UNIVERSITI SAINS MALAYSIA

Peperiksaan Semester Kedua
Sidang Akademik 1995/96

Mac/April 1996

ZSE 416 - Pengantar Mikropemprosesan/Mikrokomputer

Masa : [3 jam]

Sila pastikan bahawa kertas peperiksaan ini mengandungi TUJUH muka surat yang bercetak sebelum anda memulakan peperiksaan ini.

Jawab kesemua ENAM soalan. Kesemuanya wajib dijawab di dalam Bahasa Malaysia.

1(a) Tuliskan aturcara penghimpunan serta kod mesin (heksadesimal) bagi mengira 51 + 61 + 18 - 32. Aturcara tersebut hendaklah disimpan di lokasi RAM bermula daripada alamat 2030H manakala jawapan daripada hasil pengiraan tersebut hendaklah disimpan di lokasi ingatan 20BBH.

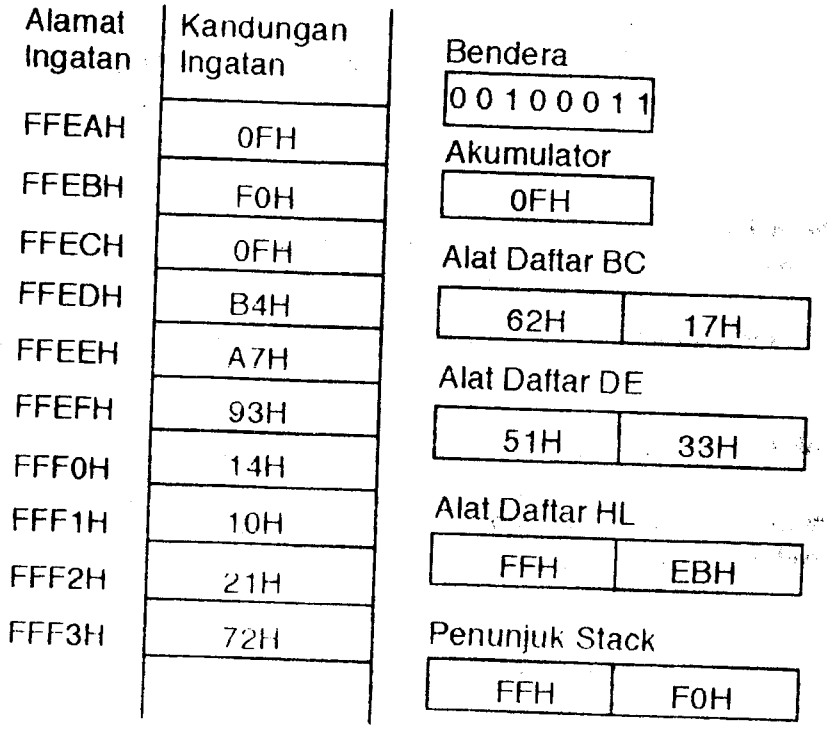
(40/100)

(b) Nyatakan dua kegunaan utama stack.

(20/100)

(c) Rajah 1 menunjukkan kandungan semasa sebahagian ingatan dan alat daftar CPU yang sepadan. Tunjukkan kandungan bahagian ingatan dan alat daftar CPU tersebut selepas perlaksanaan turutan arahan berikut:

```
LDA FFEAH
ADC M
PUSH BC
PUSH DE
POP  BC
```



Rajah 1

(40/100)

2(a) **Terangkan tiga jenis mod pengalamatan dan berikan satu contoh arahan bagi setiap mod pengalamatan tersebut.**

(20/100)

(b) **Fahami aturcara penghimpunan berikut:**

```
        MVI A, FEH
        MVI B, 01H
        MVI C, 32H

GELUNG: SUB B
        DCR C
        JNZ GELUNG
        STA AAFFH
        HLT
```

Berdasarkan aturcara penghimpunan di atas:

(i) Berapa kalikah DCR C dilaksanakan? (5/100)

(ii) Berapakah kalikah aturcara melompat ke GELUNG? (5/100)

(iii) Apakah kandungan ingatan AAFFH setelah selesai aturcara di atas dilaksanakan? (5/100)

(iv) Kirakan masa yang diambil oleh sistem mikropemproses 8085 bagi melaksanakan aturcara tersebut sekiranya frekuensi jam adalah 2.5 MHz. (35/100)

(v) Bagaimanakah aturcara tersebut boleh diubahsuai supaya boleh melompat ke GELUNG sebanyak 250 kali? (15/100)

(vi) Bagaimana pula mengubahsuainya bagi melompat 300 kali? (15/100)

3(a) Terangkan maksud ingatan berlipat di dalam pengalamatan sesuatu sistem mikropemproses. (20/100)

(b) Berdasarkan sistem mikropemproses 8085 dalam Rajah 2 tentukan alamat-alamat berikut:-

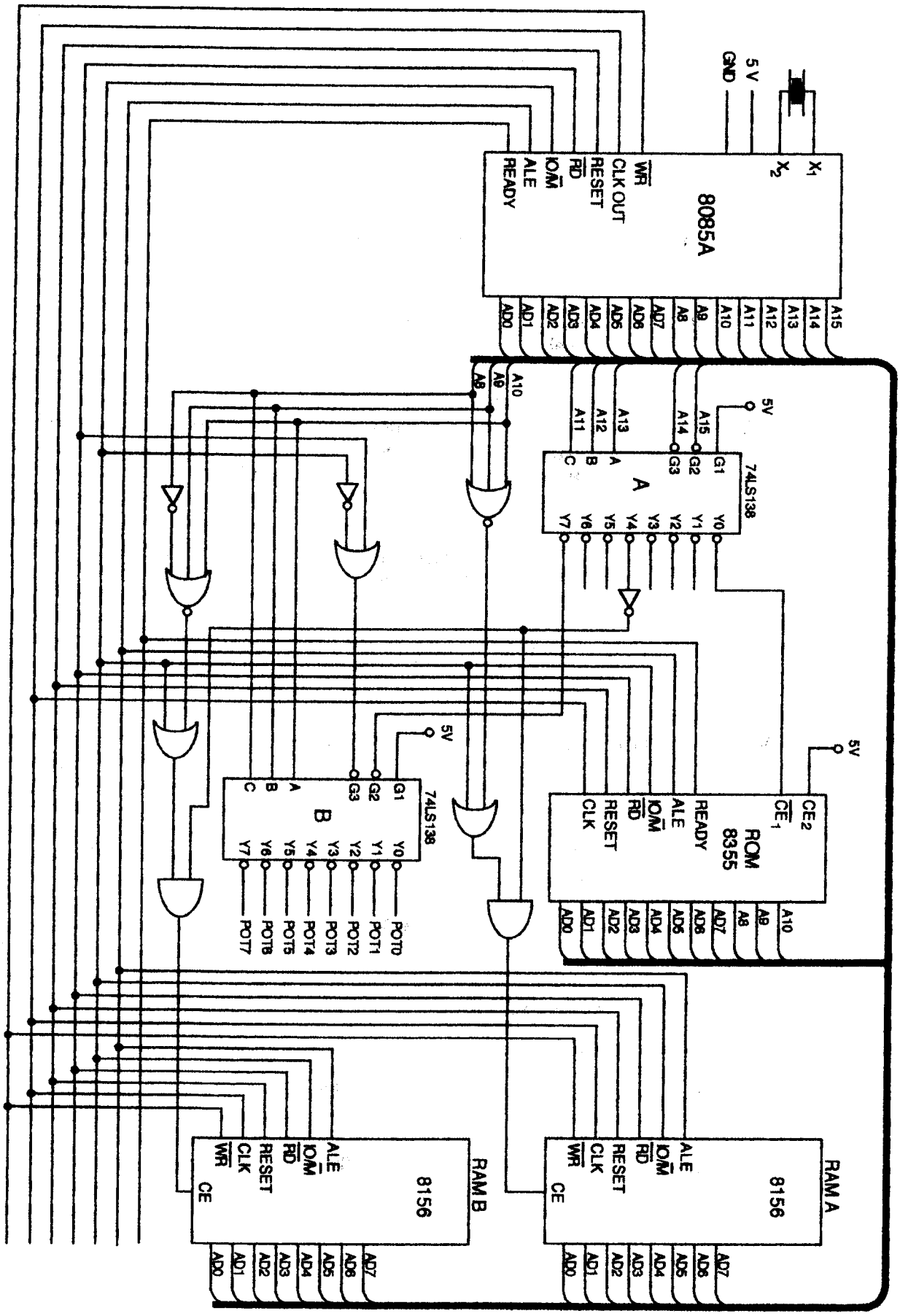
(i) Lokasi alamat bagi ROM, RAM A dan RAM B (20/100)

(ii) Alamat bagi pot 0 hingga pot 7 (25/100)

(iii) Alamat-alamat lain yang dikodkan oleh pengekodan A (25/100)

(iv) Lokasi alamat yang tidak dikodkan oleh pengekodan A. (10/100)

...4/-


8085A
X1
X2
5 V
GND
WR
CLK OUT
RESET
RD
IO/M
ALE
READY
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
74LS138
A
B
C
G1
G2
G3
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
5V
ROM
8355
CE2
CE1
CLK
RESET
READY
ALE
IO/M
RD
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A8
A9
A10
POT0
POT1
POT2
POT3
POT4
POT5
POT6
POT7
RAM A
RAM B
8156
WR
CLK
RESET
ALE
IO/M
RD
CE

Rajah 2

...5/-

4(a) Terangkan maksud *pertindihan ambil-laksana* (*fetch-execute overlap*) di dalam perlaksanaan arahan mikropemproses 8085.

(20/100)

(b) Aturcara berikut di bawah merupakan aturcara ujian bagi memaparkan kitaran mesin di layar osiloskop. Fungsi keseluruhan aturcara tersebut ialah untuk mengambil data dari pot input 29H yang bernilai F0H, kemudian menyimpan data tersebut ke satu lokasi ingatan RAM.

alamat	mnemonik		arahan
2010H	21H 30H 20H		LX1 H, 2030H
2013H	D8H 29H	GELUNG:	IN 29H
2015H	77H		MOV M, A
2016H	C3H 13H 20H		JMP GELUNG
2019H	76H		HLT

(i) Nyatakan lokasi RAM di mana data input tersebut disimpankan.

(10/100)

(ii) Senaraikan pin-keluaran mikropemproses yang harus disambung kepada pemacu-luar osiloskop bagi memastikan arahan tersebut dilaksanakan oleh mikropemproses.

(15/100)

(iii) Terangkan bahagian manakah daripada aturcara tersebut di atas yang akan menghasilkan gelombang pegun di layar osiloskop?

(10/100)

(iv) Dengan menggunakan kertas gelombang isyarat yang dibekalkan, lukiskan gelombang pegun yang terhasil di layar osiloskop semasa aturcara tersebut dijalankan (run). Tuliskan nilai pada bas data dan alamat di setiap kitaran mesin dalam lakaran tersebut.

(45/100)

5(a) Terangkan maksud ungkapan berikut merujuk kepada sampukan mikropemproses 8085:

(i) Restart perisian

(10/100)

(ii) Restart perkakasan

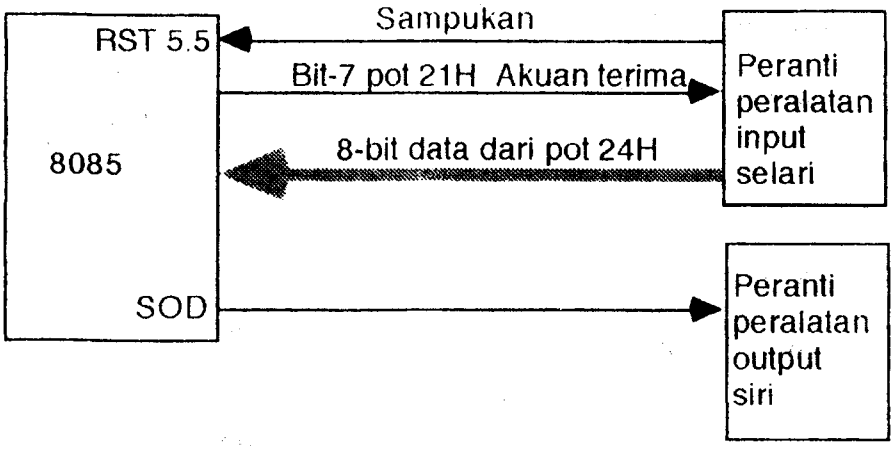
(10/100)

(iii) Sampukan bertopeng

(10/100)

...6/-

(b) Rajah 3 menunjukan peranti peralatan yang disambung kepada sampukkan RST 5.5. Selepas CPU menerima byte data daripada pot 24H, isyarat akuan terima akan dihantar semula kepada peranti peralatan melalui bit-7 pot 21 yang berlogik 1. Arahan di lokasi vektor RST 5.5 adalah JMP F100H.



Rajah 3

(i) Terangkan tindakbalas mikropemproses apabila ia menerima bit 1 daripada input RST 5.5

(30/100)

(ii) Tunjukkan subrutin yang akan membaca data input daripada peranti peralatan tersebut kemudian mengeluarkannya secara selari melalui pin SOD.

(40/100)

6. Suatu sistem mikropemproses 8085 menggunakan cip 8156 sebagai sebahagian daripada RAM dan pot input/output.

(a) Sekiranya lokasi alamat RAM 8156 adalah 3000H hingga 30FFH tentukan alamat bagi:

(i) Alat daftar status-perintah

(5/100)

(ii) Pot-pot input/output

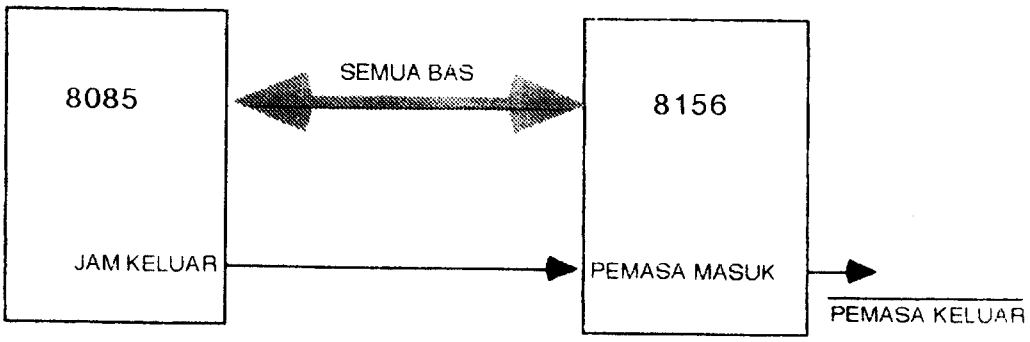
(10/100)

(iii) Byte-byte pembilang pemasa

(5/100)

...7/-

(b) Rajah 4 menunjukkan pin jam keluaran mikropemproses 8085 disambung kepada input pemasa cip 8156. Sekiranya frekuensi jam adalah 2.5 MHz, tunjukkan segmen aturcara yang membolehkan pemasa keluaran cip 8156 mengeluarkan gelombang persegi 1 kHz secara berterusan, berserta dengan perintah pemula pemasa, membenarkan pot sampukan, menjadikan pot A sebagai pot input, pot B sebagai pot output dan pot C sebagai pot berjabat tangan sepenuhnya.



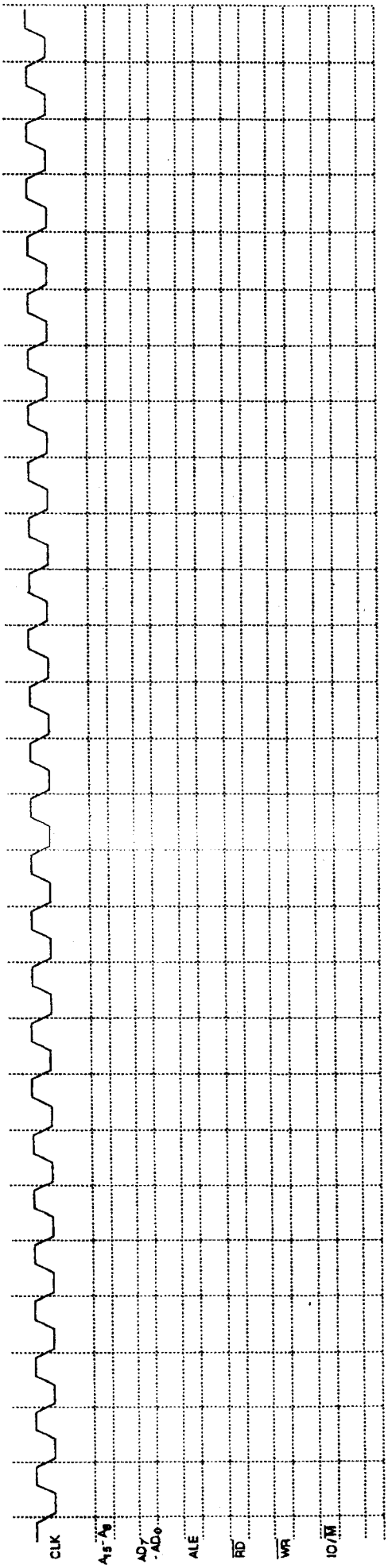
Rajah 4

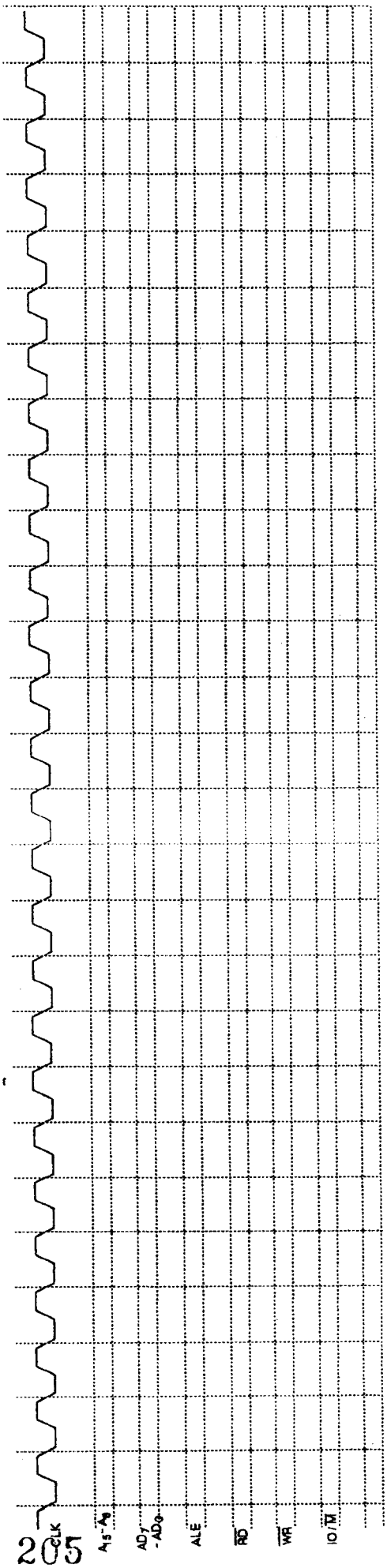
(50/100)

(c) Berdasarkan segmen aturcara (b) lukiskan litar pandu-sampuk yang berjabat tangan dengan pot A dan B.

(30/100)

- ooo0ooo -


CLK
A15-A8
AD7-AD0
ALE
RD
WR
IO/M


CLK
A15-A8
AD7-AD0
ALE
RD
WR
IO/M

8085 Instruction Set

4.6 INSTRUCTION SET ENCYCLOPEDIA

In the ensuing dozen pages, the complete 8085A instruction set is described, grouped in order under five different functional headings, as follows:

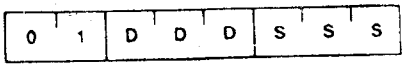
1. **Data Transfer Group** — Moves data between registers or between memory locations and registers. Includes moves, loads, stores, and exchanges. (See below.)
2. **Arithmetic Group** — Adds, subtracts, increments, or decrements data in registers or memory. (See page 4-13.)
3. **Logic Group** — ANDs, ORs, XORs, compares, rotates, or complements data in registers or between memory and a register. (See page 4-16.)
4. **Branch Group** — Initiates conditional or unconditional jumps, calls, returns, and restarts. (See page 4-20.)
5. **Stack, I/O, and Machine Control Group** — Includes instructions for maintaining the stack, reading from input ports, writing to output ports, setting and reading interrupt masks, and setting and clearing flags. (See page 4-22.)

The formats described in the encyclopedia reflect the assembly language processed by Intel-supplied assembler, used with the Intellec® development systems.

4.6.1 Data Transfer Group

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected by any instruction in this group.**

MOV r1, r2 (Move Register)
(r1) ← (r2)
The content of register r2 is moved to register r1.

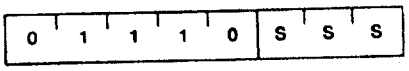


Cycles: 1
States: 4
Addressing: register
Flags: none

MOV r, M (Move from memory)
(r) ← ((H) (L))
The content of the memory location, whose address is in registers H and L, is moved to register r.

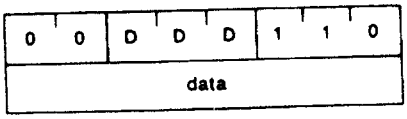
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MOV M, r (Move to memory)
((H)) (L)) ← (r)
The content of register r is moved to the memory location whose address is in registers H and L.



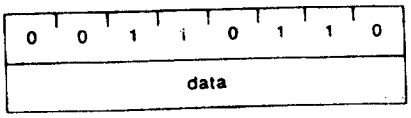
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MVI r, data (Move Immediate)
(r) ← (byte 2)
The content of byte 2 of the instruction is moved to register r.



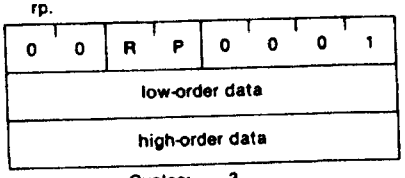
Cycles: 2
States: 7
Addressing: immediate
Flags: none

MVI M, data (Move to memory immediate)
((H) (L)) ← (byte 2)
The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.



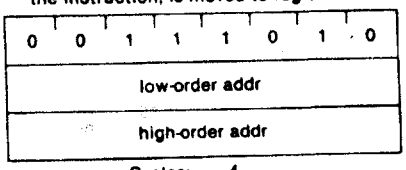
Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

LXI rp, data 16 (Load register pair immediate)
(rh) ← (byte 3),
(rl) ← (byte 2)
Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.



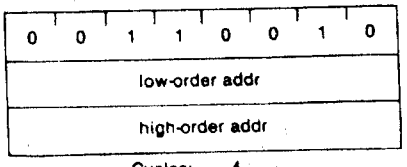
Cycles: 3
States: 10
Addressing: immediate
Flags: none

LDA addr (Load Accumulator direct)
(A) ← ((byte 3)(byte 2))
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.



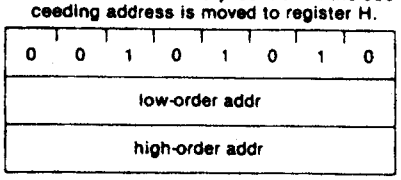
Cycles: 4
States: 13
Addressing: direct
Flags: none

STA addr (Store Accumulator direct)
((byte 3)(byte 2)) ← (A)
The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.



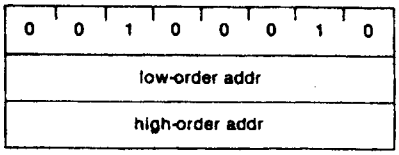
Cycles: 4
States: 13
Addressing: direct
Flags: none

LHLD addr (Load H and L direct)
(L) – ((byte 3)(byte 2))
(H) – ((byte 3)(byte 2) + 1)
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.



Cycles: 5
States: 16
Addressing: direct
Flags: none

SHLD addr (Store H and L direct)
((byte 3)(byte 2)) – (L)
((byte 3)(byte 2) + 1) – (H)
The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.



Cycles: 5
States: 16
Addressing: direct
Flags: none

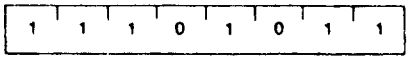
LDAX rp (Load accumulator indirect)
(A) – ((rp))
The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

STAX rp (Store accumulator indirect)
((rp)) – (A)
The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

XCHG (Exchange H and L with D and E)
(H) – (D)
(L) – (E)
The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1
States: 4
Addressing: register
Flags: none

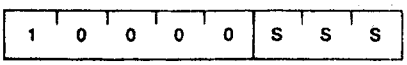
4.6.2 Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

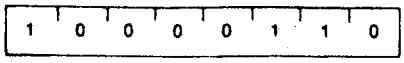
All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add Register)
(A) – (A) + (r)
The content of register r is added to the content of the accumulator. The result is placed in the accumulator.



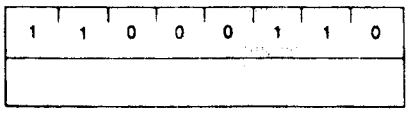
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADD M (Add memory)
(A) – (A) + ((H) (L))
The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.



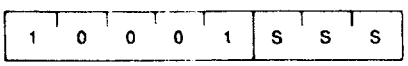
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ADI data (Add immediate)
(A) – (A) + (byte 2)
The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



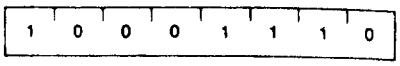
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ADC r (Add Register with carry)
(A) – (A) + (r) + (CY)
The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.



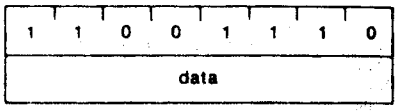
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADC M (Add memory with carry)
(A) – (A) + ((H) (L)) + (CY)
The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.



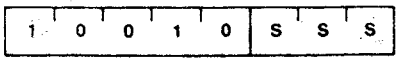
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ACI data (Add immediate with carry)
(A) – (A) + (byte 2) + (CY)
The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.



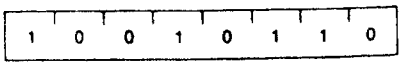
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SUB r (Subtract Register)
(A) – (A) – (r)
The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



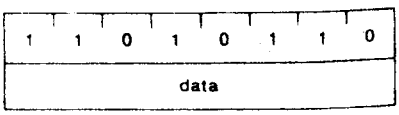
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SUB M (Subtract memory)
(A) – (A) – ((H) (L))
The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.



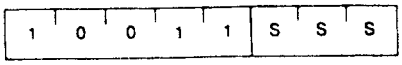
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SUI data (Subtract immediate)
(A) – (A) – (byte 2)
The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.



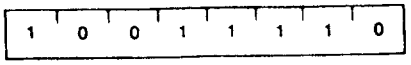
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SBB r (Subtract Register with borrow)
(A) ← (A) − (r) − (CY)
The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



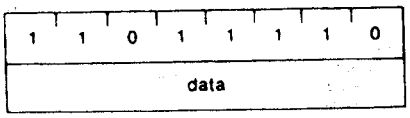
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SBB M (Subtract memory with borrow)
(A) ← (A) − ((H) (L)) − (CY)
The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



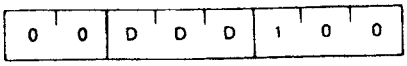
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SBI data (Subtract immediate with borrow)
(A) ← (A) − (byte 2) − (CY)
The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



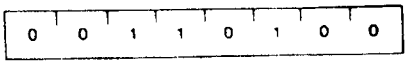
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

INR r (Increment Register)
(r) ← (r) + 1
The content of register r is incremented by one. Note: All condition flags **except CY** are affected.



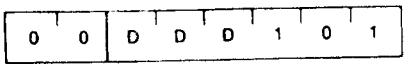
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,AC

INR M (Increment memory)
((H) (L)) ← ((H) (L)) + 1
The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags **except CY** are affected.



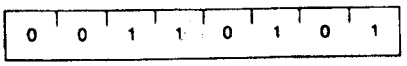
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

DCR r (Decrement Register)
(r) ← (r) − 1
The content of register r is decremented by one. Note: All condition flags **except CY** are affected.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,AC

DCR M (Decrement memory)
((H) (L)) ← ((H) (L)) − 1
The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags **except CY** are affected.

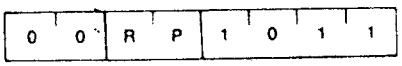


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

INX rp (Increment register pair)
(rh) (rl) ← (rh) (rl) + 1
The content of the register pair rp is incremented by one. Note: **No condition flags are affected.**

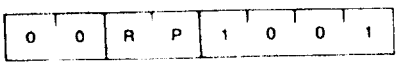
Cycles: 1
States: 6
Addressing: register
Flags: none

DCX rp (Decrement register pair)
(rh) (rl) ← (rh) (rl) − 1
The content of the register pair rp is decremented by one. Note: **No condition flags are affected.**



Cycles: 1
States: 6
Addressing: register
Flags: none

DAD rp (Add register pair to H and L)
(H) (L) ← (H) (L) + (rh) (rl)
The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add, otherwise it is reset.

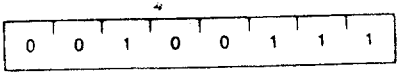


Cycles: 3
States: 10
Addressing: register
Flags: CY

DAA (Decimal Adjust Accumulator)
The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the lease significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.



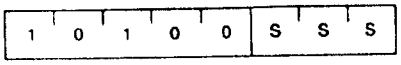
Cycles: 1
States: 4
Flags: Z,S,P,CY,AC

4.6.3 Logic Group

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

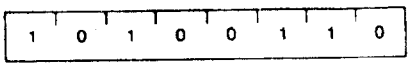
Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

ANA r (AND Register)
(A) ← (A) ∧ (r)
The content of register r is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**



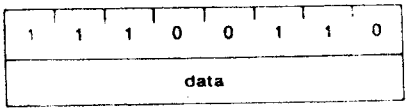
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ANA M (AND memory)
(A) ← (A) ∧ ((H) (L))
The contents of the memory location whose address is contained in the H and L registers is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ANI data (AND immediate)
(A) ← (A) ∧ (byte 2)
The content of the second byte of the instruction is logically ANDed with the contents of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

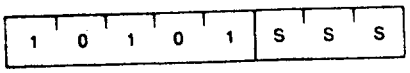


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA r (Exclusive OR Register)

(A) ← (A) ⊻ (r)

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

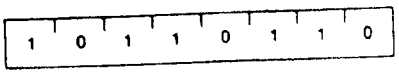


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ORA M (OR memory)

(A) ← (A) V ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

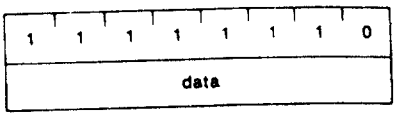


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

CPI data (Compare immediate)

(A) - (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. **The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).**

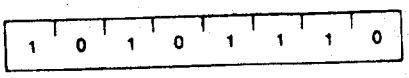


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA M (Exclusive OR Memory)

(A) ← (A) ⊻ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

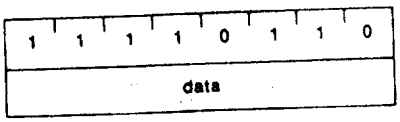


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ORI data (OR Immediate)

(A) ← (A) V (byte 2)

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**.



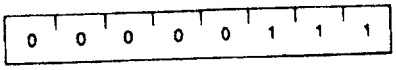
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$

(CY) ← (A_7)

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**

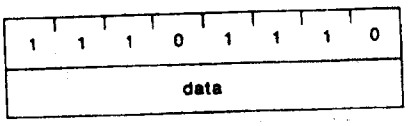


Cycles: 1
States: 4
Flags: CY

XRI data (Exclusive OR immediate)

(A) ← (A) ⊻ (byte 2)

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

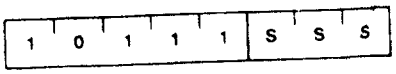


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

CMP r (Compare Register)

(A) - (r)

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**



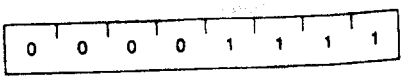
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

RRC (Rotate right)

$(A_n) \leftarrow (A_{n+1})$; $(A_7) \leftarrow (A_0)$

(CY) ← (A_0)

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**

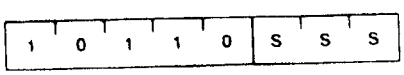


Cycles: 1
States: 4
Flags: CY

ORA r (OR Register)

(A) ← (A) V (r)

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

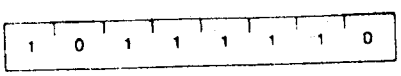


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

CMP M (Compare memory)

(A) - ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).**



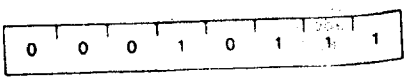
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; (CY) ← (A_7)

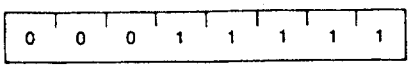
(A_0) ← (CY)

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**



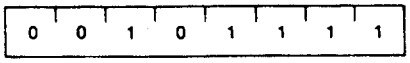
Cycles: 1
States: 4
Flags: CY

RAR (Rotate right through carry)
$(A_n) \leftarrow (A_{n+1})$;$(CY) \leftarrow (A_0)$
$(A_7) \leftarrow (CY)$
The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**



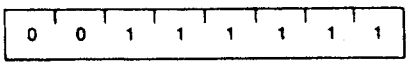
Cycles: 1
States: 4
Flags: CY

CMA (Complement accumulator)
$(A) \leftarrow (\overline{A})$
The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**



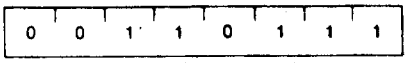
Cycles: 1
States: 4
Flags: none

CMC (Complement carry)
$(CY) \leftarrow (CY)$
The CY flag is complemented. **No other flags are affected.**



Cycles: 1
States: 4
Flags: CY

STC (Set carry)
$(CY) \leftarrow 1$
The CY flag is set to 1. **No other flags are affected.**



Cycles: 1
States: 4
Flags: CY

4.6.4 Branch Group

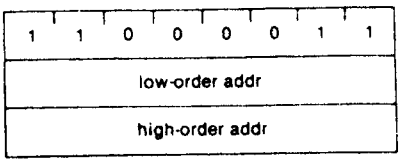
This group of instructions alter normal sequential program flow.

Condition flags are not affected by any instruction in this group.

The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

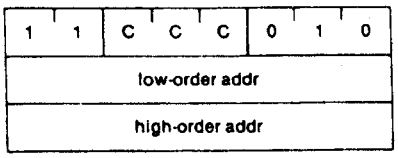
CONDITION	CCC
NZ — not zero (Z = 0)	000
Z — zero (Z = 1)	001
NC — no carry (CY = 0)	010
C — carry (CY = 1)	011
PO — parity odd (P = 0)	100
PE — parity even (P = 1)	101
P — plus (S = 0)	110
M — minus (S = 1)	111

JMP addr (Jump)
(PC) — (byte 3) (byte 2)
Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



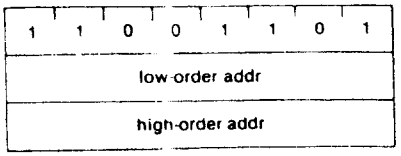
Cycles: 3
States: 10
Addressing: immediate
Flags: none

Jcondition addr (Conditional jump)
If (CCC),
(PC) — (byte 3) (byte 2)
If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruciton; otherwise, control continues sequentially.



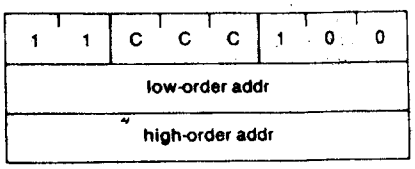
Cycles: 2/3
States: 7/10
Addressing: immediate
Flags: none

CALL addr (Call)
((SP) − 1) — (PCH)
((SP) − 2) — (PCL)
(SP) — (SP) − 2
(PC) — (byte 3) (byte 2)
The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



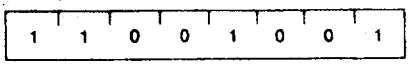
Cycles: 5
States: 18
Addressing: immediate/ reg. indirect
Flags: none

Ccondition addr (Condition call)
If (CCC),
((SP) − 1) — (PCH)
((SP) − 2) — (PCL)
(SP) — (SP) − 2
(PC) — (byte 3) (byte 2)
If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.



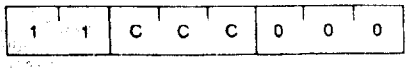
Cycles: 2/5
States: 9/18
Addressing: immediate/ reg. indirect
Flags: none

RET (Return)
(PCL) — ((SP));
(PCH) — ((SP) + 1);
(SP) — (SP) + 2;
The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

Rcondition (Conditional return)
If (CCC),
(PCL) — ((SP))
(PCH) — ((SP) + 1)
(SP) — (SP) + 2
If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.



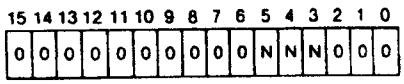
Cycles: 1/3
States: 6/12
Addressing: reg. indirect
Flags: none

RST n (Restart)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← 8 * (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

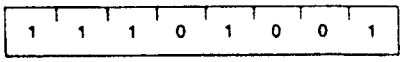


Program Counter After Restart

PCHL (Jump H and L indirect — move H and L to PC)

(PCH) ← (H)
(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 1
States: 6
Addressing: register
Flags: none

4.6.5 Stack, I/O, and Machine Control Group

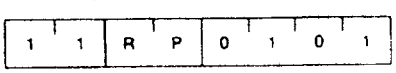
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, condition flags are not affected by any instructions in this group.

PUSH rp (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
((SP) ← (SP) − 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**



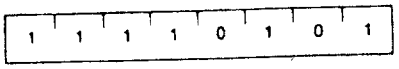
Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

((SP) − 1) ← (A)
$((SP) - 2)_0$ ← (CY), $((SP) - 2)_1$ ← X
$((SP) - 2)_2$ ← (P), $((SP) - 2)_3$ ← X
$((SP) - 2)_4$ ← (AC), $((SP) - 2)_5$ ← X
$((SP) - 2)_6$ ← (Z), $((SP) - 2)_7$ ← (S)
(SP) ← (SP) − 2

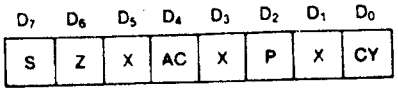
X: Undefined

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.



Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

FLAG WORD

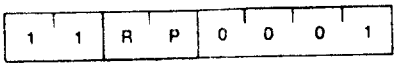


X: undefined

POP rp (POP)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**

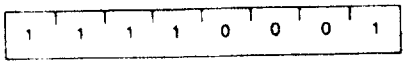


Cycles: 3
States: 10
Addressing: reg.indirect
Flags: none

POP PSW (Pop processor status word)

(CY) ← $((SP))_0$
(P) ← $((SP))_2$
(AC) ← $((SP))_4$
(Z) ← $((SP))_6$
(S) ← $((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

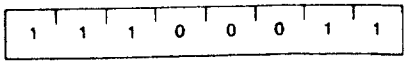


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XTHL (Exchange stack top with H and L)

(L) ← ((SP))
(H) ← ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

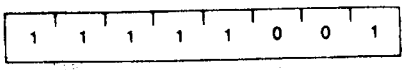


Cycles: 5
States: 16
Addressing: reg. indirect
Flags: none

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

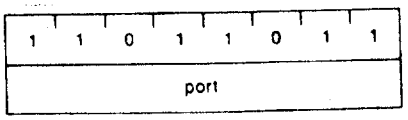


Cycles: 1
States: 6
Addressing: register
Flags: none

IN port (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

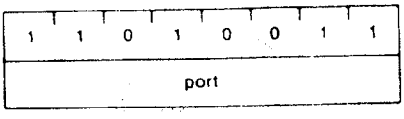


Cycles: 3
States: 10
Addressing: direct
Flags: none

OUT port (Output)

(data) ← (A)

The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.



Cycles: 3
States: 10
Addressing: direct
Flags: none

EI (Enable interrupts)

The interrupt system is enabled **following the execution of the next instruction.**

1	1	1	1	1	0	1	1

Cycles: 1
States: 4
Flags: none

NOTE: Interrupts are not recognized during the EI instruction. Placing an EI instruction on the bus in response to $\overline{INTA}$ during an INA cycle is prohibited.

DI (Disable interrupts)

The interupt system is disabled **immediately following the execution of the DI instruction.**

1	1	1	1	0	0	1	1

Cycles: 1
States: 4
Flags: none

NOTE: Interrupts are not recognized during the DI instruction. Placing a DI instruction on the bus in response to $\overline{INTA}$ during an INA cycle is prohibited.

HLT (Halt)

The processor is stopped. The registers and flags are unaffected. A second ALE is generated during the execution of HLT to strobe out the Halt cycle status information.

0	1	1	1	0	1	1	0

Cycles: 1+
States: 5
Flags: none

NOP (No op)

No operation is performed. The registers and flags are unaffected.

0	0	0	0	0	0	0	0

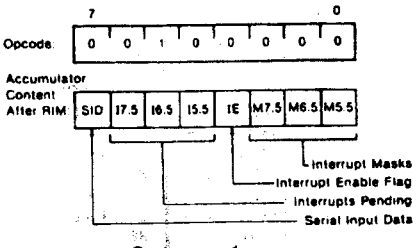
Cycles: 1
States: 4
Flags: none

RIM (Read Interrupt Masks)

The RIM instruction loads data into the accumulator relating to interrupts and the serial input. This data contains the following information:

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware interrupts (1 = mask disabled)
- Current interrupt enable flag status (1 = interrupts enabled) except immediately following a TRAP interrupt. (See below.)
- Hardware interrupts pending (i.e., signal received but not yet serviced), on the RST 5.5, 6.5, and 7.5 lines.
- Serial input data.

Immediately following a TRAP interrupt, the RIM instruction must be executed as a part of the service routine if you need to retrieve current interrupt status later. Bit 3 of the accumulator is (in this special case only) loaded with the interrupt enable (IE) flag status that existed prior to the TRAP interrupt. Following an RST 5.5, 6.5, 7.5, or INTR interrupt, the interrupt flag flip-flop reflects the current interrupt enable status. Bit 6 of the accumulator (I7.5) is loaded with the status of the RST 7.5 flip-flop, which is always set (edge-triggered) by an input on the RST 7.5 input line, even when that interrupt has been previously masked. (See SIM Instruction.)



Cycles: 1
States: 4
Flags: none

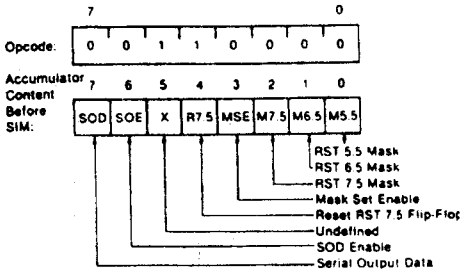
SIM (Set Interrupt Masks)

The execution of the SIM instruction uses the contents of the accumulator (which must be previously loaded) to perform the following functions:

- Program the interrupt mask for the RST 5.5, 6.5, and 7.5 hardware interrupts.
- Reset the edge-triggered RST 7.5 input latch.
- Load the SOD output latch.

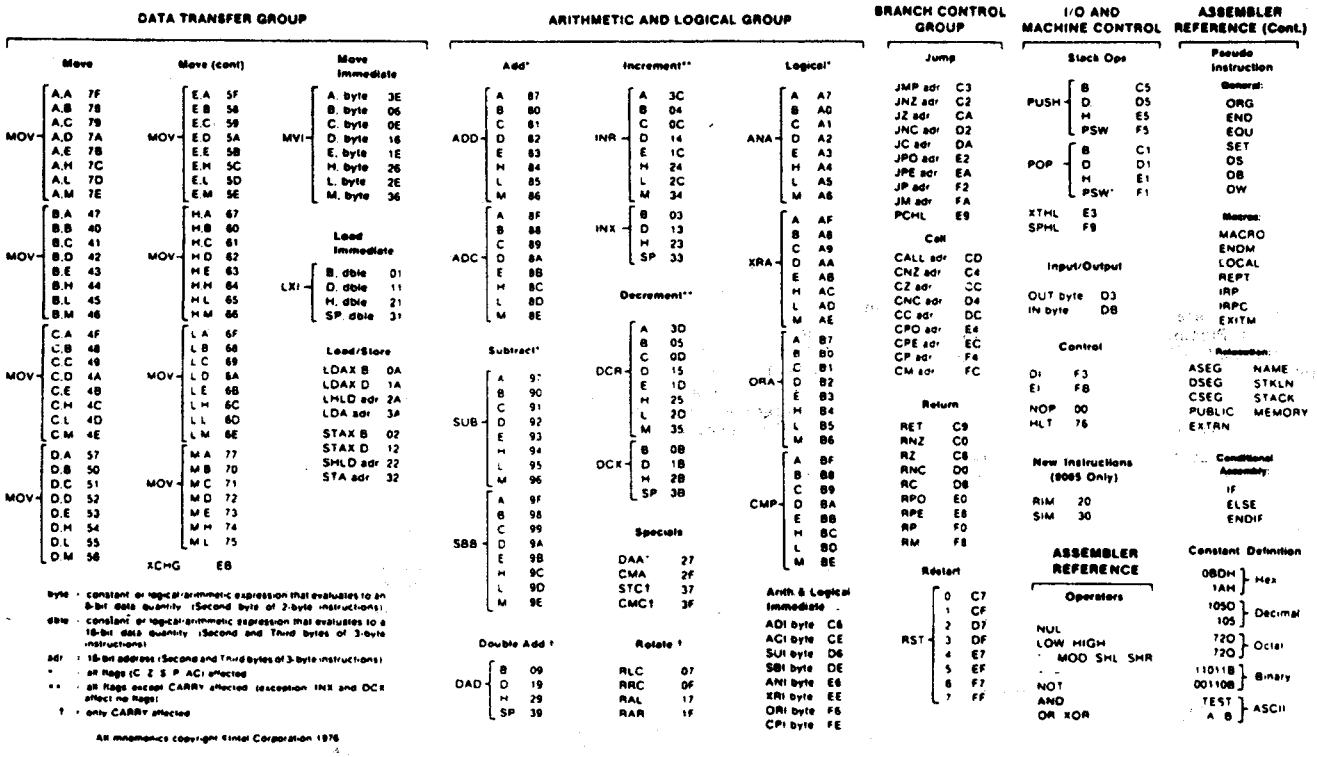
To program the interrupt masks, first set accumulator bit 3 to 1 and set to 1 any bits 0, 1, and 2, which disable interrupts RST 5.5, 6.5, and 7.5, respectively. Then do a SIM instruction. If accumulator bit 3 is 0 when the SIM instruction is executed, the interrupt mask register will not change. If accumulator bit 4 is 1 when the SIM instruction is executed, the RST 7.5 latch is then reset. RST 7.5 is distinguished by the fact that its latch is always set by a rising edge on the RST 7.5 input pin, even if the jump to service routine is inhibited by masking. This latch remains high until cleared by a $\overline{RESET\ IN}$, by a SIM instruction with accumulator bit 4 high, or by an internal processor acknowledge to an RST 7.5 interrupt subsequent to the removal of the mask (by a SIM instruction). The $\overline{RESET\ IN}$ signal always sets all three RST mask bits.

If accumulator bit 6 is at the 1 level when the SIM instruction is executed, the state of accumulator bit 7 is loaded into the SOD latch and thus becomes available for interface to an external device. The SOD latch is unaffected by the SIM instruction if bit 6 is 0. SOD is always reset by the $\overline{RESET\ IN}$ signal.



Cycles: 1
States: 4
Flags: none

Kad Rujukan Bahasa Penghimpunan 8085



8085A CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	28	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	–
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	–
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	–	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	–	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	–	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	–
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	–	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	–	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	–
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	–	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

Adr = 16-bit address.

ASCII Code Table

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	D	C	E	F
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p	Ç	É	á				∝	≡
1	1	☺	◄	!	1	A	Q	a	q	ü	Æ	í				β	±
2	2	☻	↕	"	2	B	R	b	r	é	FE	ó				γ	≥
3	3	♥	‼	#	3	C	S	c	s	â	ô	ú				π	≤
4	4	♦	¶	$	4	D	T	d	t	ä	ö	ñ				Σ	⌠
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ				σ	⌡
6	6	♠	▬	&	6	F	V	f	v	å	û	a				μ	÷
7	7	•	↨	’	7	G	W	g	w	ç	ù	o				τ	≈
8	8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿				Φ	°
9	9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐				Θ	•
10	A	◙	→	*	:	J	Z	j	x	è	Ü	¬				Ω	•
11	B	♂	←	+	;	K	[	k	{	ï	¢	½				δ	√
12	C	♀	∟	,	<	L	\	l	¦	î	£	¼				∞	η
13	D	♪	↔	—	=	M	]	m	}	ì	¥	¡				∅	2
14	E	♫	▲	.	>	N	^	n	~	Ä	Pts	«				∈	■
15	F	☼	▼	/	?	O	_	o	⌂	Å	f	»				∩	BLANK 'FF'