Instruction cycle

The nessary steps, that a cpu carries out to fetch an instruction and nessary data from memory and to exen it, constitute and Instruction cycle.

An instruction cycle consists of Fetch cycle and exente cy

Instruction cycle = Fetch cycle + Exemte cycle.

Fetch cycle.

In fetch cycle, a cpu fetches opcode from memory. The nessare steps which are carried out to fetch an 'opcode' from memory constitute a fetch cycle

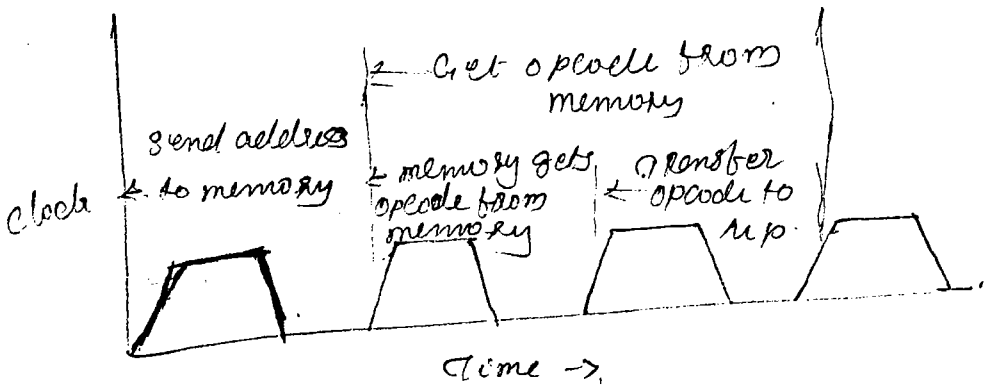
The first byte of an instruction is opcode. An instruction may be more than 1 byte long. The other bytes are data o operand address. The program counter keeps track of instru execution. In the beginning of a fetch cycle. content of program counter, which is the address of memory location where opcode is available, is sent to the memory. The memory places the opcode on the databus so as to transfer it to the cpu. The entire operation of fetching an opcod takes 3 clockcycles (1 for execution

Execute cycle.

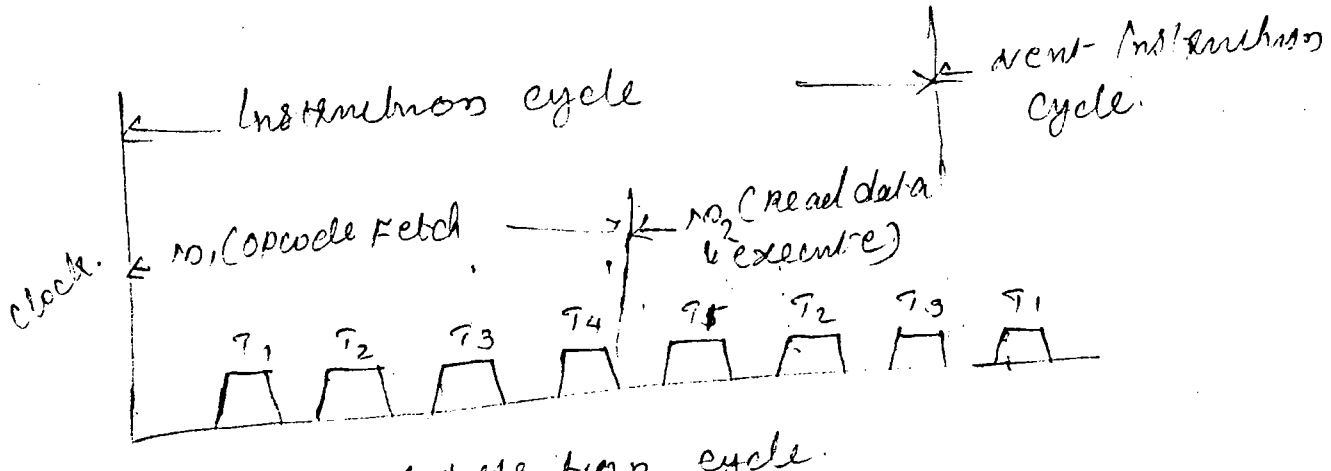
The nessary steps which are carried out to get data, if any from the memory and to perform the specific oper specified in an instruction, constitute an execution cycle.

The opcode fetched from memory goes to the data register (data/address buffer) then to instruction register IR. From IR it goes to the decoder circuitry within the MP which decodes the instruction. After that execution begins if the operand is in the general purpose registers, execution is comparatively performed. It takes 1 clock cycle (decode+execution). If the instruction contains data or operand address which are st in the memory, the cpu has to perform some read operation to get the desired data. some write cycles are also be perform to send data from cpu to memory or I/O device.

So execution cycle = Read+write or Read or write.



fetch cycle



Typical instruction cycle.

machine cycle

The necessary steps carried out to perform the operation of accessing either memory or I/O device, constitute a machine cycle. In other words, necessary steps carried out to perform a fetch, a read or write operation constitute a machine cycle.

machine cycle → one basic operation such as opcode fetch, memory read, memory write, I/O read or I/O write is performed.

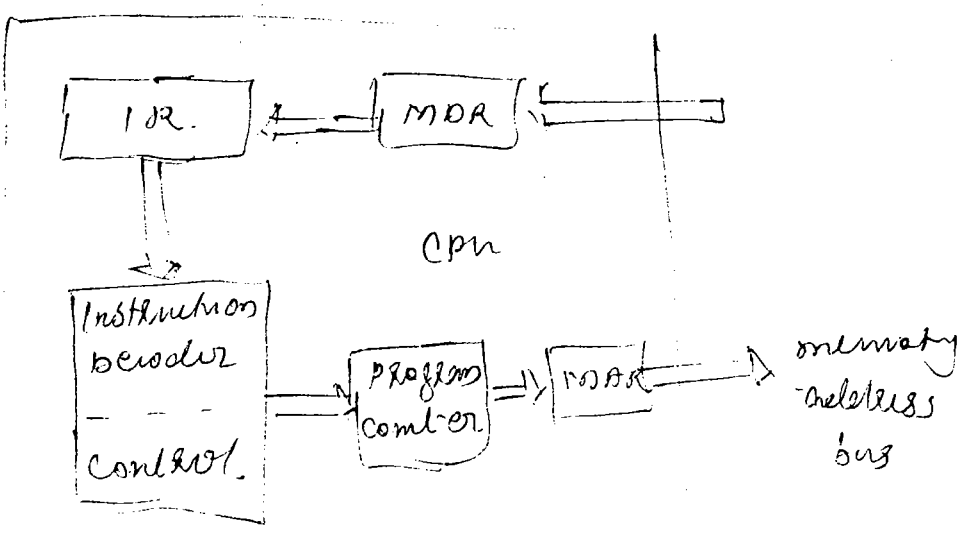
An instruction cycle consist of several machine cycles

T-state / state

One subdivision of an operation, performed in one clock cycle is called a state or T state. Que subdivisions are synchronized with system clock.

Instruction & data flow

a. flow of instruction word (opcode)



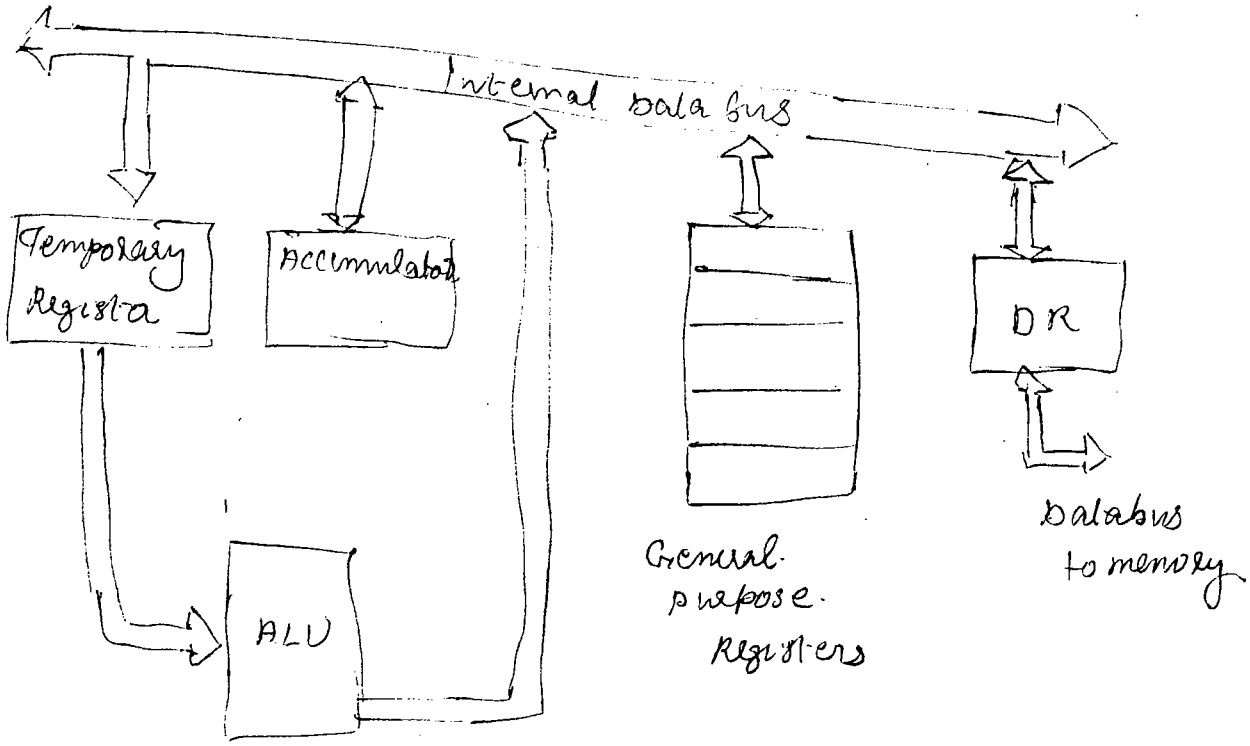
A computer receives instructions and data both in binary form. A group of 8 bits is called a byte. Data and instructions for computers are specified in byte forms.

The 'word length' of a computer is the number of bits it handles at a time. Two kinds of words, namely instruction word (opcode) and data word are processed during an instruction cycle.

In the beginning of a fetch cycle, the content of PC is transferred to MAR. The content of MAR is transferred to memory through address bus. By sending certain control signals to the memory the MP also indicates that it wants to read the content of memory. The decoder circuitry in the memory is activated and the memory understands what is to be done.

Then the memory sends opcode to the microprocessor through the data bus. The opcode first comes to MDR. The opcode is then place in the instruction register (IR). The instruction is decoded by the instruction decoder and it is executed. Finally PC is incremented.

b. Flow of data word.



The execution of an instruction requires the flow of data word in the most of the instructions. The flow of a data word is shown in figure. A data word is required either from memory or input device. The data word flows to the processor through databus and is placed in the accumulator or any other general purpose register depending upon the instruction. After the execution of a program result is placed in memory or sent to an output device. When data word is written into the memory, it is also held in DR until the write operation is complete.

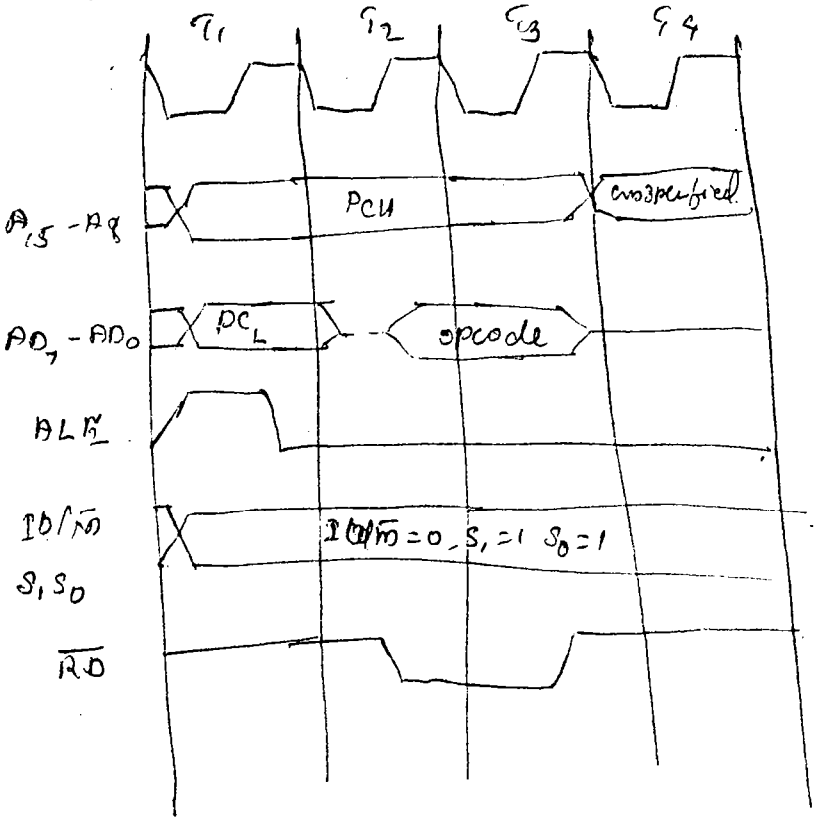
Timing Diagram

The necessary steps which are carried out in a machine cycle can be represented graphically such a graphical representation is called timing diagram.

Timing diagram for opcode fetch cycle

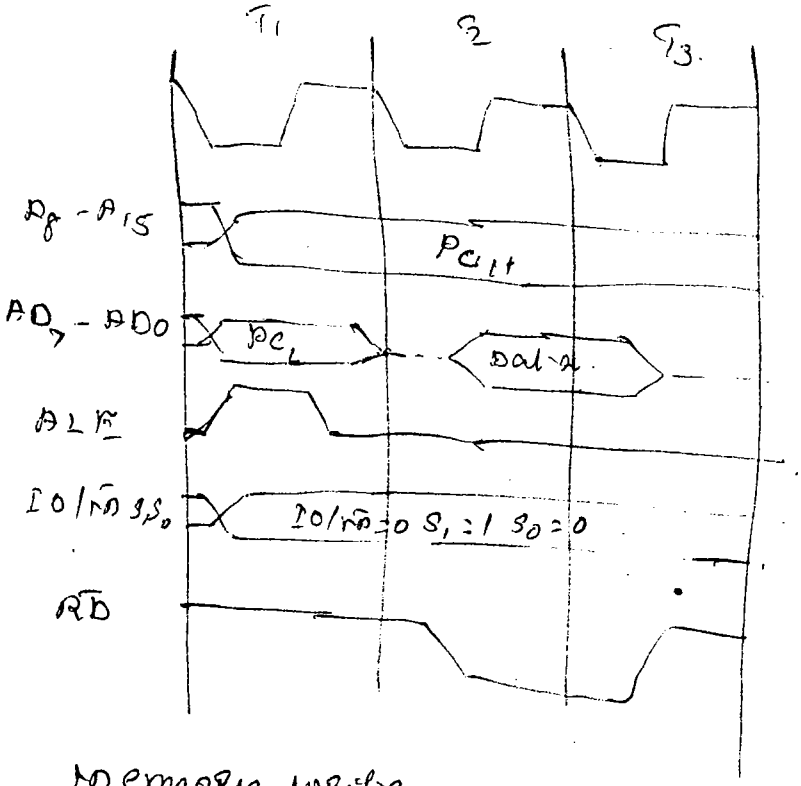
T_1 T_2 T_3 and T_4 are consecutive 4 clock cycles. The microprocessor issues a low $\overline{IO/M}$ to indicate that it wants to make communication with the memory. Again the microprocessor sends out high S_0 and S_1 signals to indicate that it is going to perform fetch operation. During the first clock cycle, T, the MP sends out address of the memory location where the opcode is available. The 16 bit memory address is sent through the address bus A and address/data bus AD. The 8 MSBs of the memory address are sent over A bus and 8 LSBs of the memory address over AD bus. Since the AD bus is needed to transfer data during subsequent clock cycles, it is used in time-multiplexed mode. Therefore it has to be made available to carry data during T_2 and T_3. To accomplish this the MP sends out ALE_2 (address latch enable) to latch the 8 LSBs of the memory address either in the memory or an external latch so that the complete 16 bit address may be available in the subsequent clock cycles. During T_2, the AD bus becomes ready to carry data. In T_2, the MP makes $\overline{RD}$ low. Now processor gets the opcode from the specified memory location and places it on the data bus. During T_3 the opcode is placed in instruction register (IR). The memory is disabled

When $\overline{RD}$ goes high during T_3. The fetch cycle is completed by T_3. The opcode is decoded in T_4.



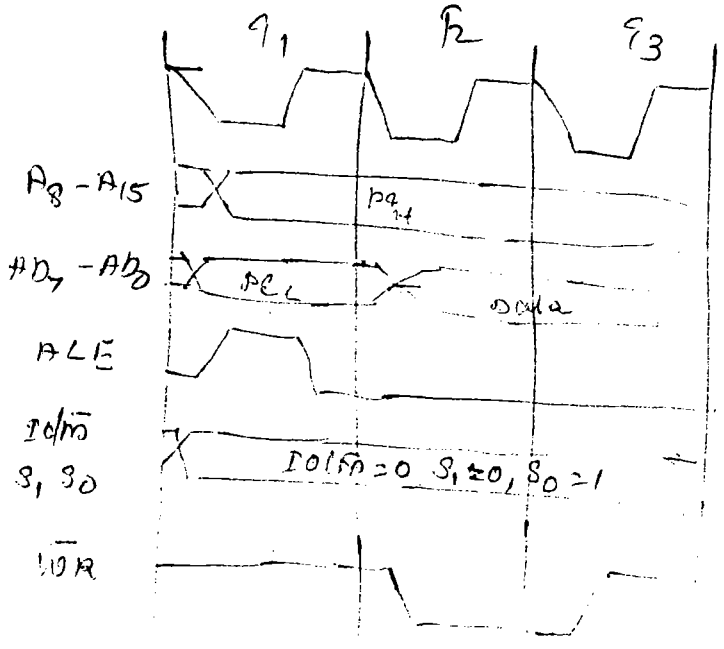
memory Read

In memory read cycle, the mp reads the content of the memory location. In this cycle IO/$\overline{M}$ goes low indicating that the address is for memory. S_1 and S_0 are set to 1 and 0 respectively for read operation. On the address lines $A_8 - A_{15}$ the 8 MSBs of the memory address of the data are sent. During T_1, 8 LSBs of the memory address of the data are sent on $AD_0 - AD_7$. During T_2, 8 LSBs of the address is latched and $AD_0 - AD_7$ are made free for data transmission. $\overline{RD}$ goes low in T_2 to enable the memory for read operation. Now data is placed on data bus. During T_3 the data enters into the CPU. In T_3 $\overline{RD}$ goes high and disables the memory. memory read contains 3 clock cycles.



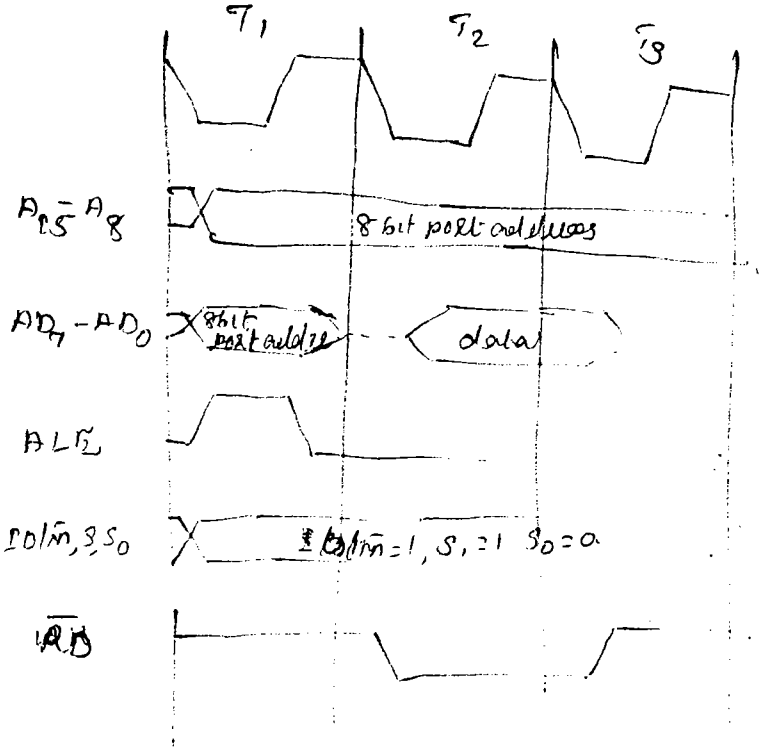
Memory write

In a memory write cycle the cpu sends data from accumulator or any other register to the memory. The status signals S_0 S_1 are 1 and 0 respectively for write operation. WR goes low in T_2 indicating that write operation to be performed. During T_2 the AD bus is not disable. But the data to be sent out to the memory is placed on the AD bus. As soon as WR goes high in T_3 the write operation is terminated



I/O read

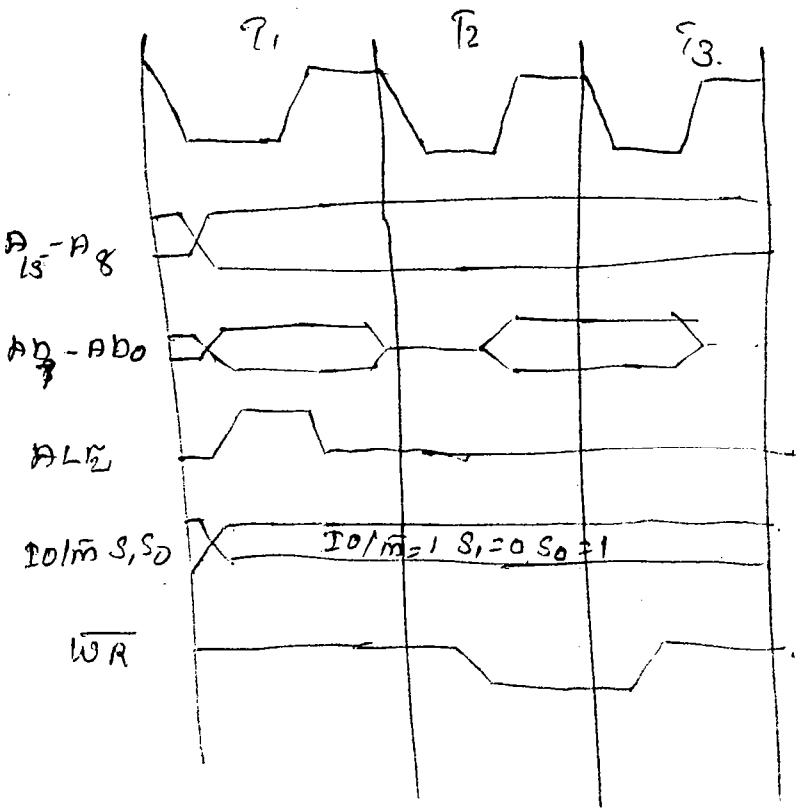
In an I/O read cycle the microprocessor reads the data available at an input port or input device. The data is placed in the accumulator. An I/O read cycle is similar to memory read cycle. The only difference between a memory read cycle and an I/O read cycle is that the signal IO/$\overline{M}$ goes high in case of I/O read. It indicates that the address on the A bus is for an input device. In case of I/O device or I/O port address is only 8 bits long and. therefore, the address of I/O device is duplicated on both A and D buses. It requires 3 machine cycles for execution.



I/O write

In an I/O write cycle, the CPU sends data to an I/O port or I/O device from the accumulator. In case of I/O write cycle IO/$\overline{M}$ goes high indicating that the address sent out by the CPU is for I/O device or I/O port. The address of I/O port

01 device is duplicated on both A and AB buses. It requires 3 ~~machine~~ cycles:



Address space partitioning

Intel 8085 uses a 16 bit wide address bus for addressing memories and I/O devices. Using 16 bit wide address bus it can access $2^{16} = 64$ kB of memory and I/O devices. The 64k addresses are all assigned to memory and I/O devices for addressing.

The 2 schemes for allocation of addresses to memory & I/O devices:

1. memory mapped I/O
2. I/O mapped I/O (peripheral mapped I/O

memory mapped I/O

In memory mapped IO scheme, there is only one address space. address space is defined as the set of all possible addresses that a microprocessor can generate. Some addresses are assigned to memories and some addresses to I/O devices. An I/O device is treated as a memory location and one address is assigned to it.

To transfer data between mpu and I/O devices, memory related instructions such as LDA, STA etc are used. Control signals MEMR and MEMW are used. The microprocessor communicates with an I/O device as if it were one of the memory locations.

Eg: STA 8000H

if an o/p device, instead of a memory register is connected at this address, the accumulator contents will be transferred to the o/p device.

mov A,M (HL pair contains address of memory location)

In this scheme all data transfer instructions of the µp can be used for memory as well as I/O device.

I/O mapped I/O (peripheral mapped I/O

In this type of I/O, the µp uses 8 address lines to identify an i/p or o/p device. This is known as I/O mapped I/O. This is a 8 bit numbering system for I/O's used in conjunction with I/O and O/p instructions. This is also known as I/O space. seperate from memory space, which is a 16 bit numbering system.

The 8 address lines can have 256 (2^8 combinations) addresses, thus µp can identify 256 i/p devices and 256 o/p devices, with address ranging from 00H to FF H. The i/p and o/p devices are differentiated by control signals. The µp uses I/O read control for i/p devices and I/O write control signals for o/p devices. The entire range of I/O addresses from 00H to FF H is known as I/O map and individual addresses are referred to as I/O device addresses or I/O port numbers

Comparison

Characteristics	Memory mapped I/O	I/O mapped I/O
1. Device address	16 bit	8 bit
2. Control signals for I/p & O/p	MEMR, MEMW	IOR, IOW
3. Instructions available	STA, LDA, LDAX, STAX MOV M, R, ADD etc	IN & OUT
4. Data transfer	Between any register and I/O	Only between I/O and accumulator
5. Maximum no of I/O possible	memory map (64k) is shared between I/O's and S/M memory	256 I/p devices & 256 O/p devices can be connected
6. Execution speed	13 - 7 states 7 - 7 states	10 - 9 states
7. Hardware Requirements	more hardware needed to decode 16 bit addresses	less hardware needs to decode 8 bit address.

Mnemonic	Op	Description
ACI n	CE	Add with Carry Immediate
ADC r	8F	Add with Carry
ADC M	8E	Add with Carry to Memory
ADD r	87	Add
ADD M	86	Add to Memory
ADI n	C6	Add Immediate
ANA r	A7	AND Accumulator
ANA M	A6	AND Accumulator and Memory
ANI n	E6	AND Immediate
CALL a	CD	Call unconditional
CC a	DC	Call on Carry
CM a	FC	Call on Minus
CMA	2F	Complement Accumulator
CMC	3F	Complement Carry
CMP r	BF	Compare
CMP M	BF	Compare with Memory
CNC a	D4	Call on No Carry
CNZ a	C4	Call on No Zero
CP a	F4	Call on Plus
CPE a	EC	Call on Parity Even
CPI n	FE	Compare Immediate
CPO a	E4	Call on Parity Odd
CZ a	CC	Call on Zero
DAA	27	Decimal Adjust Accumulator
DAD B	09	Double Add BC to HL
DAD D	19	Double Add DE to HL
DAD H	29	Double Add HL to HL
DAD SP	39	Double Add SP to HL
DCR r	3D	Decrement
DCR M	35	Decrement Memory
DCX B	0B	Decrement BC
DCX D	1B	Decrement DE
DCX H	2B	Decrement HL

Mnemonic	Op	Description
DCX SP	3B	Decrement Stack Pointer
DI	F3	Disable Interrupts
EI	FB	Enable Interrupts
HLT	76	Halt
IN p	DB	Input
INR r	3C	Increment
INR M	3C	Increment Memory
INX B	03	Increment BC
INX D	13	Increment DE
INX H	23	Increment HL
INX SP	33	Increment Stack Pointer
JMP a	C3	Jump unconditional
JC a	DA	Jump on Carry
JM a	FA	Jump on Minus
JNC a	D2	Jump on No Carry
JNZ a	C2	Jump on No Zero
JP a	F2	Jump on Plus
JPE a	EA	Jump on Parity Even
JPO a	E2	Jump on Parity Odd
JZ a	CA	Jump on Zero
LDA a	3A	Load Accumulator direct
LDAX B	0A	Load Accumulator indirect
LDAX D	1A	Load Accumulator indirect
LHLD a	2A	Load HL Direct
LXI B,nn	01	Load Immediate BC
LXI D,nn	11	Load Immediate DE
LXI H,nn	21	Load Immediate HL
LXI SP,nr	31	Load Immediate Stack Ptr
MOV r1,r2	7F	Move register to register
MOV M,r	77	Move register to Memory
MOV r,M	7E	Move Memory to register
MVI r,n	3E	Move Immediate
MVI M,n	36	Move Immediate to Memory
NOP	00	No Operation

Mnemonic	Op	Description
ORA r	B7	Inclusive OR Accumulator
ORA M	B6	Inclusive OR Accumulator
ORI n	F6	Inclusive OR Immediate
OUT p	D3	Output
PCHL	E9	Jump HL indirect
POP B	C1	Pop BC
POP D	D1	Pop DE
POP H	E1	Pop HL
POP PSW	F1	Pop Processor Status Word
PUSH B	C5	Push BC
PUSH D	D5	Push DE
PUSH H	E5	Push HL
PUSH PSW	F5	Push Processor Status Word
RAL	17	Rotate Accumulator Left
RAR	1F	Rotate Accumulator Righ
RET	C9	Return
RC	D8	Return on Carry
RIM	20	Read Interrupt Mask
RM	F8	Return on Minus
RNC	D0	Return on No Carry
RNZ	C0	Return on No Zero
RP	F0	Return on Plus
RPE	E8	Return on Parity Even
RPO	E0	Return on Parity Odd
RZ	C8	Return on Zero
RLC	07	Rotate Left Circular
RRC	0F	Rotate Right Circular
RST z	C7	Restart
SBB r	9F	Subtract with Borrow
SBB M	9E	Subtract with Borrow
SBI n	DE	Subtract with Borrow Immed
SHLD a	22	Store HL Direct
SIM	30	Set Interrupt Mask
SPHL	F9	Move HL to SP
STA a	32	Store Accumulator
STAX B	02	Store Accumulator indirect
STAX D	12	Store Accumulator indirect
STC	37	Set Carry
SUB r	97	Subtract
SUB M	96	Subtract Memory
SUI n	D6	Subtract Immediate
XCHG	EB	Exchange HL with DE
XRA r	AF	Exclusive OR Accumulator
XRA M	AE	Exclusive OR Accumulator
XRI n	EE	Exclusive OR Immediate
XTHL	E3	Exchange stack Top with HL

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

Instruction Timing (T States)

Data Transfer		16 bit		Immediate		Specials		Control		Stack	
MOV rd,rs	4	DAD x	10	ADI	7	DAA	4	DI	4	PUSH	12
MOV M,r	7	INX x	6	ACI	7	CMA	4	EI	4	POP	10
MOV r,M	7	DCX x	6	SUI	7	STC	4	NOP	4		
XCHG	4			SBI	7	CMC	4	HLT	5		
MVI r	7	Rotate		ANI	7					Branch	
MVI M	10	RLC	4	XRI	7	I/O				RST x	12
LXI x	10	RRC	4	ORI	7	OUT	10			JMP	10
Load/Store		RAL	4	CPI	7	IN	10			Jx	7/10
LDAX x	7	RAR	4			RIM	4			CALL	18
LDA	13					SIM	4			Cx	9/18
STAX x	7									RET	10
SHLD	16									Rx	6/12
STA	13										
Arith & Logic										T-State	Time
ADD r	4									CPU f	tcyc
ADD M	7									8.0MHz	125nS
ADC r	4									6.0	166.7
ADC M	7									5.0	200
SUB r	4									4.0	250
SUB M	7									3.58	279
SBB r	4									3.125	320
SBB M	7									3.072	325.5
INR R	4									3.0	333.3
INR M	10									2.5	400
DCR r	4									2.0	500
DCR M	10									1.25	800
ANA r	4									1.2	833.3
ANA M	7									1.0	1000
XRA r	4									0.75	1333
XRA M	7										
ORA r	4										
ORA M	7										

Accumulator Operations

Instruction	Code	Function
XRA A	AF	Clear A and Clear Carry
CMA	2F	Complement Accumulator
ORA A	B7	Clear Carry
CMC	3F	Complement Carry
STC	37	Set Carry
RLC	7	Rotate Left, MSB=CY
RRC	0F	Rotate Right, LSB=>CY
RAL	17	Rotate Ledt Thru Carry
RAR	1F	Rotate Right Thru Carry

After RIM:

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

SID = Serial In Data
Ix.5 = Interrupt Pending
IE = Interrupt Enable Flag
Mx.5 = Interrupt Masks for external lines

Before SIM:

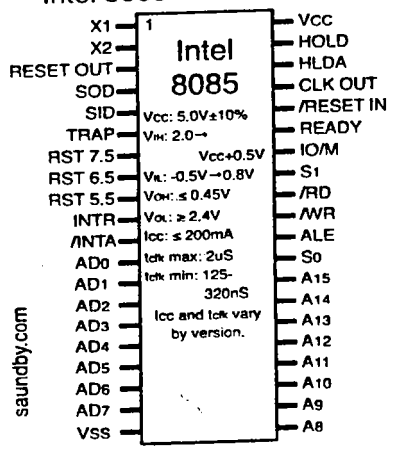
SOD	SOE	---	R7.5	MSE	M7.5	M6.5	M5.5

SOD = Serial Out Data
SOE = Serial Out Enable (enable SOD output)
R7.5= Reset (clear) RST 7.5 Interrupt
MSE = Mask Set Enable (1 to set new masks)
Mx.5= New RSTx.5 Mask Setting (1 to enable)

Note: RST5.5, RST6.5, RST7.5 are masked by default. Masks must be cleared before use.

Visit http://saundby.com/ for more 8085 info

Intel 8085 Reference Card



saundby.com

Register Organization

A(8)	F(8) PSW(16)
B(8)	C(8) B/C(16)
D(8)	E(8) D/E(16)
H(8)	L(8) H/L(16)
Program Counter	PC (16)
Stack Pointer	SP (16)

Flags

S	Z	[U]	AC	-	P	[V]	C

Sign, Zero, [U], Aux Carry, Parity, [V], Carry

* Undocumented/8085 only

Data Transfer Instruction Group

```
MOV A,A 7F   MOV E,A 5F   MVI A,b 3E
    A,B 78       E,B 58       B,b 06
    A,C 79       E,C 59       C,b 0E
    A,D 7A       E,D 5A       D,b 16
    A,E 7B       E,E 5B       E,b 1E
    A,H 7C       E,H 5C       H,b 26
    A,L 7D       E,L 5D       L,b 2E
    A,M 7E       E,M 5E       M,b 36

MOV B,A 47   MOV H,A 67   XCHG    EB
    B,B 40       H,B 60   LXI B,d 01
    B,C 41       H,C 61       D,d 11
    B,D 42       H,D 62       H,d 21
    B,E 43       H,E 63       SP,d 31
    B,H 44       H,H 64
    B,L 45       H,L 65   Load/Store
    B,M 46       H,M 66   LDAX B  0A
                          LDAX D  1A
MOV C,A 4F   MOV L,A 6F   LHLD a  2A
    C,B 48       L,B 68   LDA  a  3A
    C,C 49       L,C 69   [LHLX]  [ED]
    C,D 4A       L,D 6A   STAX B  02
    C,E 4B       L,E 6B   STAX D  12
    C,H 4C       L,H 6C   SHLD a  22
    C,L 4D       L,L 6D   STA  a  32
    C,M 4E       L,M 6E   [SHLX]  [D9]

MOV D,A 57   MOV M,A 77   arguments:
    D,B 50       M,B 70   b = byte data
    D,C 51       M,C 71   d = 16b data
    D,D 52       M,D 72   a = 16b address
    D,E 53       M,E 73   flags:
    D,H 54       M,H 73   * = affects all
    D,L 55       M,L 75   † = carry only
    D,M 56                ¶ = all but carry
```

Arithmetic & Logical Instruction Group

```
ADD* A 87    INR† A 3C    ORA* A B7
     B 80         B 04         B B0
     C 81         C 0C         C B1
     D 82         D 14         D B2
     E 83         E 1C         E B3
     H 84         H 24         H B4
     L 85         L 2C         L B5
     M 86         M 34         M B6

ADC* A 8F    DCR† A 3D    CMP* A BF
     B 88         B 05         B B8
     C 89         C 0D         C B9
     D 8A         D 15         D BA
     E 8B         E 1D         E BB
     H 8C         H 25         H BC
     L 8D         L 2D         L BD
     M 8E         M 35         M BE

SUB* A 97    ANA* A A7    16 Bit
     B 90         B A0    DAD† B 09
     C 91         C A1         D 19
     D 92         D A2         H 29
     E 93         E A3         SP 39
     H 94         H A4    INX  B 03
     L 95         L A5         D 13
     M 96         M A6         H 23
                               SP 33
SBB* A 9F    XRA* A AF    DCX  B 0B
     B 98         B A8         D 1B
     C 99         C A9         H 2B
     D 9A         D AA         SP 3B
     E 9B         E AB
     H 9C         H AC    [DSUB]  [08]
     L 9D         L AD
     M 9E         M AE
```

Arith & Logic continued -->

Arith & Logic

```
Rotate
RLC†   07
RRC†   0F
RAL†   17
RAR†   1F
[AHRL] [10]

Immediate
ADI  b C6
ACI  b CE
SUI  b D6
SBI  b DE
ANI  b E6
XRI  b EE
ORI  b F6
CPI  b FE
[LDHI b] [28]
[LDSI b] [38]

Specials
DAA*   27
CMA    2F
STC†   37
CMC†   3F
```

I/O & Control

```
OUT  b D3
IN   b DB

RIM    20
SIM    30

Control
DI     F3
EI     FB
NOP    00
HLT    76
```

Stack

```
PUSH B  C5
     D  D5
     H  E5
     PSW F5
POP  B  C1
     D  D1
     H  E1
     PSW* F1
XTHL    E3
SPHL    F9
```

Branch

```
Restart
RST 0   C7
RST 1   CF
RST 2   D7
RST 3   DF
RST 4   E7
RST 5   EF
RST 6   F7
RST 7   FF
[RST V] [CB]

Restart Vector
RST 0   0000H
RST 1   0008H
RST 2   0010H
RST 3   0018H
RST 4   0020H
TRAP    0024H
RST 5   0028H
RST 5.5 002CH
RST 6   0030H
RST 6.5 0034H
RST 7   0038H
RST 7.5 003CH
```

Branch

```
Jump
JMP  a C3
JNZ  a C2
JZ   a CA
JNC  a D2
JC   a DA
JPO  a E2
JPE  a EA
JP   a F2
JM   a FA
PCHL   E9
[JNUI] [DD]
[JUI]  [FD]

Call
CALL a CD
CNZ  a C4
CZ   a CC
CNC  a D4
CC   a DC
CPO  a E4
CPE  a EC
CP   a F4
CM   a FC

Return
RET    C9
RNZ    C0
RZ     C8
RNC    D0
RC     D8
RPO    E0
RPE    E8
RP     F0
RM     F8
```