IN H,(C)
ED68	IN L,(C)
34	INC (HL)
DD34*05*	INC (IX+d)
FD34*05*	INC (IY+d)
3C	INC A
04	INC B
03	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD23	INC IX
FD23	INC IY
2C	INC L
33	INC SP
EDAA	IND
EDBA	INDR
EDA2	INI
EDB2	INIR
E9	JP (HL)
DDE9	JP (IX)
FDE9	JP (IY)
DA*8405*	JP C,nn
FA*8405*	JP M,nn
D2*8405*	JP NC,nn
C3*8405*	JP nn

Hexadecimal notation	Statement
C2*8405*	JP NZ,nn
F2*8405*	JP P,nn
EA*8405*	JP PE,nn
E2*8405*	JP PO,nn
CA*8405*	JP Z,nn
38*2E*	JR C,e
18*2E*	JR e
30*2E*	JR NC,e
20*2E*	JR NZ,e
28*2E*	JR Z,e
02	LD (BC),A
12	LD (DE),A
77	LD (HL),A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
36*20*	LD (HL),n
DD77*05*	LD (IX+d),A
DD70*05*	LD (IX+d),B
DD71*05*	LD (IX+d),C
DD72*05*	LD (IX+d),D
DD73*05*	LD (IX+d),E
DD74*05*	LD (IX+d),H
DD75*05*	LD (IX+d),L
DD36*0520*	LD (IX+d),n
FD77*05*	LD (IY+d),A
FD70*05*	LD (IY+d),B
FD71*05*	LD (IY+d),C
FD72*05*	LD (IY+d),D
FD73*05*	LD (IY+d),E
FD74*05*	LD (IY+d),H
FD75*05*	LD (IY+d),L
FD36*0520*	LD (IY+d),n
32*8405*	LD (nn),A
ED43*8405*	LD (nn),BC
ED53*8405*	LD (nn),DE
22*8405*	LD (nn),HL
DD22*8405*	LD (nn),IX
FD22*8405*	LD (nn),IY
ED73*8405*	LD (nn),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD7E*05*	LD A,(IX+d)
FD7E*05*	LD A,(IY+d)
3A*8405*	LD A,(nn)
7F	LD A,A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
ED57	LD A,I
7D	LD A,L
3E*20*	LD A,n
46	LD B,(HL)
DD46*05*	LD B,(IX+d)
FD46*05*	LD B,(IY+d)
47	LD B,A
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
06*20*	LD B,n
ED4B*8405*	LD BC,(nn)
01*8405*	LD BC,nn
4E	LD C,(HL)
DD4E*05*	LD C,(IX+d)
FD4E*05*	LD C,(IY+d)
4F	LD C,A
48	LD C,B
49	LD C,C
4A	LD C,D

Hexadecimal notation	Statement
4B	LD C,E
4C	LD C,H
4D	LD C,L
0E*20*	LD C,n
56	LD D,(HL)
DD56*05*	LD D,(IX+d)
FD56*05*	LD D,(IY+d)
57	LD D,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
16*20*	LD D,n
ED5B*8405*	LD DE,(nn)
11*8405*	LD DE,nn
5E	LD E,(HL)
DD5E*05*	LD E,(IX+d)
FD5E*05*	LD E,(IY+d)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E*20*	LD E,n
66	LD H,(HL)
DD66*05*	LD H,(IX+d)
FD66*05*	LD H,(IY+d)
67	LD H,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
26*20*	LD H,n
2A*8405*	LD HL,(nn)

Hexadecimal notation	Statement
21*8405*	LD HL,nn
ED47	LD I,A
DD2A*8405*	LD IX,(nn)
DD21*8405*	LD IX,nn
FD2A*8405*	LD IY,(nn)
FD21*8405*	LD IY,nn
6E	LD L,(HL)
DD6E*05*	LD L,(IX+d)
FD6E*05*	LD L,(IY+d)
6F	LD L,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
2E*20*	LD L,n
ED7B*8405*	LD SP,(nn)
F9	LD SP,HL
DDF9	LD SP,IX
FDF9	LD SP,IY
31*8405*	LD SP,nn
EDA8	LDD
EDB8	LDDR
EDA0	LDI
EDB0	LDIR
ED44	NEG
00	NOP
B6	OR (HL)
DDB6*05*	OR (IX+d)
FDB6*05*	OR (IY+d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E

Hexadecimal notation	Statement
B4	OR H
B5	OR L
F6*20*	OR n
EDBB	OTDR
EDB3	OTIR
ED79	OUT (C),A
ED41	OUT (C),B
ED49	OUT (C),C
ED51	OUT (C),D
ED59	OUT (C),E
ED61	OUT (C),H
ED69	OUT (C),L
D3*20*	OUT (n),A
EDAB	OUTD
EDA3	OUTI
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DDE1	POP IX
FDE1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DDE5	PUSH IX
FDE5	PUSH IY
CB86	RES 0,(HL)
DDCB*05*86	RES 0,(IX+d)
FDCB*05*86	RES 0,(IY+d)
CB87	RES 0,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H

Hexadecimal notation	Statement
CB85	RES 0,L
CB8E	RES 1,(HL)
DDCB*05*8E	RES 1,(IX+d)
FDCB*05*8E	RES 1,(IY+d)
CB8F	RES 1,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB96	RES 2,(HL)
DDCB*05*96	RES 2,(IX+d)
FDCB*05*96	RES 2,(IY+d)
CB97	RES 2,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB9E	RES 3,(HL)
DDCB*05*9E	RES 3,(IX+d)
FDCB*05*9E	RES 3,(IY+d)
CB9F	RES 3,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CBA6	RES 4,(HL)
DDCB*05*A6	RES 4,(IX+d)
FDCB*05*A6	RES 4,(IY+d)
CBA7	RES 4,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H

Hexadecimal notation	Statement
CBA5	RES 4, L
CBAE	RES 5, (HL)
DDCB05AE	RES 5, (IX+d)
FDCB05AE	RES 5, (IY+d)
CBAF	RES 5, A
CBA8	RES 5, B
CBA9	RES 5, C
CBAA	RES 5, D
CBAB	RES 5, E
CBAC	RES 5, H
CBAD	RES 5, L
CBB6	RES 6, (HL)
DDCB05B6	RES 6, (IX+d)
FDCB05B6	RES 6, (IY+d)
CBB7	RES 6, A
CBB0	RES 6, B
CBB1	RES 6, C
CBB2	RES 6, D
CBB3	RES 6, E
CBB4	RES 6, H
CBB5	RES 6, L
CBBE	RES 7, (HL)
DDCB05BE	RES 7, (IX+d)
FDCB05BE	RES 7, (IY+d)
CBBF	RES 7, A
CBB8	RES 7, B
CBB9	RES 7, C
CBBA	RES 7, D
CBBB	RES 7, E
CBBC	RES 7, H
CBBD	RES 7, L
C9	RET
D8	RET C
F8	RET M
D0	RET NC
C0	RET NZ
F0	RET P
E8	RET PE
E0	RET PO

Hexadecimal notation	Statement
C8	RET Z
ED4D	RETI
ED45	RETN
CB16	RL (HL)
DDCB0516	RL (IX+d)
FDCB0516	RL (IY+d)
CB17	RL A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
17	RLA
CB06	RLC (HL)
DDCB0506	RLC (IX+d)
FDCB0506	RLC (IY+d)
CB07	RLC A
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
07	RLCA
ED6F	RLD
CB1E	RR (HL)
DDCB051E	RR (IX+d)
FDCB051E	RR (IY+d)
CB1F	RR A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA

Hexadecimal notation	Statement
CB0E	RRC (HL)
DDCB*05*0E	RRC (IX+d)
FDCB*05*0E	RRC (IY+d)
CB0F	RRC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
0F	RRCA
ED67	RRD
C7	RST 0
D7	RST 10H
DF	RST 18H
E7	RST 20H
EF	RST 28H
F7	RST 30H
FF	RST 38H
CF	RST 8
9E	SBC A,(HL)
DD9E*05*	SBC A,(IX+d)
FD9E*05*	SBC A,(IY+d)
9F	SBC A,A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
DE*20*	SBC A,n
ED42	SBC HL,BC
ED52	SBC HL,DE
ED62	SBC HL,HL
ED72	SBC HL,SP
37	SCF

Hexadecimal notation	Statement
CBC6	SET 0,(HL)
DDCB*05*C6	SET 0,(IX+d)
FDCB*05*C6	SET 0,(IY+d)
CBC7	SET 0,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBCE	SET 1,(HL)
DDCB*05*CE	SET 1,(IX+d)
FDCB*05*CE	SET 1,(IY+d)
CBCF	SET 1,A
CBC8	SET 1,B
CBC9	SET 1,C
CBCA	SET 1,D
CBCB	SET 1,E
CBCC	SET 1,H
CBCD	SET 1,L
CBD6	SET 2,(HL)
DDCB*05*D6	SET 2,(IX+d)
FDCB*05*D6	SET 2,(IY+d)
CBD7	SET 2,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD8	SET 3,B
CBDE	SET 3,(HL)
DDCB*05*DE	SET 3,(IX+d)
FDCB*05*DE	SET 3,(IY+d)
CBDF	SET 3,A
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L

Hexadecimal notation	Statement
CBE6	SET 4, (HL)
DDCB*05*E6	SET 4, (IX+d)
FDCB*05*E6	SET 4, (IY+d)
CBE7	SET 4, A
CBE0	SET 4, B
CBE1	SET 4, C
CBE2	SET 4, D
CBE3	SET 4, E
CBE4	SET 4, H
CBE5	SET 4, L
CBEE	SET 5, (HL)
DDCB*05*EE	SET 5, (IX+d)
FDCB*05*EE	SET 5, (IY+d)
CBEF	SET 5, A
CBE8	SET 5, B
CBE9	SET 5, C
CBEA	SET 5, D
CBEB	SET 5, E
CBEC	SET 5, H
CBED	SET 5, L
CBF6	SET 6, (HL)
DDCB*05*F6	SET 6, (IX+d)
FDCB*05*F6	SET 6, (IY+d)
CBF7	SET 6, A
CBF0	SET 6, B
CBF1	SET 6, C
CBF2	SET 6, D
CBF3	SET 6, E
CBF4	SET 6, H
CBF5	SET 6, L
CBFE	SET 7, (HL)
DDCB*05*FE	SET 7, (IX+d)
FDCB*05*FE	SET 7, (IY+d)
CBFF	SET 7, A
CBF8	SET 7, B
CBF9	SET 7, C
CBFA	SET 7, D
CBFB	SET 7, E
CBFC	SET 7, H
CBFD	SET 7, L

Hexadecimal notation	Statement
CB26	SLA (HL)
DDCB*05*26	SLA (IX+d)
FDCB*05*26	SLA (IY+d)
CB27	SLA A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB2E	SRA (HL)
DDCB*05*2E	SRA (IX+d)
FDCB*05*2E	SRA (IY+d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB*05*3E	SRL (IX+d)
FDCB*05*3E	SRL (IY+d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD96*05*	SUB (IX+d)
FD96*05*	SUB (IY+d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D

Hexadecimal notation	Statement
93	SUB E
94	SUB H
95	SUB L
D6*20*	SUB n
AE	XOR (HL)
DDAE*05*	XOR (IX+d)
FDAE*05*	XOR (IY+d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE*20*	XOR n

Example

As for symbols **nn, n, d,** and **e**, the following are exemplified; **nn**=584H, **n**=20H, **d**=5, **e**=30H. In hexadecimal notation column, the codes equivalent to these numbers are represented in italics and underlined.

Statement: Machine Language Comparison Table (hexadecimal order)

Hexadecimal notation	Statement
00	NOP
01*8405*	LD BC,nn
02	LD (BC),A
03	INC BC
04	INC B
05	DEC B
06*20*	LD B,n
07	RLCA
08	EX AF,AF'
09	ADD HL,BC
0A	LD A,(BC)
0B	DEC BC
0C	INC C
0D	DEC C
0E*20*	LD C,n
0F	RRCA
10*2E*	DJNZ e
11*8405*	LD DE,nn
12	LD (DE),A
13	INC DE
14	INC D
15	DEC D
16*20*	LD D,n
17	RLA
18*2E*	JR e
19	ADD HL,DE
1A	LD A,(DE)
1B	DEC DE
1C	INC E
1D	DEC E
1E*20*	LD E,n
1F	RRA
20*2E*	JR NZ,e
21*8405*	LD HL,nn
22*8405*	LD (nn),HL
23	INC HL
24	INC H
25	DEC H
26*20*	LD H,n
27	DAA
28*2E*	JR Z,e
29	ADD HL,HL
2A *8405*	LD HL,(nn)
2B	DEC HL
2C	INC L
2D	DEC L
2E *20*	LD L,n
2F	CPL
30*2E*	JR NC,e
31*8405*	LD SP,nn
32*8405*	LD (nn),A
33	INC SP
34	INC (HL)
35	DEC (HL)
36*20*	LD (HL),n
37	SCF
38*2E*	JR C,e
39	ADD HL,SP
3A *8405*	LD A,(nn)
3B	DEC SP
3C	INC A
3D	DEC A
3E*20*	LD A,n
3F	CCF
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
46	LD B,(HL)
47	LD B,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E

Hexadecimal notation	Statement
4C	LD C,H
4D	LD C,L
4E	LD C,(HL)
4F	LD C,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
56	LD D,(HL)
57	LD D,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
5E	LD E,(HL)
5F	LD E,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
66	LD H,(HL)
67	LD H,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
6E	LD L,(HL)
6F	LD L,A
70	LD (HL),B

Hexadecimal notation	Statement
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
76	HALT
77	LD (HL),A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
7D	LD A,L
7E	LD A,(HL)
7F	LD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
86	ADD A,(HL)
87	ADD A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
8E	ADC A,(HL)
8F	ADC A,A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)

Hexadecimal notation	Statement
97	SUB A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
9E	SBC A,(HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H

Hexadecimal notation	Statement
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C2*8405*	JP NZ,nn
C3*8405*	JP nn
C4*8405*	CALL NZ,nn
C5	PUSH BC
C6*20*	ADD A,n
C7	RST 0
C8	RET Z
C9	RET
CA*8405*	JP Z,nn
CC*8405*	CALL Z,nn
CD*8405*	CALL nn
CE*20*	ADC A,n
CF	RST 8
D0	RET NC
D1	POP DE
D2*8405*	JP NC,nn
D3*20*	OUT (n),A
D4*8405*	CALL NC,nn
D5	PUSH DE
D6*20*	SUB n
D7	RST 10H
D8	RET C
D9	EXX
DA*8405*	JP C,nn
DB*20*	IN A,(n)
DC*8405*	CALL C,nn
DE*20*	SBC A,n
DF	RST 18H
E0	RET PO
E1	POP HL
E2*8405*	JP PO,nn
E3	EX (SP),HL

Hexadecimal notation	Statement
E4*8405*	CALL PO,nn
E5	PUSH HL
E6*20*	AND n
E7	RST 20H
E8	RET PE
E9	JP (HL)
EA*8405*	JP PE,nn
EB	EX DE,HL
EC*8405*	CALL PE,nn
EE*20*	XOR n
EF	RST 28H
F0	RET P
F1	POP AF
F2*8405*	JP P,nn
F3	DI
F4*8405*	CALL P,nn
F5	PUSH AF
F6*20*	OR n
F7	RST 30H
F8	RET M
F9	LD SP,HL
FA*8405*	JP M,nn
FB	EI
FC*8405*	CALL M,nn
FE*20*	CP n
FF	RST 38H
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E

Hexadecimal notation	Statement
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	RRC A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
CB16	RL (HL)
CB17	RL A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
CB1E	RR (HL)
CB1F	RR A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB26	SLA (HL)
CB27	SLA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SRA (HL)
CB2F	SRA A
CB38	SRL B

Hexadecimal notation	Statement
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
CB3E	SRL (HL)
CB3F	SRL A
CB40	BIT 0, B
CB41	BIT 0, C
CB42	BIT 0, D
CB43	BIT 0, E
CB44	BIT 0, H
CB45	BIT 0, L
CB46	BIT 0, (HL)
CB47	BIT 0, A
CB48	BIT 1, B
CB49	BIT 1, C
CB4A	BIT 1, D
CB4B	BIT 1, E
CB4C	BIT 1, H
CB4D	BIT 1, L
CB4E	BIT 1, (HL)
CB4F	BIT 1, A
CB50	BIT 2, B
CB51	BIT 2, C
CB52	BIT 2, D
CB53	BIT 2, E
CB54	BIT 2, H
CB55	BIT 2, L
CB56	BIT 2, (HL)
CB57	BIT 2, A
CB58	BIT 3, B
CB59	BIT 3, C
CB5A	BIT 3, D
CB5B	BIT 3, E
CB5C	BIT 3, H
CB5D	BIT 3, L
CB5E	BIT 3, (HL)

Hexadecimal notation	Statement
CB5F	BIT 3, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB66	BIT 4, (HL)
CB67	BIT 4, A
CB68	BIT 5, B
CB69	BIT 5, C
CB6A	BIT 5, D
CB6B	BIT 5, E
CB6C	BIT 5, H
CB6D	BIT 5, L
CB6E	BIT 5, (HL)
CB6F	BIT 5, A
CB70	BIT 6, B
CB71	BIT 6, C
CB72	BIT 6, D
CB73	BIT 6, E
CB74	BIT 6, H
CB75	BIT 6, L
CB76	BIT 6, (HL)
CB77	BIT 6, A
CB78	BIT 7, B
CB79	BIT 7, C
CB7A	BIT 7, D
CB7B	BIT 7, E
CB7C	BIT 7, H
CB7D	BIT 7, L
CB7E	BIT 7, (HL)
CB7F	BIT 7, A
CB80	RES 0, B
CB81	RES 0, C
CB82	RES 0, D
CB83	RES 0, E

Hexadecimal notation	Statement
CB84	RES 0, H
CB85	RES 0, L
CB86	RES 0, (HL)
CB87	RES 0, A
CB88	RES 1, B
CB89	RES 1, C
CB8A	RES 1, D
CB8B	RES 1, E
CB8C	RES 1, H
CB8D	RES 1, L
CB8E	RES 1, (HL)
CB8F	RES 1, A
CB90	RES 2, B
CB91	RES 2, C
CB92	RES 2, D
CB93	RES 2, E
CB94	RES 2, H
CB95	RES 2, L
CB96	RES 2, (HL)
CB97	RES 2, A
CB98	RES 3, B
CB99	RES 3, C
CB9A	RES 3, D
CB9B	RES 3, E
CB9C	RES 3, H
CB9D	RES 3, L
CB9E	RES 3, (HL)
CB9F	RES 3, A
CBA0	RES 4, B
CBA1	RES 4, C
CBA2	RES 4, D
CBA3	RES 4, E
CBA4	RES 4, H
CBA5	RES 4, L
CBA6	RES 4, (HL)
CBA7	RES 4, A
CBA8	RES 5, B
CBA9	RES 5, C

Hexadecimal notation	Statement
CBAA	RES 5, D
CBAB	RES 5, E
CBAC	RES 5, H
CBAD	RES 5, L
CBAE	RES 5, (HL)
CBAF	RES 5, A
CBB0	RES 6, B
CBB1	RES 6, C
CBB2	RES 6, D
CBB3	RES 6, E
CBB4	RES 6, H
CBB5	RES 6, L
CBB6	RES 6, (HL)
CBB7	RES 6, A
CBB8	RES 7, B
CBB9	RES 7, C
CBBA	RES 7, D
CBBB	RES 7, E
CBBC	RES 7, H
CBBD	RES 7, L
CBBE	RES 7, (HL)
CBBF	RES 7, A
CBC0	SET 0, B
CBC1	SET 0, C
CBC2	SET 0, D
CBC3	SET 0, E
CBC4	SET 0, H
CBC5	SET 0, L
CBC6	SET 0, (HL)
CBC7	SET 0, A
CBC8	SET 1, B
CBC9	SET 1, C
CBCA	SET 1, D
CBCB	SET 1, E
CBCC	SET 1, H
CBCD	SET 1, L
CBCE	SET 1, (HL)
CBCF	SET 1, A

Hexadecimal notation	Statement
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD6	SET 2,(HL)
CBD7	SET 2,A
CBD8	SET 3,B
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBDE	SET 3,(HL)
CBDF	SET 3,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E
CBE4	SET 4,H
CBE5	SET 4,L
CBE6	SET 4,(HL)
CBE7	SET 4,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBEE	SET 5,(HL)
CBEF	SET 5,A
CBF0	SET 6,B
CBF1	SET 6,C
CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L

Hexadecimal notation	Statement
CBF6	SET 6,(HL)
CBF7	SET 6,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L
CBFE	SET 7,(HL)
CBFF	SET 7,A
DD09	ADD IX,BC
DD19	ADD IX,DE
DD21 *8405*	LD IX,nn
DD22 *8405*	LD (nn),IX
DD23	INC IX
DD29	ADD IX,IX
DD2A*8405*	LD IX,(nn)
DD2B	DEC IX
DD34 *05*	INC (IX+d)
DD35 *05*	DEC (IX+d)
DD36 *0520*	LD (IX+d),n
DD39	ADD IX,SP
DD46 *05*	LD B,(IX+d)
DD4E*05*	LD C,(IX+d)
DD56 *05*	LD D,(IX+d)
DD5E*05*	LD E,(IX+d)
DD66 *05*	LD H,(IX+d)
DD6E*05*	LD L,(IX+d)
DD70 *05*	LD (IX+d),B
DD71 *05*	LD (IX+d),C
DD72 *05*	LD (IX+d),D
DD73 *05*	LD (IX+d),E
DD74 *05*	LD (IX+d),H
DD75 *05*	LD (IX+d),L
DD77 *05*	LD (IX+d),A
DD7E*05*	LD A,(IX+d)
DD86 *05*	ADD A,(IX+d)
DD8E*05*	ADC A,(IX+d)
DD96 *05*	SUB (IX+d)

Hexadecimal notation	Statement
DD9E *05*	SBC A,(IX+d)
DDA6 *05*	AND (IX+d)
DDAE*05*	XOR (IX+d)
DDB6 *05*	OR (IX+d)
DDBE*05*	CP (IX+d)
DDE1	POP IX
DDE3	EX (SP),IX
DDE5	PUSH IX
DDE9	JP (IX)
DDF9	LD SP,IX
DDCB*05*06	RLC (IX+d)
DDCB*05*0E	RRC (IX+d)
DDCB*05*16	RL (IX+d)
DDCB*05*1E	RR (IX+d)
DDCB*05*26	SLA (IX+d)
DDCB*05*2E	SRA (IX+d)
DDCB*05*3E	SRL (IX+d)
DDCB*05*46	BIT 0,(IX+d)
DDCB*05*4E	BIT 1,(IX+d)
DDCB*05*56	BIT 2,(IX+d)
DDCB*05*5E	BIT 3,(IX+d)
DDCB*05*66	BIT 4,(IX+d)
DDCB*05*6E	BIT 5,(IX+d)
DDCB*05*76	BIT 6,(IX+d)
DDCB*05*7E	BIT 7,(IX+d)
DDCB*05*86	RES 0,(IX+d)
DDCB*05*8E	RES 1,(IX+d)
DDCB*05*96	RES 2,(IX+d)
DDCB*05*9E	RES 3,(IX+d)
DDCB*05*A6	RES 4,(IX+d)
DDCB*05*AE	RES 5,(IX+d)
DDCB*05*B6	RES 6,(IX+d)
DDCB*05*BE	RES 7,(IX+d)
DDCB*05*C6	SET 0,(IX+d)
DDCB*05*CE	SET 1,(IX+d)
DDCB*05*D6	SET 2,(IX+d)
DDCB*05*DE	SET 3,(IX+d)
DDCB*05*E6	SET 4,(IX+d)
DDCB*05*EE	SET 5,(IX+d)

Hexadecimal notation	Statement
DDCB*05*F6	SET 6,(IX+d)
DDCB*05*FE	SET 7,(IX+d)
ED40	IN B,(C)
ED41	OUT (C),B
ED42	SBC HL,BC
ED43*8405*	LD (nn),BC
ED44	NEG
ED45	RETN
ED46	IM 0
ED47	LD I,A
ED48	IN C,(C)
ED49	OUT (C),C
ED4A	ADC HL,BC
ED4B*8405*	LD BC,(nn)
ED4D	RETI
ED50	IN D,(C)
ED51	OUT (C),D
ED52	SBC HL,DE
ED53*8405*	LD (nn),DE
ED56	IM 1
ED57	LD A,I
ED58	IN E,(C)
ED59	OUT (C),E
ED5A	ADC HL,DE
ED5B*8405*	LD DE,(nn)
ED5E	IM 2
ED60	IN H,(C)
ED61	OUT (C),H
ED62	SBC HL,HL
ED67	RRD
ED68	IN L,(C)
ED69	OUT (C),L
ED6A	ADC HL,HL
ED6F	RLD
ED72	SBC HL,SP
ED73*8405*	LD (nn),SP
ED78	IN A,(C)
ED79	OUT (C),A
ED7A	ADC HL,SP

Hexadecimal notation	Statement
ED7B8405	LD SP,(nn)
EDA0	LDI
EDA1	CPI
EDA2	INI
EDA3	OUTI
EDA8	LDD
EDA9	CPD
EDAA	IND
EDAB	OUTD
EDB0	LDIR
EDB1	CPIR
EDB2	INIR
EDB3	OTIR
EDB8	LDDR
EDB9	CPDR
EDBA	INDR
EDBB	OTDR
FD09	ADD IY,BC
FD19	ADD IY,DE
FD218405	LD IY,nn
FD228405	LD (nn),IY
FD23	INC IY
FD29	ADD IY,IY
FD2A8405	LD IY,(nn)
FD2B	DEC IY
FD3405	INC (IY+d)
FD3505	DEC (IY+d)
FD360520	LD (IY+d),n
FD39	ADD IY,SP
FD46 05	LD B,(IY+d)
FD4E05	LD C,(IY+d)
FD56 05	LD D,(IY+d)
FD5E05	LD E,(IY+d)
FD66 05	LD H,(IY+d)
FD6E05	LD L,(IY+d)
FD70 05	LD (IY+d),B
FD71 05	LD (IY+d),C
FD72 05	LD (IY+d),D
FD73 05	LD (IY+d),E

Hexadecimal notation	Statement
FD74 05	LD (IY+d),H
FD75 05	LD (IY+d),L
FD77 05	LD (IY+d),A
FD7E 05	LD A,(IY+d)
FD86 05	ADD A,(IY+d)
FD8E 05	ADC A,(IY+d)
FD96 05	SUB (IY+d)
FD9E 05	SBC A,(IY+d)
FDA6 05	AND (IY+d)
FDAE05	XOR (IY+d)
FDB6 05	OR (IY+d)
FDBE05	CP (IY+d)
FDE1	POP IY
FDE3	EX (SP),IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP,IY
FDCB0506	RLC (IY+d)
FDCB050E	RRC (IY+d)
FDCB0516	RL (IY+d)
FDCB051E	RR (IY+d)
FDCB0526	SLA (IY+d)
FDCB052E	SRA (IY+d)
FDCB053E	SRL (IY+d)
FDCB0546	BIT 0,(IY+d)
FDCB054E	BIT 1,(IY+d)
FDCB0556	BIT 2,(IY+d)
FDCB055E	BIT 3,(IY+d)
FDCB0566	BIT 4,(IY+d)
FDCB056E	BIT 5,(IY+d)
FDCB0576	BIT 6,(IY+d)
FDCB057E	BIT 7,(IY+d)
FDCB0586	RES 0,(IY+d)
FDCB058E	RES 1,(IY+d)
FDCB0596	RES 2,(IY+d)
FDCB059E	RES 3,(IY+d)
FDCB05A6	RES 4,(IY+d)
FDCB05AE	RES 5,(IY+d)
FDCB05B6	RES 6,(IY+d)

Hexadecimal notation	Statement
FDCB*05*BE	RES 7,(IY+d)
FDCB*05*C6	SET 0,(IY+d)
FDCB*05*CE	SET 1,(IY+d)
FDCB*05*D6	SET 2,(IY+d)
FDCB*05*DE	SET 3,(IY+d)
FDCB*05*E6	SET 4,(IY+d)
FDCB*05*EE	SET 5,(IY+d)
FDCB*05*F6	SET 6,(IY+d)
FDCB*05*FE	SET 7,(IY+d)

Example

As for symbols **nn, n, d,** and **e,** the following are xemplified; **nn**=584H, **n**=20H, **d**=5, **e**=30H. In hexadecimal notation column, the codes equivalent to these numbers are represented in italics and underlined.