UNIVERSITI SAINS MALAYSIA

Peperiksaan Semester Pertama
Sidang Akademik 1997/98

September 1997

ZAT 381/4 - Pengantar Mikropemproses

Masa: [3 jam]

---

Sila pastikan bahawa kertas peperiksaan ini mengandungi TUJUH muka surat yang bercetak sebelum anda memulakan peperiksaan ini. Satu set jadual 21 muka surat juga dilampirkan.

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

1. (a) Berikan perbezaan di antara bahasa-bahasa berikut:

(i) Bahasa mesin dan bahasa penghimpunan.
(ii) Bahasa paras tinggi dan bahasa paras rendah.

(20/100)

(b) Tuliskan aturcara penghimpunan bagi menyelesaikan aritmatik berikut:

244 + 300 + 120 - 321

dan simpankan jawapannya di lokasi ingatan mulai 20E2H.

(40/100)

(c) Tuliskan aturcara penghimpunan bagi mendapatkan nombor integer terbesar daripada senarai yang tersimpan di ingatan 20A0H hingga 20A9H dan simpan nombor tersebut di ingatan 20B0H.

(40/100)

2. (a) Berikan dua kegunaan utama timbunan (stack).

(20/100)

.../2-

(b) Fahami aturcara di bawah kemudian jawab soalan yang berikutnya:

```
2000H   LXI SP, 2100H   ;Tetapkan kedudukan penunjuk timbunan
2003H   LXI B, 0000H    ;Kosongkan alat daftar B dan C
2006H   PUSH B          ;Simpan kandungan alat daftar B dan C
2007H   POP PSW         ;Baca nilai perkataan status aturcara
2008H   LXI H, 200BH    ;Berikan nilai alat daftar H dan L
200BH   CALL 2064H      ;Panggil subrutin
200EH   OUT 01H         ;Keluarkan kandungan akumulator ke pot
                        ;no. 1
2010H   HLT             ;Berhenti pemprosesan
.
.
2064H   PUSH H          ;Selamatkan kandungan alat daftar HL
2065H   PUSH B          ;Selamatkan kandungan alat daftar BC
2066H   LXI B, 00FFH    ;Muatkan alat daftar BC dengan
                        ;kandungan baru
2069H   DCX B           ;Kurangkan nilai alat daftar BC
206AH   MOV A, B        ;Pindahkan kandungan B kepada A
206BH   ORA C           ;Ataukan kandungan C dengan
                        ;akumulator
206CH   JNZ 2069H       ;Ulang arahan di lokasi 2069H jika
                        ;bendera sifar reset
206FH   POP B           ;Ambil kandungan asal alat daftar BC
        POP HL          ;Ambil kandungan asal HL
2070H   RET             ;Kembali ke aturcara utama
```

(i) Apakah status bendera dan kandungan akumulator selepas perlaksanaan arahan POP di lokasi 2007H?

(ii) Nyatakan kedudukan lokasi stack dan kandungannya selepas perlaksanaan arahan CALL di lokasi ingatan 200BH.

(iii) Apakah kandungan alatdaftar penunjuk stack dan penunjuk program selepas perlaksanaan arahan CALL tersebut?

.../3-

(iv) Nyatakan lokasi ingatan di mana penunjuk program kembali selepas perlaksanaan subrutin.

(v) Nyatakan fungsi keseluruhan aturcara.

(50/100)

(c) Kirakan masa yang diambil oleh sistem mikropemproses untuk melaksanakan subrutin di alamat 2064H hingga 2070H.

(30/100)

3. Aturcara ujian berikut digunakan untuk menguji antaramuka di antara mikropemproses 8085 berfrekuensi jam 2.5 MHz. dengan pot output 23H:

```
20A0H    MVI A, 2AH
20A2H    OUT 23H
20A4H    JMP 20A2H
```

(a) Tentukan dasar masa osiloskop yang sesuai supaya suatu gelombang pegun terhasil daripada aturcara ujian tersebut.

(20/100)

(b) Lakarkan bentuk gelombang pegun yang mungkin diperhatikan dalam helaian gambarajah pemasa yang dilampirkan (LAMPIRAN A), sekiranya pin-pin yang dinyatakan itu disambung kepada penduga (probe) osiloskop. Serta berikan kandungan bas alamat dan bas data yang sepadan.

(50/100)

(c) Berapa kalikah isyarat ($\overline{RD}$) diperlukan bagi setiap kitaran pegun tersebut?

(15/100)

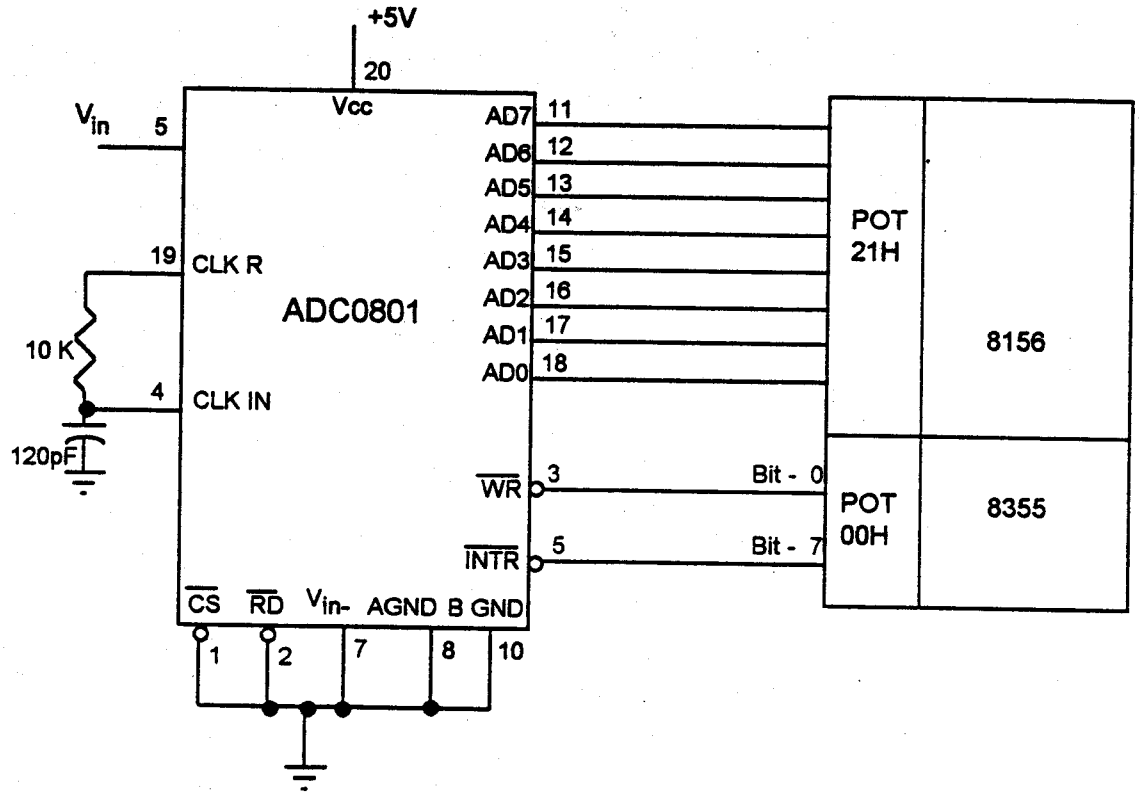
(d) Terangkan maksud pertindihan ambil laksana (fetch-execute overlap) yang berlaku dalam sebahagian arahan 8085. Berikan dua contoh arahan yang melibatkan pertindihan tersebut.

(15/100)

4. Rajah 1 menunjukkan suatu sistem penukar analog ke digital secara berjabat tangan perisian yang berasaskan mikropemproses 8085. Penukaran dimulakan dengan menghantar isyarat mula penukaran kepada input $\overline{WR}$ . Kemudian isyarat tamat penukaran $\overline{INTR}$ perlu dibaca yang menyatakan data telah tersedia. Data digital kemudiannya dibaca dan disimpan di dalam ingatan RAM. Bahagian aturcara berikut membolehkan operasi tersebut:

```
2000H   LXI DE 4000     ;Setkan bilangan data

2003H   MVI A, 01H      ;Set bit-0 (sebagai output)
2005H   OUT 02H         ;Sediakan pot-0H
2007H   MVI A, 00H      ;Reset bit-0 (pot 21 sebagai input)
2009H   OUT 20H         ;Sediakan pot-21H

200BH   MVI A, 00H      ;Reset bit-0
200DH   OUT 00H         ;Hantar isyarat rendah mula penukaran
200FH   MVI A, 01H      ;Set bit-0
2011H   OUT 00H         ;Hantar isyarat tinggi mula penukaran

2013H   IN 00H          ;mula gelung
2015H   ANI 80H         ;Topengkan kecuali bit-7
2017H   JNZ 2013H       ;Gelung jika isyarat akhir penukaran tinggi

201AH   IN 21H          ;Inputkan data yang telah ditukarkan
201CH   STAX D          ;Simpan kandungan A
201DH   INX D           ;Kurangkan kandungan alat daftar DE
201EH   MOV A, D        ;Pindahkan kandungan alat daftar D ke
                        ;dalam akumulator
201FH   CPI FFH         ;Bandingkan kandungan akumulator
                        ;dengan FFH
2021H   JNZ 200BH       ;Ambil data berikutnya
2024H   HLT             ;Selesai pemprosesan data
```

.../5-




Rajah 1

(a) Terangkan fungsi keseluruhan arahan-arahan di alamat 2003H - 2009H dan terangkan fungsi setiap arahan tersebut.

(20/100)

(b) Terangkan fungsi keseluruhan arahan-arahan di alamat berikut:

(i) 200BH - 2011H
(ii) 2013H - 2017H
(iii) 201AH - 2021H

(30/100)

(c) Berapa banyakkah data analog yang ditukar kepada digital oleh sistem mikropemproses tersebut?

(10/100)

.../6-

(d) Penukaran analog ke digital tersebut boleh juga dilakukan secara berjabat tangan perkakasan.

(i) Lukiskan litar berasakan ADC0801 serta get-get logik yang bersesuaian bagi menukarkan data analog kepada digital. Isyarat sampuk RST7.5 hendaklah digunakan bagi membolehkan data dibaca melalui pot FFH dan disimpan di ingataan RAM.

(ii) Tuliskan aturcara yang sesuai bagi membolehkan perkakasan tersebut beroperasi dan data dibaca dan disimpan sama seperti yang dilaksanakan oleh sistem berjabat tangan perisian.

(40/100)

5. (a) Terangkan maksud ingatan berlipat di dalam pengantaramukaan mikropemproses dan cip ingatan.

(10/100)

(b) Berdasarkan litar sistem mikropemproses yang ditunjukkan di Rajah 2, tentukan alamat-alamat berikut:

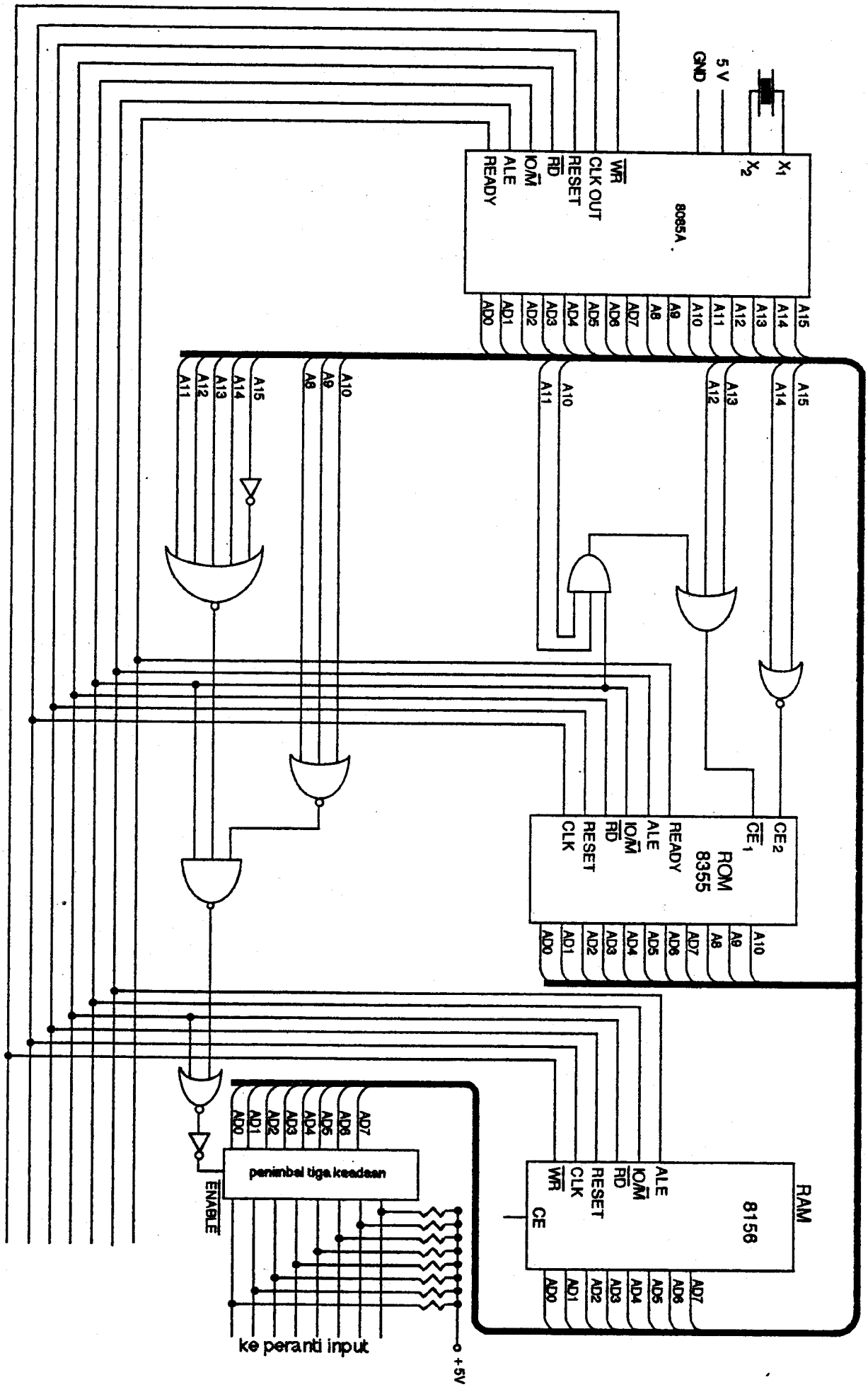
(i) ROM
(ii) Nombor pot-pot dan alat daftar kawalan 8355
(iii) Nombor pot penimbal

(50/100)

(c) Lakarkan litar nyahkod garis alamat yang sesuai bagi mengantara mukakan cip 8156 kepada sistem mikropemproses di Rajah 2 dengan ingatan RAM pada alamat 4000H - 40FFH dan tidak berlipat. Nyatakan alamat pot-pot dan pemasa daripada cip tersebut.

(40/100)

- ooo0ooo -


5 V
GND
X1
X2
8085A
WR
CLK OUT
RESET
RD
IO/M
ALE
READY
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A8
A9
A10
A11
A12
A13
A14
A15
ROM
8355
CE2
CE1
CLK
RESET
RD
IO/M
ALE
READY
penimbal tiga keadaan
ENABLE
ke peranti input
+5V
RAM
8156
CE
WR
CLK
RESET
RD
IO/M
ALE

Rajah 2


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

# 8085 Instruction Set

## 4.8 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.

### 4.8.1 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.

| 0 | 1 | D | D | D | S | S | S |
|---|---|---|---|---|---|---|---|

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.

| 0 | 1 | D | D | D | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 1 | 1 | 1 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | D | D | D | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |

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.

| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |

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.

| 0 | 0 | R | P | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| low-order data | | | | | | | |
| high-order data | | | | | | | |

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.

| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |

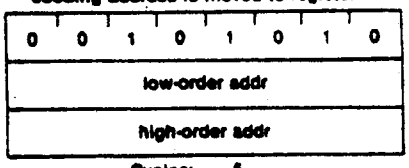
Cycles: 4
States: 13
Addressing: direct
Flags: none

**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.

| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |

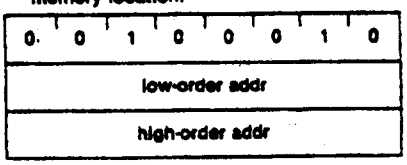
Cycles: 4
States: 13
Addressing: direct
Flags: none

**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.



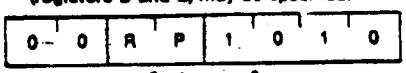

Cycles: 5
States: 16
Addressing: direct
Flags: none

**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.

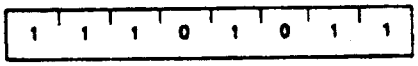



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

**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

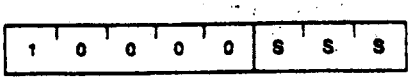
### 4.8.2 Arithmetic Group

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.**

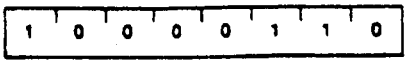
All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

**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.



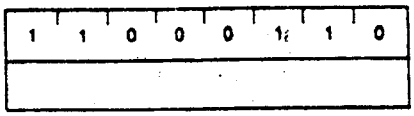

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**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.



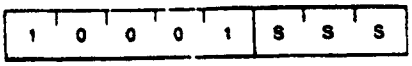

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**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.



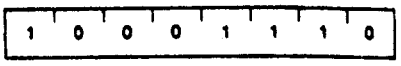

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**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.



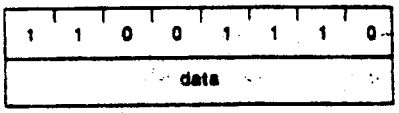

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**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.



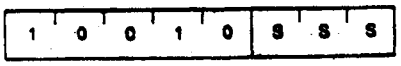

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**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.



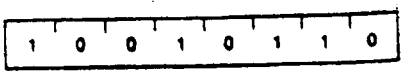

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**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.



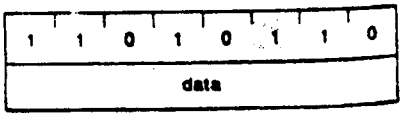

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**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.

| 1 | 0 | 0 | 1 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|

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.

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

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.

| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |

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.

| 0 | 0 | D | D | D | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | D | D | D | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | R | P | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | R | P | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | R | P | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

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.

| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

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

### 4.8.3 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.

| 1 | 0 | 1 | 0 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|

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.

| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

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.

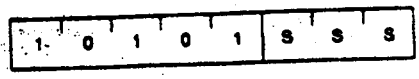
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |

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.

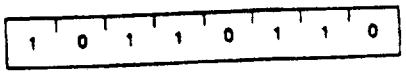



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**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.

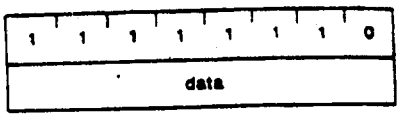



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**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).

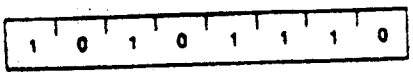



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**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.

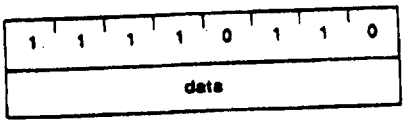



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**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.



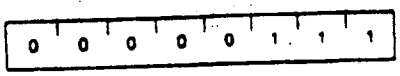

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**RLC** (Rotate left)

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

$(CY) \leftarrow (A_7)$

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.

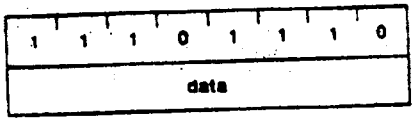



Cycles: 1
States: 4
Flags: CY

**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.

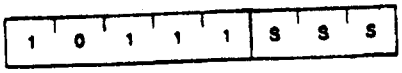



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**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).



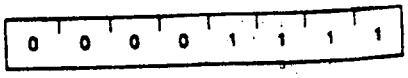

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**RRC** (Rotate right)

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

$(CY) \leftarrow (A_0)$

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.

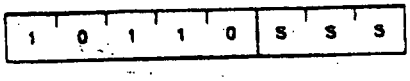



Cycles: 1
States: 4
Flags: CY

**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.

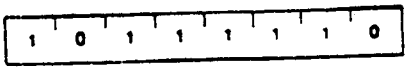



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**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)).



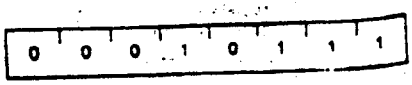

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**RAL** (Rotate left through carry)

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

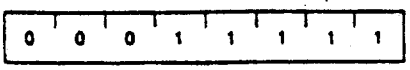
$(A_0) \leftarrow (CY)$

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.



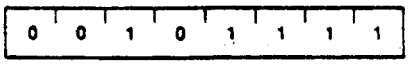

Cycles: 1
States: 4
Flags: CY

**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.



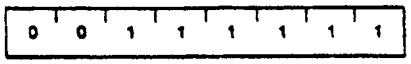

Cycles: 1
States: 4
Flags: CY

**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.



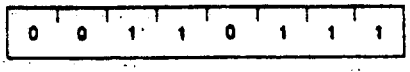

Cycles: 1
States: 4
Flags: none

**CMC** (Complement carry)
$(CY) \leftarrow (\overline{CY})$
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

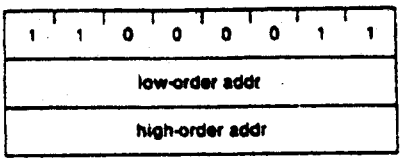
This group of instructions alter normal sequential program flow.

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:

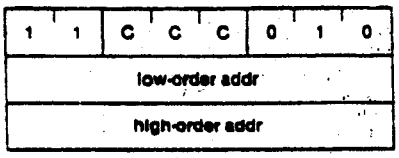
| CONDITION | CCC |
|---|---|
| NZ — not zero (Z = 0) | 000 |
| Z — zero (Z = 1) | 001 |
| NC — no carry (CY = 0) | 010 |
| C — carry (CY = 1) | 011 |
| PO — parity odd (P = 0) | 100 |
| PE — parity even (P = 1) | 101 |
| P — plus (S = 0) | 110 |
| M — minus (S = 1) | 111 |

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



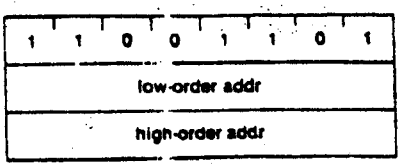

Cycles: 3
States: 10
Addressing: immediate
Flags: none

**Jcondition addr** (Conditional jump)
If (CCC),
$(PC) \leftarrow$ (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.



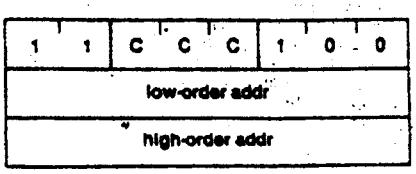

Cycles: 2/3
States: 7/10
Addressing: immediate
Flags: none

**CALL addr** (Call)
$((SP) - 1) \leftarrow (PCH)$
$((SP) - 2) \leftarrow (PCL)$
$(SP) \leftarrow (SP) - 2$
$(PC) \leftarrow$ (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.



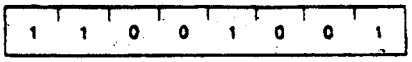

Cycles: 5
States: 18
Addressing: immediate/ reg. indirect
Flags: none

**Ccondition addr** (Condition call)
If (CCC),
$((SP) - 1) \leftarrow (PCH)$
$((SP) - 2) \leftarrow (PCL)$
$(SP) \leftarrow (SP) - 2$
$(PC) \leftarrow$ (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)
$(PCL) \leftarrow ((SP))$;
$(PCH) \leftarrow ((SP) + 1)$;
$(SP) \leftarrow (SP) + 2$;
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),
$(PCL) \leftarrow ((SP))$
$(PCH) \leftarrow ((SP) + 1)$
$(SP) \leftarrow (SP) + 2$
If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

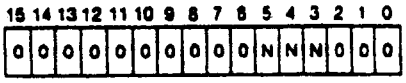

Cycles: 1/3
States: 5/12
Addressing: reg. indirect
Flags: none

**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

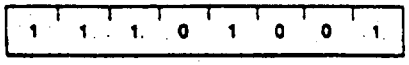



Program Counter After Restart

**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.

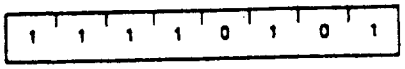

Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

**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

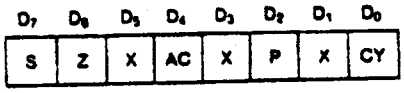
X: Undefined.

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

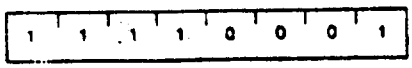
The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register rp. The 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.

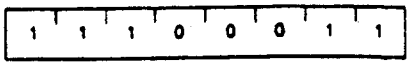



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**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.

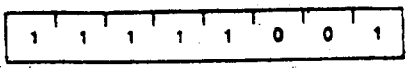



Cycles: 5
States: 16
Addressing: reg. indirect
Flags: none

**SPHL** (Move HL to SP)

(SP) ← (H) (L)

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

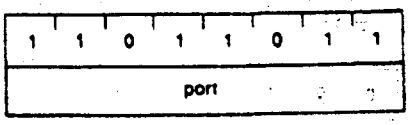



Cycles: 1
States: 6
Addressing: register
Flags: none

**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.

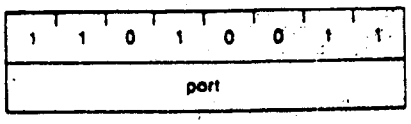



Cycles: 3
States: 10
Addressing: direct
Flags: none

**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 |
|---|---|---|---|---|---|---|---|

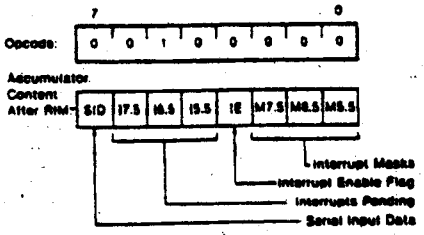
Cycles: 1
States: 4
Flags: none

**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

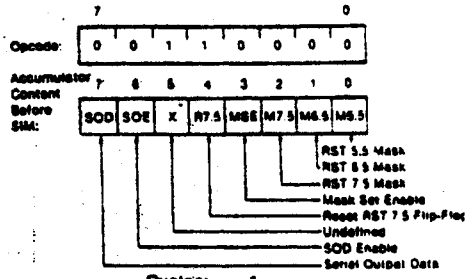
**SIM** (Set Interrupt Masks)

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.

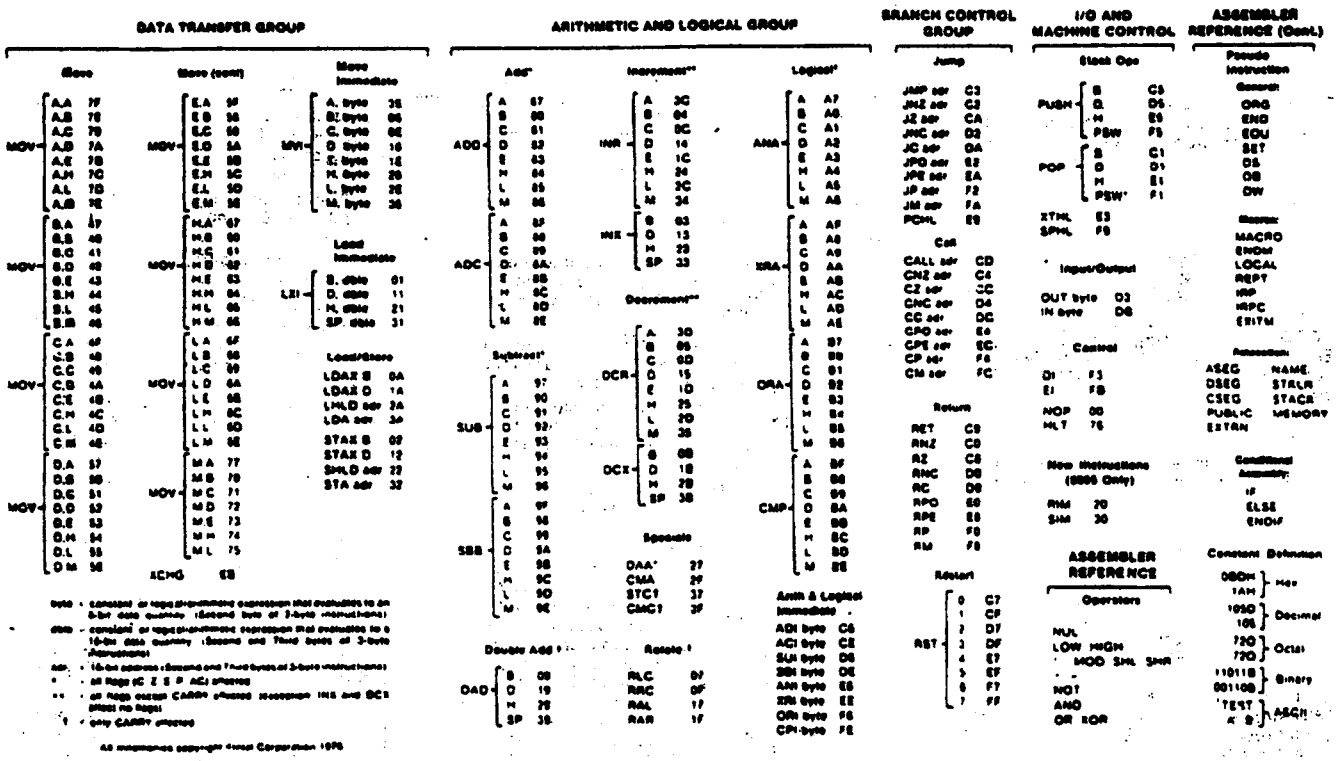
To program the interrupt masks, first set accumulator bit 3 to 1 and set to 1 any bits 0, 1, and 2, which disable interrupts RST 5.5, 6.5, and 7.5, respectively. Then do a SIM instruction. If accumulator bit 3 is 0 when the SIM instruction is executed, the interrupt mask register will not change. If 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

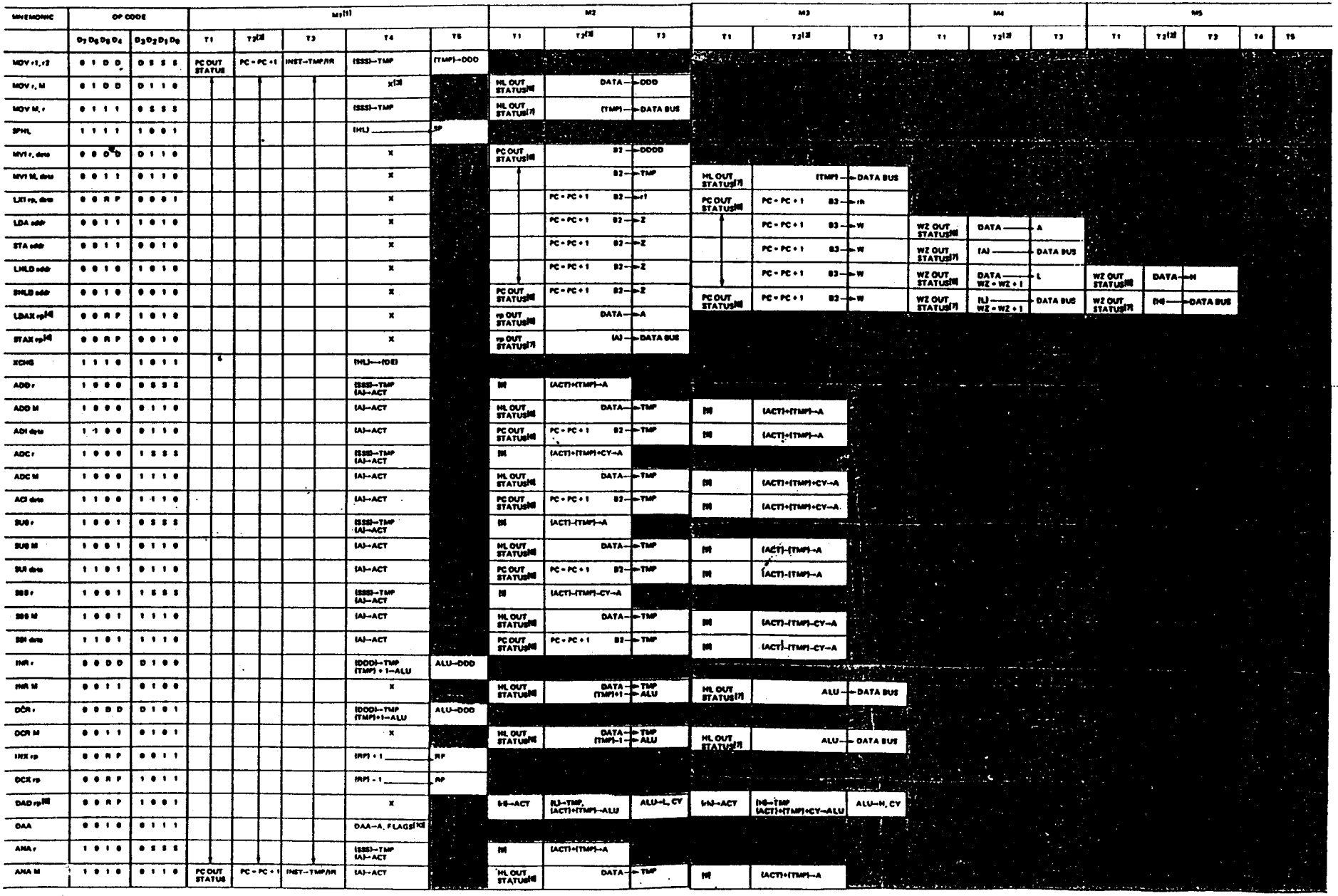
| OP CODE | MNEMONIC | OP CODE | MNEMONIC | OP CODE | MNEMONIC | OP CODE | MNEMONIC | OP CODE | MNEMONIC | OP CODE | MNEMONIC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | NOP | 28 | DCX H | 56 | MOV D,M | 81 | ADD C | AC | XRA H | D7 | RST 2 |
| 01 | LXI B,D16 | 2C | INR L | 57 | MOV D,A | 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.

# 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 |

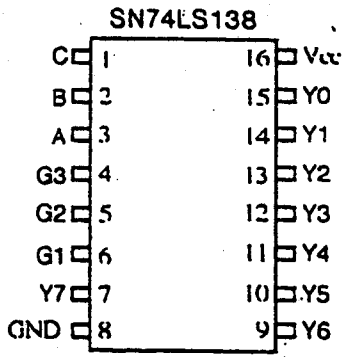


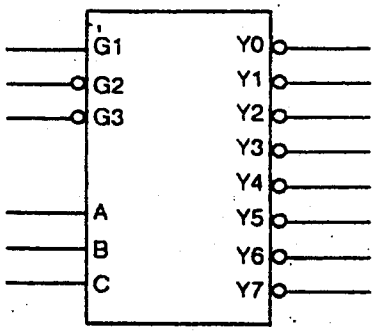
| MNEMONIC | OP CODE | | M1[1] | | | | | M2 | | | M3 | | | M4 | | | M5 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $D_7\ D_6\ D_5\ D_4$ | $D_3\ D_2\ D_1\ D_0$ | T1 | T2[2] | T3 | T4 | T5 | T1 | T2[2] | T3 | T1 | T2[2] | T3 | T1 | T2[2] | T3 | T1 | T2[2] | T3 | T4 | T5 |
| MOV r1, r2 | 0 1 D D | D S S S | PC OUT STATUS | PC ← PC + 1 | INST→TMP/IR | (SSS)→TMP | (TMP)→DDD | | | | | | | | | | | | | | |
| MOV r, M | 0 1 D D | D 1 1 0 | | | | X[3] | | HL OUT STATUS[6] | | DATA → DDD | | | | | | | | | | | |
| MOV M, r | 0 1 1 1 | 0 S S S | | | | (SSS)→TMP | | HL OUT STATUS[7] | | (TMP) → DATA BUS | | | | | | | | | | | |
| SPHL | 1 1 1 1 | 1 0 0 1 | | | | (HL) → SP | | | | | | | | | | | | | | | |
| MVI r, data | 0 0 D D | D 1 1 0 | | | | X | | PC OUT STATUS[6] | | B2 → DDD | | | | | | | | | | | |
| MVI M, data | 0 0 1 1 | 0 1 1 0 | | | | X | | | | B2 → TMP | HL OUT STATUS[7] | | (TMP) → DATA BUS | | | | | | | | |
| LXI rp, data | 0 0 R P | 0 0 0 1 | | | | X | | | PC ← PC + 1 | B2 → rl | PC OUT STATUS[6] | PC ← PC + 1 | B3 → rh | | | | | | | | |
| LDA addr | 0 0 1 1 | 1 0 1 0 | | | | X | | | PC ← PC + 1 | B2 → Z | | PC ← PC + 1 | B3 → W | WZ OUT STATUS[6] | DATA → A | | | | | | |
| STA addr | 0 0 1 1 | 0 0 1 0 | | | | X | | | PC ← PC + 1 | B2 → Z | | PC ← PC + 1 | B3 → W | WZ OUT STATUS[7] | (A) → DATA BUS | | | | | | |
| LHLD addr | 0 0 1 0 | 1 0 1 0 | | | | X | | | PC ← PC + 1 | B2 → Z | | PC ← PC + 1 | B3 → W | WZ OUT STATUS[6] | DATA → L; WZ ← WZ + 1 | | WZ OUT STATUS[6] | DATA → H | | | |
| SHLD addr | 0 0 1 0 | 0 0 1 0 | | | | X | | PC OUT STATUS[6] | PC ← PC + 1 | B2 → Z | PC OUT STATUS[6] | PC ← PC + 1 | B3 → W | WZ OUT STATUS[7] | (L) → DATA BUS; WZ ← WZ + 1 | | WZ OUT STATUS[7] | (H) → DATA BUS | | | |
| LDAX rp[4] | 0 0 R P | 1 0 1 0 | | | | X | | rp OUT STATUS[6] | | DATA → A | | | | | | | | | | | |
| STAX rp[4] | 0 0 R P | 0 0 1 0 | | | | X | | rp OUT STATUS[7] | | (A) → DATA BUS | | | | | | | | | | | |
| XCHG | 1 1 1 0 | 1 0 1 1 | | | | (HL)↔(DE) | | | | | | | | | | | | | | | |
| ADD r | 1 0 0 0 | 0 S S S | | | | (SSS)→TMP; (A)→ACT | | [8] | | (ACT)+(TMP)→A | | | | | | | | | | | |
| ADD M | 1 0 0 0 | 0 1 1 0 | | | | (A)→ACT | | HL OUT STATUS[6] | | DATA → TMP | [8] | | (ACT)+(TMP)→A | | | | | | | | |
| ADI data | 1 1 0 0 | 0 1 1 0 | | | | (A)→ACT | | PC OUT STATUS[6] | PC ← PC + 1 | B2 → TMP | [8] | | (ACT)+(TMP)→A | | | | | | | | |
| ADC r | 1 0 0 0 | 1 S S S | | | | (SSS)→TMP; (A)→ACT | | [8] | | (ACT)+(TMP)+CY→A | | | | | | | | | | | |
| ADC M | 1 0 0 0 | 1 1 1 0 | | | | (A)→ACT | | HL OUT STATUS[6] | | DATA → TMP | [8] | | (ACT)+(TMP)+CY→A | | | | | | | | |
| ACI data | 1 1 0 0 | 1 1 1 0 | | | | (A)→ACT | | PC OUT STATUS[6] | PC ← PC + 1 | B2 → TMP | [8] | | (ACT)+(TMP)+CY→A | | | | | | | | |
| SUB r | 1 0 0 1 | 0 S S S | | | | (SSS)→TMP; (A)→ACT | | [8] | | (ACT)−(TMP)→A | | | | | | | | | | | |
| SUB M | 1 0 0 1 | 0 1 1 0 | | | | (A)→ACT | | HL OUT STATUS[6] | | DATA → TMP | [8] | | (ACT)−(TMP)→A | | | | | | | | |
| SUI data | 1 1 0 1 | 0 1 1 0 | | | | (A)→ACT | | PC OUT STATUS[6] | PC ← PC + 1 | B2 → TMP | [8] | | (ACT)−(TMP)→A | | | | | | | | |
| SBB r | 1 0 0 1 | 1 S S S | | | | (SSS)→TMP; (A)→ACT | | [8] | | (ACT)−(TMP)−CY→A | | | | | | | | | | | |
| SBB M | 1 0 0 1 | 1 1 1 0 | | | | (A)→ACT | | HL OUT STATUS[6] | | DATA → TMP | [8] | | (ACT)−(TMP)−CY→A | | | | | | | | |
| SBI data | 1 1 0 1 | 1 1 1 0 | | | | (A)→ACT | | PC OUT STATUS[6] | PC ← PC + 1 | B2 → TMP | [8] | | (ACT)−(TMP)−CY→A | | | | | | | | |
| INR r | 0 0 D D | D 1 0 0 | | | | (DDD)→TMP; (TMP) + 1→ALU | ALU→DDD | | | | | | | | | | | | | | |
| INR M | 0 0 1 1 | 0 1 0 0 | | | | X | | HL OUT STATUS[6] | | DATA → TMP; (TMP)+1 → ALU | HL OUT STATUS[7] | | ALU → DATA BUS | | | | | | | | |
| DCR r | 0 0 D D | D 1 0 1 | | | | (DDD)→TMP; (TMP)+1→ALU | ALU→DDD | | | | | | | | | | | | | | |
| DCR M | 0 0 1 1 | 0 1 0 1 | | | | X | | HL OUT STATUS[6] | | DATA → TMP; (TMP)−1 → ALU | HL OUT STATUS[7] | | ALU → DATA BUS | | | | | | | | |
| INX rp | 0 0 R P | 0 0 1 1 | | | | (RP) + 1 → RP | | | | | | | | | | | | | | | |
| DCX rp | 0 0 R P | 1 0 1 1 | | | | (RP) − 1 → RP | | | | | | | | | | | | | | | |
| DAD rp[5] | 0 0 R P | 1 0 0 1 | | | | X | | 5-8→ACT | (L)→TMP; (ACT)+(TMP)→ALU | ALU→L, CY | 5-8→ACT | (H)→TMP; (ACT)+(TMP)+CY→ALU | ALU→H, CY | | | | | | | | |
| DAA | 0 0 1 0 | 0 1 1 1 | | | | DAA→A, FLAGS[10] | | | | | | | | | | | | | | | |
| ANA r | 1 0 1 0 | 0 S S S | | | | (SSS)→TMP; (A)→ACT | | [8] | | (ACT)+(TMP)→A | | | | | | | | | | | |
| ANA M | 1 0 1 0 | 0 1 1 0 | PC OUT STATUS | PC ← PC + 1 | INST→TMP/IR | (A)→ACT | | HL OUT STATUS[6] | | DATA → TMP | [8] | | (ACT)+(TMP)→A | | | | | | | | |

| MNEMONIC | OP CODE | | M1 | | | | M2 | | | M3 | | | M4 | | | M5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $D_7D_6D_5D_4$ | $D_3D_2D_1D_0$ | T1 | T2/3 | T3 | T4 | T5 | T1 | T2/3 | T3 | T1 | T2/3 | T3 | T1 | T2/3 | T3 | T4 | T5 |
| ANI data | 1 1 1 0 | 0 1 1 0 | PC OUT STATUS | PC ← PC + 1 | INST←TMP/IR | (ALU)→ACT | PC OUT STATUS[M] | PC ← PC + 1 | B2→TMP | MR | (ACT)+(TMP)→A | | | | | | | | |
| XRA r | 1 0 1 0 | 1 S S S | | | | (ALU)→ACT SSS→TMP | | | | | | | | | | | | | |
| XRA M | 1 0 1 0 | 1 1 1 0 | | | | (ALU)→ACT | | DATA→TMP | | MR | (ACT)+(TMP)→A | | | | | | | | |
| XRI data | 1 1 1 0 | 1 1 1 0 | | | | (ALU)→ACT | | PC ← PC + 1 | B2→TMP | MR | (ACT)+(TMP)→A | | | | | | | | |
| ORA r | 1 0 1 1 | 0 S S S | | | | (ALU)→ACT SSS→TMP | | | | | | | | | | | | | |
| ORA M | 1 0 1 1 | 0 1 1 0 | | | | (ALU)→ACT | | DATA→TMP | | MR | (ACT)+(TMP)→A | | | | | | | | |
| ORI data | 1 1 1 1 | 0 1 1 0 | | | | (ALU)→ACT | | PC ← PC + 1 | B2→TMP | MR | (ACT)+(TMP)→A | | | | | | | | |
| CMP r | 1 0 1 1 | 1 S S S | | | | (ALU)→ACT SSS→TMP | | | | | | | | | | | | | |
| CMP M | 1 0 1 1 | 1 1 1 0 | | | | (ALU)→ACT | | DATA→TMP | | MR | (ACT)-(TMP); FLAGS | | | | | | | | |
| CPI data | 1 1 1 1 | 1 1 1 0 | | | | (ALU)→ACT | | PC ← PC + 1 | B2→TMP | MR | (ACT)-(TMP); FLAGS | | | | | | | | |
| RLC | 0 0 0 0 | 0 1 1 1 | | | | (ALU)→ALU ROTATE | | ALU-A, CY | | MR | | | | | | | | | |
| RRC | 0 0 0 0 | 1 1 1 1 | | | | (ALU)→ALU ROTATE | | ALU-A, CY | | MR | | | | | | | | | |
| RAL | 0 0 0 1 | 0 1 1 1 | | | | (ALU, CY)→ALU ROTATE | | ALU-A, CY | | MR | | | | | | | | | |
| RAR | 0 0 0 1 | 1 1 1 1 | | | | (ALU, CY)→ALU ROTATE | | ALU-A, CY | | MR | | | | | | | | | |
| CMA | 0 0 1 0 | 1 1 1 1 | | | | (ALU)→A | | | | | | | | | | | | | |
| CMC | 0 0 1 1 | 1 1 1 1 | | | | CY→CY | | | | | | | | | | | | | |
| STC | 0 0 1 1 | 0 1 1 1 | | | | 1→CY | | | | | | | | | | | | | |
| JMP addr | 1 1 0 0 | 0 0 1 1 | | | | JUDGE CONDITION | | PC OUT STATUS[M] | PC ← PC + 1 | B2→Z | | | | | | | | | |
| J cond addr[1/3] | 1 1 C C | C 0 1 0 | | | | | | PC OUT STATUS[M] | PC ← PC + 1 | B2→Z | | | | | | | | | |
| CALL addr | 1 1 0 1 | 1 1 0 1 | | | | SP ← SP - 1 | | PC OUT STATUS[M] | PC ← PC + 1 | B2→Z | | | | | | | | | |
| C cond addr[1/3] | 1 1 C C | C 1 0 0 | | | | JUDGE CONDITION IF TRUE, SP ← SP - 1 | | PC OUT STATUS[M] | PC ← PC + 1 | B2→Z | | | | | | | | | |
| RST | 1 1 0 0 | 1 0 0 1 | | | | JUDGE CONDITION[1/3] | | SP OUT STATUS[M] | SP ← SP + 1 | DATA→Z | | | | | | | | | |
| R cond addr[1/3] | 1 1 C C | C 0 0 0 | | | | | | SP OUT STATUS[M] | SP ← SP + 1 | PCH→DATA BUS | | | | | | | | | |
| RST n | 1 1 N N | N 1 1 1 | | | INST←TMP/IR | 0→H0 INST←TMP/IR | | SP OUT STATUS[M] | SP ← SP - 1 | PCL→DATA BUS | | | | | | | | | |
| PCHL | 1 1 1 0 | 1 0 0 1 | | | | INH | | | | | | | | | | | | | |
| PUSH rp | 1 1 R P | 0 1 0 1 | | | | SP ← SP - 1 | PC | | | | | | | | | | | | |
| PUSH PSW | 1 1 1 1 | 0 1 0 1 | | | | SP ← SP - 1 | | SP OUT STATUS[M] | SP ← SP - 1 | INH→DATA BUS | | | | | | | | | |
| POP rp | 1 1 R P | 0 0 0 1 | | | | | | SP OUT STATUS[M] | SP ← SP - 1 | DAT→DATA BUS | | | | | | | | | |
| POP PSW | 1 1 1 1 | 0 0 0 1 | | | | | | SP OUT STATUS[M] | SP ← SP - 1 | DATA→FLAGS | | | | | | | | | |
| XTHL | 1 1 1 0 | 0 0 1 1 | | | | | | SP OUT STATUS[M] | SP ← SP + 1 | DATA→Z | | | | | | | | | |
| IN port | 1 1 0 1 | 1 0 1 1 | | | | | | PC OUT STATUS[M] | PC ← PC + 1 | B2→Z, W | | | | | | | | | |
| OUT port | 1 1 0 1 | 0 0 1 1 | | | | | | PC OUT STATUS[M] | PC ← PC + 1 | B2→Z, W | | | | | | | | | |
| EI | 1 1 1 1 | 1 0 1 1 | | | | SET INTE F/F | | | | | | | | | | | | | |
| DI | 1 1 1 1 | 0 0 1 1 | | | | RESET INTE F/F | | | | | | | | | | | | | |
| HLT | 0 1 1 1 | 0 1 1 0 | | | | | | HALT MODE[2/3] | | | | | | | | | | | |
| NOP | 0 0 0 0 | 0 0 0 0 | | | | | | | | | | | | | | | | | |

# Pengkodan SN74LS138

## Rajah pin keluaran dan simbolnya




## Jadual fungsi

| G1 | G2 | G3 | A | B | C | | Yo | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | X | 1 | X | X | X | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | 1 | X | X | X | X | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | X | X | X | X | X | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

intel

# 8085AH/8085AH-2/8085AH-1
# 8-BIT HMOS MICROPROCESSORS

- Single +5V Power Supply with 10% Voltage Margins
- 3 MHz, 5 MHz and 6 MHz Selections Available
- 20% Lower Power Consumption than 8085A for 3 MHz and 5 MHz
- 1.3 $\mu$s Instruction Cycle (8085AH); 0.8 $\mu$s (8085AH-2); 0.67 $\mu$s (8085AH-1)
- 100% Software Compatible with 8080A
- On-Chip Clock Generator (with External Crystal, LC or RC Network)
- On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control
- Four Vectored Interrupt Inputs (One is Non-Maskable) Plus an 8080A-Compatible Interrupt
- Serial In/Serial Out Port
- Decimal, Binary and Double Precision Arithmetic
- Direct Addressing Capability to 64K Bytes of Memory
- Available in 40-Lead Cerdip and Plastic Packages

(See Packaging Spec, Order #231369)

The Intel® 8085AH is a complete 8 bit parallel Central Processing Unit (CPU) implemented in N-channel, depletion load, silicon gate technology (HMOS). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's [8085AH (CPU), 8156H (RAM/IO) and 8755A (EPROM/IO)] while maintaining total system expandability. The 8085AH-2 and 8085AH-1 are faster versions of the 8085 AH.

The 8085AH incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080A, thereby offering a higher level of system integration.

The 8085AH uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of 8156H/8156H/8755A memory products allow a direct interface with the 8085AH.

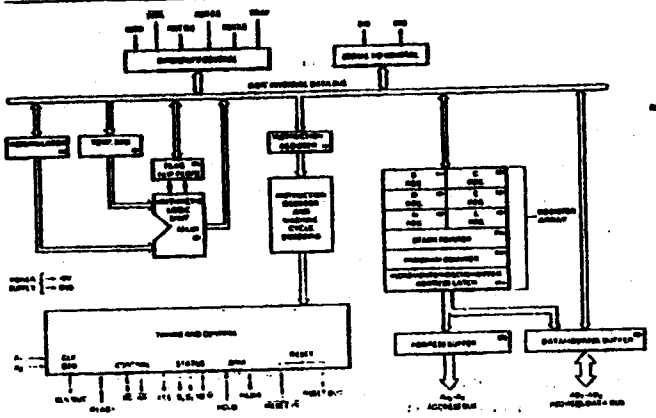


Figure 1. 8085AH CPU Functional Block Diagram

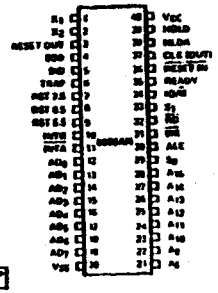


Figure 2. 8085AH Pin Configuration







Table 1. Pin Description

| Symbol | Type | Name and Function |
|---|---|---|
| $A_8$–$A_{15}$ | O | **Address Bus:** The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET. |
| $AD_0$–$_7$ | I/O | **Multiplexed Address/Data Bus:** Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles. |
| ALE | O | **Address Latch Enable:** It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated. |
| $S_0$, $S_1$, and IO/$\overline{M}$ | O | **Machine Cycle Status:**<br>IO/$\overline{M}$ $S_1$ $S_0$ Status<br>0 0 1 Memory write<br>0 1 0 Memory read<br>1 0 1 I/O write<br>1 1 0 I/O read<br>0 1 1 Opcode fetch<br>1 1 1 Interrupt Acknowledge<br>* 0 0 Halt<br>* X X Hold<br>* X X Reset<br>* = 3-state (high impedance)<br>X = unspecified<br>$S_1$ can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, $S_0$ and $S_1$ become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines. |
| $\overline{RD}$ | O | **Read Control:** A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer. 3-stated during Hold and Halt modes and during RESET. |
| $\overline{WR}$ | O | **Write Control:** A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET. |
| READY | I | **Ready:** If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times. |
| HOLD | I | **Hold:** Indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated. |
| HLDA | O | **Hold Acknowledge:** Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low. |
| INTR | I | **Interrupt Request:** Is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted. |
| $\overline{INTA}$ | O | **Interrupt Acknowledge:** Is used instead of (and has the same timing as) $\overline{RD}$ during the instruction cycle after an INTR is accepted. It can be used to activate an 8259A interrupt chip or some other interrupt port. |
| RST 5.5<br>RST 6.5<br>RST 7.5 | I | **Restart Interrupts:** These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted.<br>The priority of these interrupts is ordered as shown in Table 2. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction. |

Table 1. Pin Description (Continued)

| Symbol | Type | Name and Function |
|---|---|---|
| TRAP | I | **Trap:** Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 2.) |
| RESET IN | I | **Reset In:** Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. RESET IN is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay (see Figure 3). Upon power-up, RESET IN must remain low for at least 10 ms after minimum $V_{CC}$ has been reached. For proper reset operation after the power-up duration, RESET IN should be kept low a minimum of three clock periods. The CPU is held in the reset condition as long as RESET IN is applied. |
| RESET OUT | O | **Reset Out:** Reset Out indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods. |
| $X_1$, $X_2$ | I | $X_1$ and $X_2$: Are connected to a crystal, LC, or RC network to drive the internal clock generator. $X_1$ can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency. |
| CLK | O | **Clock:** Clock output for use as a system clock. The period of CLK is twice the $X_1$, $X_2$ input period. |
| SID | I | **Serial Input Data Line:** The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed. |
| SOD | O | **Serial Output Data Line:** The output SOD is set or reset as specified by the SIM instruction. |
| $V_{CC}$ | | **Power:** +5 volt supply. |
| $V_{SS}$ | | **Ground:** Reference. |

Table 2. Interrupt Priority, Restart Address, and Sensitivity

| Name | Priority | Address Branched To (1) When Interrupt Occurs | Type Trigger |
|---|---|---|---|
| TRAP | 1 | 24H | Rising edge AND high level until sampled. |
| RST 7.5 | 2 | 3CH | Rising edge (latched). |
| RST 6.5 | 3 | 34H | High level until sampled. |
| RST 5.5 | 4 | 2CH | High level until sampled. |
| INTR | 5 | See Note (2). | High level until sampled. |

**NOTES:**
1. The processor pushes the PC on the stack before branching to the indicated address.
2. The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.

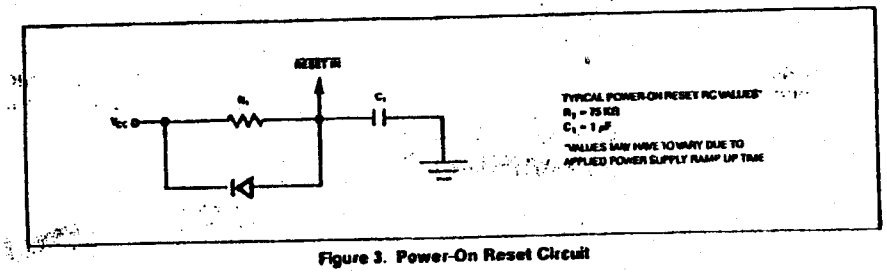



Figure 3. Power-On Reset Circuit





## FUNCTIONAL DESCRIPTION

The 8085AH is a complete 8-bit parallel central processor. It is designed with N-channel, depletion load, silicon gate technology (HMOS), and requires a single +5 volt supply. Its basic clock speed is 3 MHz (8085AH), 5 MHz (8085AH-2), or 6 MHz (8085AH-1), thus improving on the present 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU (8085AH), a RAM/IO (8156H), and an EPROM/IO chip (8755A).

The 8085AH has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085AH register set is as follows:

| Mnemonic | Register | Contents |
|---|---|---|
| ACC or A | Accumulator | 8 bits |
| PC | Program Counter | 16-bit address |
| BC,DE,HL | General-Purpose Registers; data pointer (HL) | 8 bits x 6 or 16 bits x 3 |
| SP | Stack Pointer | 16-bit address |
| Flags or F | Flag Register | 5 flags (8-bit space) |

The 8085AH uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085AH provides RD, WR, $S_0$, $S_1$, and IO/M signals for bus control. An Interrupt Acknowledge signal (INTA) is also provided. HOLD and all interrupts are synchronized with the processor's internal clock. The 8085AH also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085AH has three maskable, vector interrupt pins, one nonmaskable TRAP interrupt, and a bus vectored interrupt, INTR.

## INTERRUPT AND SERIAL I/O

The 8085AH has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The nonmaskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 2.)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* like INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive*.

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request (a normally high level signal with a low going pulse is recommended for highest system noise immunity). The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a RESET IN to the 8085AH. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and RESET IN. (See SIM, Chapter 5 of the 8080/8085 User's Manual.)

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP—highest priority, RST 7.5, RST 6.5, RST 5.5, INTR—lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive*. The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. Figure 4 illustrates the TRAP interrupt request circuitry within the 8085AH. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

intel

# 8155H/8156H/8155H-2/8156H-2
# 2048-BIT STATIC HMOS RAM WITH I/O PORTS AND TIMER

- Single +5V Power Supply with 10% Voltage Margins
- 30% Lower Power Consumption than the 8155 and 8156
- 256 Word x 8 Bits
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8-Bit I/O Ports
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/Timer
- Compatible with 8085AH and 8088 CPU
- Multiplexed Address and Data Bus
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8155H and 8156H are RAM and I/O chips implemented in N-Channel, depletion load, silicon gate technology (HMOS), to be used in the 8085AH and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085AH CPU. The 8155H-2 and 8156H-2 have maximum access times of 330 ns for use with the 8085AH-2 and the 5 MHz 8088 CPU.

The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

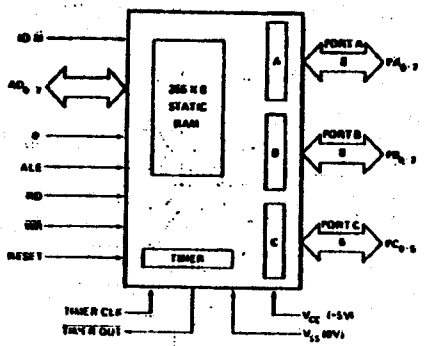



Figure 1. Block Diagram

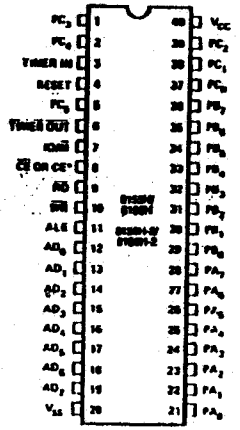



Figure 2. Pin Configuration







Table 1. Pin Description

| Symbol | Type | Name and Function |
|---|---|---|
| RESET | I | **Reset:** Pulse provided by the 8085AH to initialize the system (connect to 8085AH RESET OUT). Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulse should typically be two 8085AH clock cycle times. |
| $AD_{0-7}$ | I/O | **Address/Data:** 3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus. The 8-bit address is latched into the address latch inside the 8156H/155H on the falling edge of ALE. The address can be either for the memory section or the I/O section depending on the IO/$\overline{M}$ input. The 8-bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal. |
| CE or $\overline{CE}$ | I | **Chip Enable:** On the 8155H, this pin is $\overline{CE}$ and is ACTIVE LOW. On the 8156H, this pin is CE and is ACTIVE HIGH. |
| $\overline{RD}$ | I | **Read Control:** Input low on this line with the Chip Enable active enables one and $AD_{0-7}$ buffers. If IO/$\overline{M}$ pin is low, the RAM content will be read out to the AD bus. Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus. |
| $\overline{WR}$ | I | **Write Control:** Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register, depending on IO/$\overline{M}$. |
| ALE | I | **Address Latch Enable:** This control signal latches both the address on the $AD_{0-7}$ lines and the state of the Chip Enable and IO/$\overline{M}$ into the chip at the falling edge of ALE. |
| IO/$\overline{M}$ | I | **I/O Memory:** Selects memory if low and I/O and command/status registers if high. |
| $PA_{0-7}$(8) | I/O | **Port A:** These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register. |
| $PB_{0-7}$(8) | I/O | **Port B:** These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register. |
| $PC_{0-5}$(6) | I/O | **Port C:** These 6 pins can function as either input port, output port, or as control signals for PA and PB. Programming is done through the command register. When $PC_{0-5}$ are used as control signals, they will provide the following:<br>$PC_0$ — A INTR (Port A Interrupt)<br>$PC_1$ — ABF (Port A Buffer Full)<br>$PC_2$ — A $\overline{STB}$ (Port A Strobe)<br>$PC_3$ — B INTR (Port B Interrupt)<br>$PC_4$ — B $\overline{BF}$ (Port B Buffer Full)<br>$PC_5$ — B $\overline{STB}$ (Port B Strobe) |
| TIMER IN | I | **Timer Input:** Input to the counter-timer. |
| TIMER OUT | O | **Timer Output:** This output can be either a square wave or a pulse, depending on the timer mode. |
| $V_{CC}$ | | **Voltage:** +5 volt supply. |
| $V_{SS}$ | | **Ground:** Ground reference. |

## FUNCTIONAL DESCRIPTION

The 8155H/8156H contains the following:

- 2k Bit Static RAM organized as 256 x 8
- Two 8-bit I/O ports (PA & PB) and one 6-bit I/O port (PC)
- 14-bit timer-counter

The IO/$\overline{M}$ (IO/Memory Select) pin selects either the five registers (Command, Status, $PA_{0-7}$, $PB_{0-7}$, $PC_{0-5}$) or the memory (RAM) portion.

The 8-bit address on the Address/Data lines. Chip Enable input CE or $\overline{CE}$, and IO/$\overline{M}$ are all latched on-chip at the falling edge of ALE.

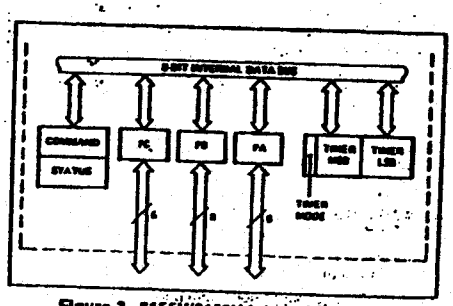



Figure 3. 8155H/8156H Internal Registers

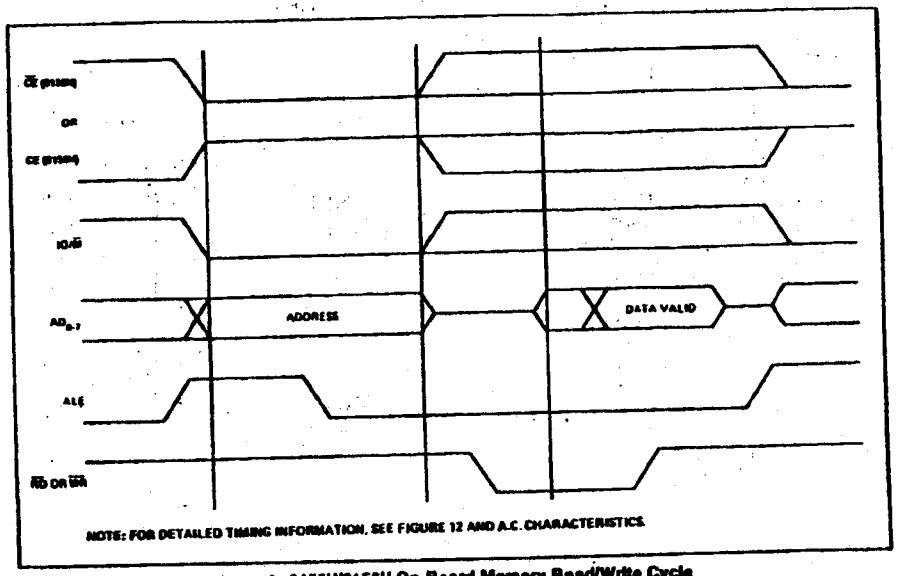



Figure 4. 8155H/8156H On-Board Memory Read/Write Cycle

## PROGRAMMING OF THE COMMAND REGISTER

The command register consists of eight latches. Four bits (0-3) define the mode of the ports, two bits (4-5) enable or disable the interrupt from port C when it acts as control port, and the last two bits (6-7) are for the timer.

The command register contents can be altered at any time by using the I/O address XXXXX000 during a WRITE operation with the Chip Enable active and IO/M = 1. The meaning of each bit of the command byte is defined in Figure 5. The contents of the command register may never be read.

## READING THE STATUS REGISTER

The status register consists of seven latches, one for each bit: six (0-5) for the status of the ports and one (6) for the status of the timer.

The status of the timer and the I/O section can be polled by reading the Status Register (Address XXXXX000). Status word format is shown in Figure 6. Note that you may never write to the status register since the command register shares the same I/O address and the command register is selected when a write to that address is issued.

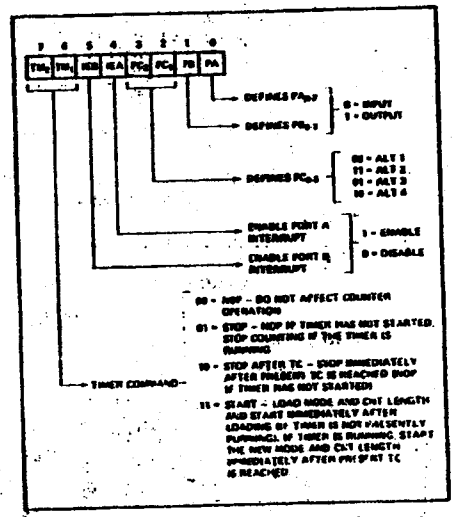



Figure 5. Command Register Bit Assignment





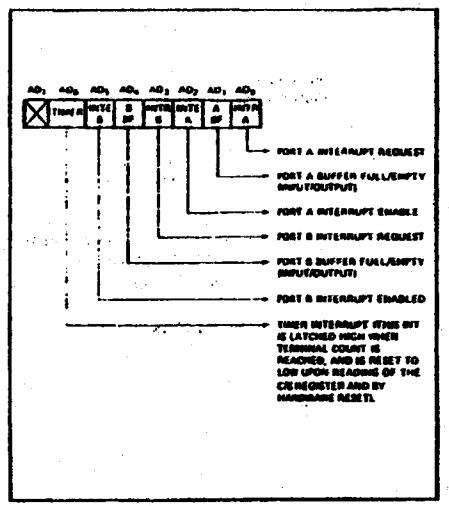



Figure 6. Status Register Bit Assignment

## INPUT/OUTPUT SECTION

The I/O section of the 8155H/8156H consists of five registers: (See Figure 7.)

- **Command/Status Register (C/S)** — Both registers are assigned the address XXXXX000. The C/S address serves the dual purpose.

  When the C/S registers are selected during WRITE operation, a command is written into the command register. The contents of this register are *not* accessible through the pins.

  When the C/S (XXXXX000) is selected during a READ operation, the status information of the I/O ports and the timer becomes available on the $AD_{0-7}$ lines.

- **PA Register** — This register can be programmed to be either input or output ports depending on the status of the contents of the C/S Register. Also depending on the command, this port can operate in either the basic mode or the strobed mode (See timing diagram). The I/O pins assigned in relation to this register are $PA_{0-7}$. The address of this register is XXXXX001.

- **PB Register** — This register functions the same as PA Register. The I/O pins assigned are $PB_{0-7}$. The address of this register is XXXXX010.

- **PC Register** — This register has the address XXXXX011 and contains only 6 bits. The 6 bits can be programmed to be either input ports, output ports or as control signals for PA and PB by properly programming the $AD_2$ and $AD_3$ bits of the C/S register.

  When $PC_{0-5}$ is used as a control port, 3 bits are assigned for Port A and 3 for Port B. The first bit is an interrupt that the 8155H sends out. The second is an output signal indicating whether the buffer is full or empty, and the third is an input pin to accept a strobe for the strobed input mode. (See Table 2.)

When the 'C' port is programmed to either ALT3 or ALT4, the control signals for PA and PB are initialized as follows:

| CONTROL | INPUT MODE | OUTPUT MODE |
|---|---|---|
| BF | Low | Low |
| INTR | Low | High |
| STB | Input Control | Input Control |

| I/O ADDRESS† | | | | | | | | SELECTION |
|---|---|---|---|---|---|---|---|---|
| A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | |
| X | X | X | X | X | 0 | 0 | 0 | Internal Command/Status Register |
| X | X | X | X | X | 0 | 0 | 1 | General Purpose I/O Port A |
| X | X | X | X | X | 0 | 1 | 0 | General Purpose I/O Port B |
| X | X | X | X | X | 0 | 1 | 1 | Port C — General Purpose I/O or Control |
| X | X | X | X | X | 1 | 0 | 0 | Low Order 8 bits of Timer Count |
| X | X | X | X | X | 1 | 0 | 1 | High 6 bits of Timer Count and 2 bits of Timer Mode |

X: Don't Care.

†: I/O Address must be qualified by CE = 1 (8155H) or CS = 0 (8156H) and IO/M = 1 in order to select the appropriate register.

Figure 7. I/O Port and Timer Addressing Scheme

Figure 8 shows how I/O PORTS A and B are structured within the 8155H and 8156H:

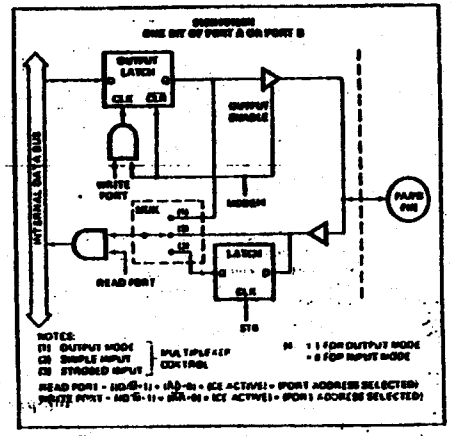



Figure 8. 8155H/8156H Port Functions

**Table 2. Port Control Assignment**

| Pin | ALT 1 | ALT 2 | ALT 3 | ALT 4 |
|---|---|---|---|---|
| PC0 | Input Port | Output Port | A INTR (Port A Interrupt) | A INTR (Port A Interrupt) |
| PC1 | Input Port | Output Port | A BF (Port A Buffer Full) | A BF (Port A Buffer Full |
| PC2 | Input Port | Output Port | A $\overline{STB}$ (Port A Strobe) | A $\overline{STB}$ (Port A Strobe) |
| PC3 | Input Port | Output Port | Output Port | B INTR (Port B Interrupt) |
| PC4 | Input Port | Output Port | Output Port | B BF (Port B Buffer Full) |
| PC5 | Input Port | Output Port | Output Port | B $\overline{STB}$ (Port B Strobe) |

Note in the diagram that when the I/O ports are programmed to be output ports, the contents of the output ports can still be read by a READ operation when appropriately addressed.

The outputs of the 8155H/8156H are "glitch-free" meaning that you can write a "1" to a bit position that was previously "1" and the level at the output pin will not change.

Note also that the output latch is cleared when the port enters the input mode. The output latch cannot be loaded by writing to the port if the port is in the input mode. The result is that each time a port mode is changed from input to output, the output pins will go low. When the 8155H/56H is RESET, the output latches are all cleared and all 3 ports enter the input mode.

When in the ALT 1 or ALT 2 modes, the bits of PORT C are structured like the diagram above in the simple input or output mode, respectively.

Reading from an input port with nothing connected to the pins will provide unpredictable results.

Figure 9 shows how the 8155H/8156H I/O ports might be configured in a typical MCS-85 system.

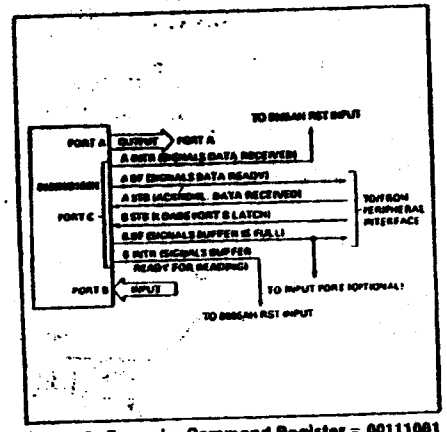



**Figure 9. Example: Command Register = 00111001**

## TIMER SECTION

The time is a 14-bit down-counter that counts the TIMER IN pulses and provides either a square wave or pulse when terminal count (TC) is reached.

The timer has the I/O address XXXXX100 for the low order byte of the register and the I/O address XXXXX101 for the high order byte of the register. (See Figure 7).

To program the timer, the COUNT LENGTH REG is loaded first, one byte at a time, by selecting the timer addresses. Bits 0-13 of the high order count register will specify the length of the next count and bits 14-15 of the high order register will specify the timer output mode (see Figure 10). The value loaded into the count length register can have any value from 2H through 3FFFH in Bits 0-13.

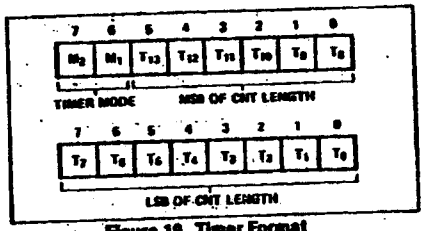



**Figure 10. Timer Format**

There are four modes to choose from: M2 and M1 define the timer mode, as shown in Figure 11.

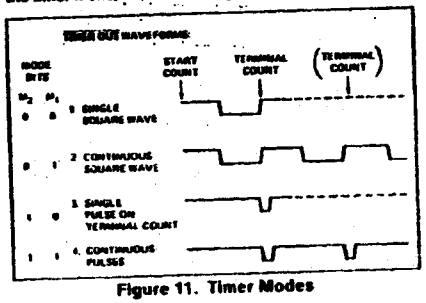



**Figure 11. Timer Modes**





Bits 6-7 ($TM_2$ and $TM_1$) of command register contents are used to start and stop the counter. There are four commands to choose from.

| $TM_2$ | $TM_1$ | |
|---|---|---|
| 0 | 0 | NOP — Do not affect counter operation |
| 0 | 1 | STOP — NOP if timer has not started, stop counting if the timer is running. |
| 1 | 0 | STOP AFTER TC — Stop immediately after present TC is reached (NOP if timer has not started) |
| 1 | 1 | START — Load mode and CNT length and start immediately after loading (if timer is not presently running). If timer is running, start the new mode and CNT length immediately after present TC is reached. |

Note that while the counter is counting, you may load a new count and mode into the count length registers. Before the new count and mode will be used by the counter, you must issue a START command to the counter. This applies even though you may only want to change the count and use the previous mode.

In case of an odd-numbered count, the first half-cycle of the squarewave output, which is high, is one count longer than the second (low) half-cycle, as shown in Figure 12.

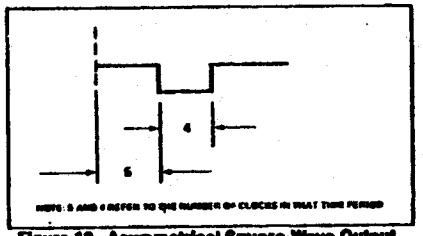



**Figure 12. Asymmetrical Square-Wave Output Resulting from Count of 9**

The counter in the 8155H is not initialized to any particular mode or count when hardware RESET occurs, but RESET does stop the counting. Therefore, counting cannot begin following RESET until a START command is issued via the C/S register.

Please note that the timer circuit on the 8155H/8156H chip is designed to be a square-wave timer, not an event counter. To achieve this, it counts down by twos twice in completing one cycle. Thus, its registers do not contain values directly representing the number of TIMER IN pulses received. You cannot load an initial value of 1 into the count register and cause the timer to operate, as its terminal count value is 10 (binary) or 2 (decimal). (For the detection of single pulses, it is suggested that one of the hardware interrupt pins on the 8085AH be used.) After the timer has started counting down, the values residing in the count registers can be used to calculate the actual number of TIMER IN pulses required to complete the timer cycle if desired. To obtain the remaining count, perform the following operations in order:

1. Stop the count
2. Read in the 16-bit value from the count length registers
3. Reset the upper two mode bits
4. Reset the carry and rotate right one position all 16 bits through carry
5. If carry is set, add 1/2 of the full original count (1/2 full count — 1 if full count is odd).

Note: If you started with an odd count and you read the count length register before the third count pulse occurs, you will not be able to discern whether one or two counts has occurred. Regardless of this, the 8155H/56H always counts out the right number of pulses in generating the TIMER OUT waveforms.

intel

# 8755A/8755A-2
# 16,384-BIT EPROM WITH I/O

- 2048 Words x 8 Bits
- Single +5V Power Supply ($V_{CC}$)
- Directly Compatible with 8085A and 8088 Microprocessors
- U.V. Erasable and Electrically Reprogrammable
- Internal Address Latch
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- 40-Pin DIP
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8755A is an erasable and electrically reprogrammable ROM (EPROM) and I/O chip to be used in the 8085AH and iAPX 88 microprocessor systems. The EPROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 450 ns to permit use with no wait states in an 8085AH CPU.

The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is individually programmable as input or output.

The 8755A-2 is a high speed selected version of the 8755A compatible with the 5 MHz 8085AH-2 and the 5 MHz iAPX 88 microprocessor.

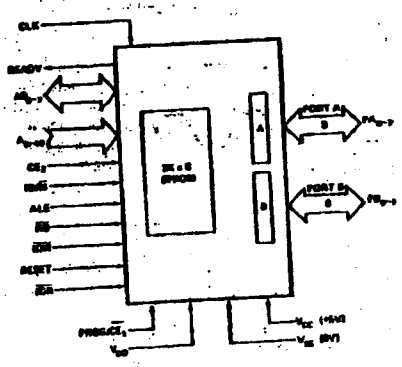



Figure 1. Block Diagram

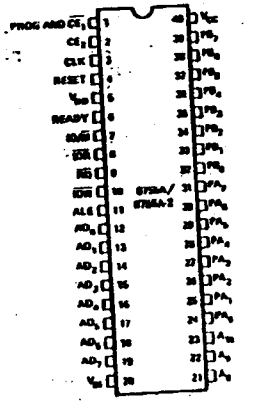



Figure 2. Pin Configuration







Table 1. Pin Description

| Symbol | Type | Name and Function |
|---|---|---|
| ALE | I | **Address Latch Enable:** When Address Latch Enable goes high, $AD_{0-7}$, $\overline{IO/M}$, $A_{8-10}$, $CE_2$, and $\overline{CE_1}$ enter the address latches. The signals (AD, $\overline{IO/M}$ $AD_{8-10}$, $CE_2$, $\overline{CE_1}$) are latched in at the trailing edge of ALE. |
| $AD_{0-7}$ | I/O | **Bidirectional Address/Data Bus:** The lower 8-bits of the PROM or I/O address are applied to the bus lines when ALE is high.<br>During an I/O cycle, Port A or B is selected based on the latched value of $AD_0$. If $\overline{RD}$ or $\overline{IOR}$ is low when the latched Chip Enables are active, the output buffers present data on the bus. |
| $A_{8-10}$ | I | **Address Bus:** These are the high order bits of the PROM address. They do not affect I/O operations. |
| PROG/$\overline{CE_1}$<br>$CE_2$ | I | **Chip Enable Inputs:** $\overline{CE_1}$ is active low and $CE_2$ is active high. The 8755A can be accessed only when both Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the $AD_{0-7}$ and READY outputs will be in a high impedance state. $\overline{CE_1}$ is also used as a programming pin. (See section on programming.) |
| $\overline{IO/M}$ | I | **I/O Memory:** If the latched $\overline{IO/M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the PROM. |
| $\overline{RD}$ | I | **Read:** If the latched Chip Enables are active when $\overline{RD}$ goes low, the $AD_{0-7}$ output buffers are enabled and output either the selected PROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the $AD_{0-7}$ output buffers are 3-stated. |
| $\overline{IOW}$ | I | **I/O Write:** If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of $AD_0$ to be written with the data on $AD_{0-7}$. The state of $IO/\overline{M}$ is ignored. |
| CLK | I | **Clock:** The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE_1}$ low, $CE_2$ high, and ALE high. |
| READY | O | **Ready** is a 3-state output controlled by $\overline{CE_1}$, $CE_2$, ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK. (See Figure 8c.) |
| $PA_{0-7}$ | I/O | **Port A:** These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from $AD_0$. $AD_1$.<br>Read Operation is selected by either $\overline{IOR}$ low and active Chip Enables and $AD_0$ and $AD_1$ low, or $IO/\overline{M}$ high, $\overline{RD}$ low, active Chip Enables, and $AD_0$ and $AD_1$ low. |
| $PB_{0-7}$ | I/O | **Port B:** This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from $AD_0$ and a 0 from $AD_1$. |
| RESET | I | **Reset:** In normal operation, an input high on RESET causes all pins in Ports A and B to assume input mode (clear DDR register). |
| $\overline{IOR}$ | I | **I/O Read:** When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination of $IO/\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to $V_{CC}$ ("1"). |
| $V_{CC}$ | | **Power:** +5 volt supply. |
| $V_{SS}$ | | **Ground:** Reference. |
| $V_{DD}$ | | **Power Supply:** $V_{DD}$ is a programming voltage, and must be tied to $V_{CC}$ when the 8755A is being read.<br>For programming, a high voltage is supplied with $V_{DD}$ = 25V, typical. (See section on programming.) |

## FUNCTIONAL DESCRIPTION

### PROM Section

The 8755A contains an 8-bit address latch which allows it to interface directly to MCS-48, MCS-85 and iAPX 88/10 Microcomputers without additional hardware.

The PROM section of the chip is addressed by the 11-bit address and the Chip Enables. The address, $\overline{CE}_1$ and $CE_2$ are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/$\overline{M}$ is low when $\overline{RD}$ goes low, the contents of the PROM location addressed by the latched address are put out on the $AD_{0-7}$ lines (provided that $V_{PD}$ is tied to $V_{CC}$).

### I/O Section

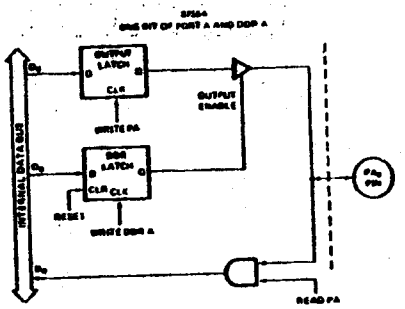
The I/O section of the chip is addressed by the latched value of $AD_{0-1}$. Two 8-bit Data Direction Registers (DDR) in 8755A determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8755A are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

| $AD_1$ | $AD_0$ | Selection |
|---|---|---|
| 0 | 0 | Port A |
| 0 | 1 | Port B |
| 1 | 0 | Port A Data Direction Register (DDR A) |
| 1 | 1 | Port B Data Direction Register (DDR B) |

When $\overline{IOW}$ goes low and the Chip Enables are active, the data on the $AD_{0-7}$ is written into I/O port selected by the latched value of $AD_{0-1}$. During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of IO/$\overline{M}$. The actual output level does not change until $\overline{IOW}$ returns high. (glitch free output)

A port can be read out when the latched Chip Enables are active and either $\overline{RD}$ goes low with IO/$\overline{M}$ high, or $\overline{IOR}$ goes low. Both input and output mode bits of a selected port will appear on lines $AD_{0-7}$.

To clarify the function of the I/O Ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.




Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the Output Latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.

**TABLE 1. 8755A PROGRAMMING MODULE CROSS REFERENCE**

| MODULE NAME | USE WITH |
|---|---|
| UPP 955 | UPP(4) |
| UPP UP2(2) | UPP 855 |
| PROMPT 975 | PROMPT 80/85(3) |
| PROMPT 475 | PROMPT 48(1) |

NOTES:
1. Described on p. 13-34 of 1978 Data Catalog.
2. Special adaptor socket.
3. Described on p. 13-39 of 1978 Data Catalog.
4. Described on p. 13-71 of 1978 Data Catalog.





## ERASURE CHARACTERISTICS

The erasure characteristics of the 8755A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8755A in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8755A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8755 window to prevent unintentional erasure.

The recommended erasure procedure for the 8755A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15W-sec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000$\mu$W/cm² power rating. The 8755A should be placed within one inch from the lamp tubes during erasure. Some lamps have a filter on their tubes and this filter should be removed before erasure.

## PROGRAMMING

Initially, and after each erasure, all bits of the EPROM portions of the 8755A are in the "1" state. Information is introduced by selectively programming "0" into the desired bit locations. A programmed "0" can only be changed to a "1" by UV erasure.

The 8755A can be programmed on the Intel® Universal PROM Programmer (UPP), and the PROMPT™ 80/85 and PROMPT-48™ design aids. The appropriate programming modules and adapters for use in programming both 8755A's and 8755's are shown in Table 1.

The program mode itself consists of programming a single address at a time, giving a single 50 msec pulse for every address. Generally, it is desirable to have a verify cycle after a program cycle for the same address as shown in the attached timing diagram. In the verify cycle (i.e., normal memory read cycle) $V_{PD}$ should be at +5V.

Preliminary timing diagrams and parameter values pertaining to the 8755A programming operation are contained in Figure 7.

## SYSTEM APPLICATIONS

### System Interface with 8085AH and iAPX 88

A system using the 8755A can use either one of the two I/O interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both $CE_2$ and $\overline{CE}_1$. By using a combination of unused address lines $A_{11-15}$ and the Chip Enable inputs, the 8085AH system can use up to 5 each 8755A's without requiring a CE decoder. See Figure 4a and 4b.

If a memory mapped I/O approach is used the 8755A will be selected by the combination of both the Chip Enables and IO/$\overline{M}$ using $AD_{8-15}$ address lines. See Figure 3.

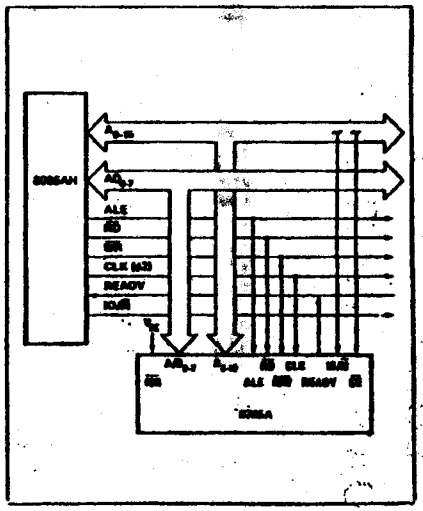



Figure 3. 8755A in 8085AH System (Memory Mapped I/O)

# RAJAH PIN LUAR CIP DAC0808, ADC 0801, DAN RAM STATIK 2114