8085 INTRODUCTION

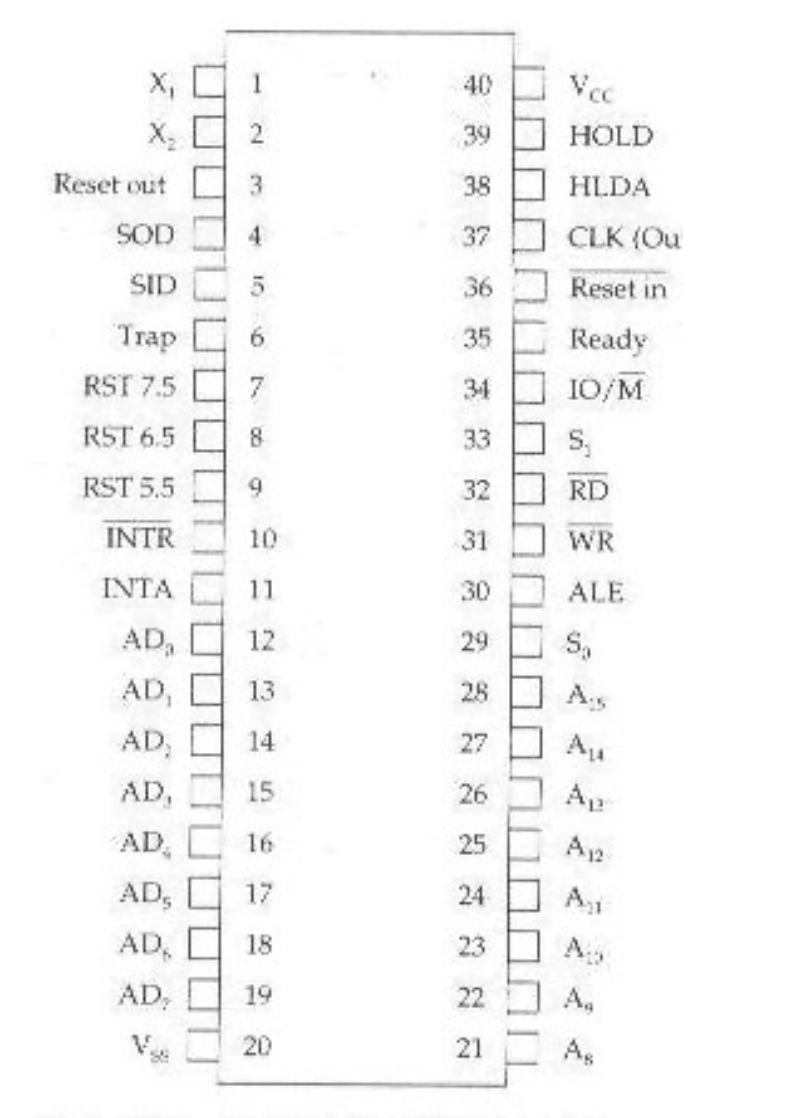
The features of INTEL 8085 are :

- It is an 8 bit processor.
- It is a single chip N-MOS device with 40 pins.
- It has multiplexed address and data bus.(AD_0-AD_7).
- It works on 5 Volt dc power supply.
- The maximum clock frequency is 3 MHz while minimum frequency is 500kHz.
- It provides 74 instructions with 5 different addressing modes.

8085 INTRODUCTION

- It provides 16 address lines so it can access 2^16 =64K bytes of memory.
- It generates 8 bit I/O address so it can access 2^8=256 input ports.
- It provides 5 hardware interrupts:TRAP, RST 5.5, RST 6.5, RST 7.5,INTR.
- It provides Acc ,one flag register ,6 general purpose registers and two special purpose registers(SP,PC).
- It provides serial lines SID ,SOD.So serial peripherals can be interfaced with 8085 directly.

8085 PIN DIAGRAM



8085 PIN DESCRIPTION

Some important pins are :

- AD_0-AD_7: Multiplexed Address and data lines.
- A_8-A_{15}: Tri-stated higher order address lines.
- ALE: Address latch enable is an output signal.It goes high when operation is started by processor .
- S0,S1: These are the status signals used to indicate type of operation.
- $\overline{RD}$: Read is active low input signal used to read data from I/O device or memory.
- $\overline{WR}$:Write is an active low output signal used write data on memory or an I/O device.

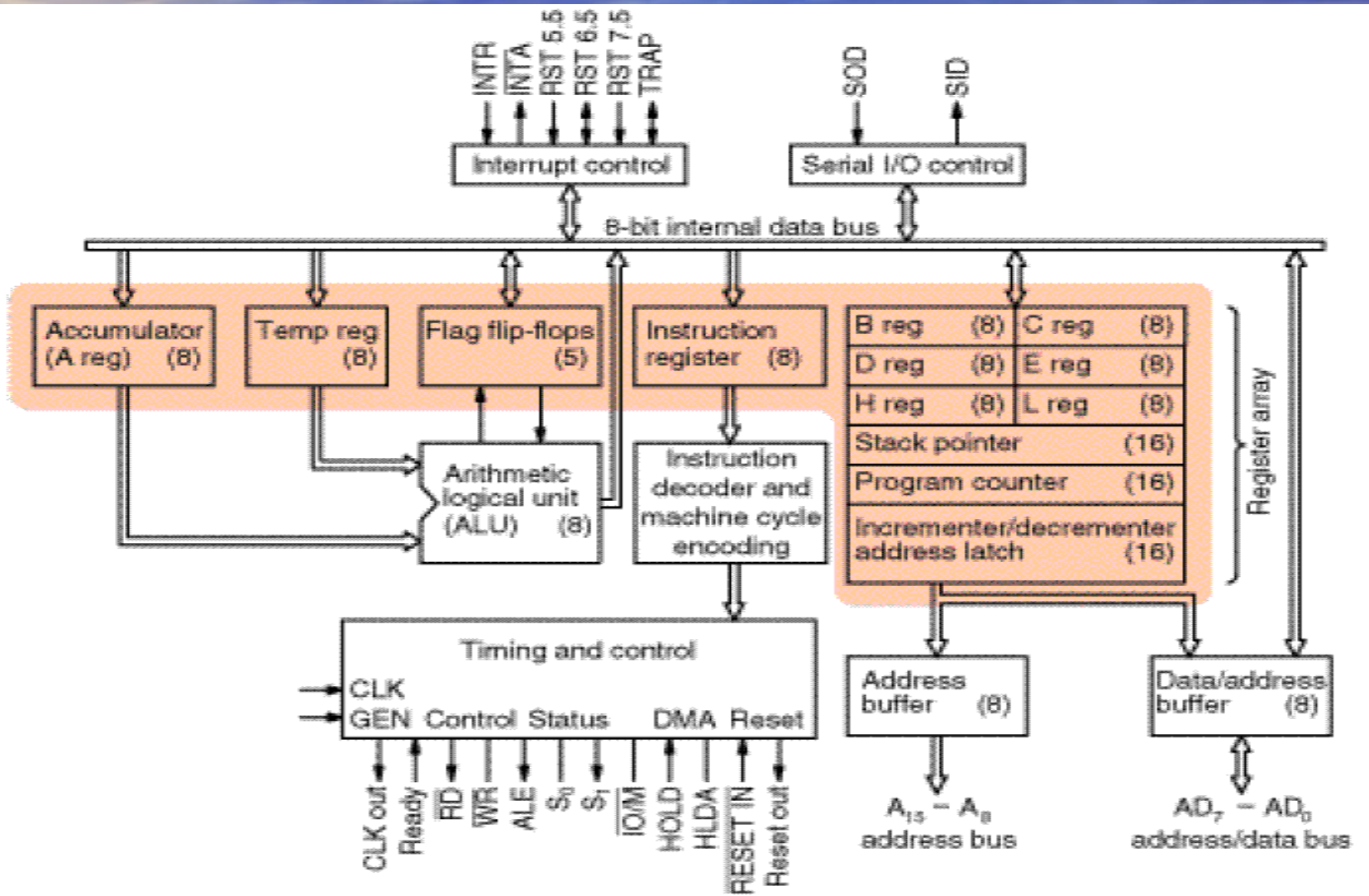
8085 PIN DESCRIPTION

- READY:This an output signal used to check the status of output device.If it is low, µP will WAIT until it is high.
- TRAP:It is an Edge triggered highest priority , non mask able interrupt. After TRAP, restart occurs and execution starts from address 0024H.
- RST5.5,6.5,7.5:These are maskable interrupts and have low priority than TRAP.
- $\overline{\text{INTR}}$&$\overline{\text{INTA}}$:INTR is a interrupt request signal after which µP generates INTA or interrupt acknowledge signal.
- IO/$\overline{\text{M}}$:This is output pin or signal used to indicate whether 8085 is working in I/O mode(IO/$\overline{\text{M}}$=1) or Memory mode(IO/$\overline{\text{M}}$=0).

8085 PIN DESCRIPTION

- HOLD&HLDA:HOLD is an input signal .When µP receives HOLD signal it completes current machine cycle and stops executing next instruction.In response to HOLD µP generates HLDA that is HOLD Acknowledge signal.
- RESET $\overline{\text{IN}}$:This is input signal.When RESET $\overline{\text{IN}}$ is low µp restarts and starts executing from location 0000H.
- SID: Serial input data is input pin used to accept serial 1 bit data .
- X_1X_2 :These are clock input signals and are connected to external LC,or RC circuit.These are divide by two so if 6 MHz is connected to X_1X_2, the operating frequency becomes 3 MHz.
- VCC&VSS:Power supply VCC=+ -5Volt& VSS=-GND

8085 ARCHITECTURE



Arithmetic and Logical group

Accumulator: It is 8 bit general purpose register.

- It is connected to ALU.
- So most of the operations are done in Acc.

Temporary register: It is not available for user

- All the arithmetic and logical operations are done in the temporary register but user can't access it.

Flag: It is a group of 5 flip flops used to know status of various operations done.

- The Flag Register along with Accumulator is called PSW

or Program Status Word.

Arithmetic and Logical group

Flag Register is given by:

S	Z	X	AC	X	P	X	CY

S:Sign flag is set when result of an operation is negative.

Z:Zero flag is set when result of an operation is 0.

Ac:Auxiliary carry flag is set when there is a carry out of lower nibble or lower four bits of the operation.

CY:Carry flag is set when there is carry generated by an operation.

P:Parity flag is set when result contains even number of 1's.

Rest are don't care flip flops.

Register Group

- Temporary registers (W,Z):These are not available for user. These are loaded only when there is an operation being performed.
- General purpose:There are six general purpose registers in 8085 namely B,C,D,E,H,L.These are used for various data manipulations.
- Special purpose :There are two special purpose registers in 8085:
- SP :Stack Pointer.
- PC:Program Counter.

Register Group

Stack Pointer: This is a temporary storage memory 16 bit register. Since there are only 6 general purpose registers, there is a need to reuse them .

- Whenever stack is to be used previous values are PUSHED on stack and then after the program is over these values are POPED back.

Program Counter: It is 16 bit register used to point the location from which the next instruction is to be fetched.

- When a single byte instruction is executed PC is automatically incremented by 1.
- Upon reset PC contents are set to 0000H and next instruction is fetched onwards.

INSTRUCTION REGISTER,DECODER & CONTROL

- Instruction register:When an instruction is fetched , it is executed in instruction register.This register takes the Opcode value only.
- Instruction decoder: It decodes the instruction from instruction register and then to control block.
- Timing and control:This is the control section of μP.It accepts clock input .

INTERRUPT CONTROL

- It accepts different interrupts like TRAP INT5.5,6.5,7.5and INTR.

SERIAL IO CONTROL GROUP

- It is used to accept the serial 1 bit data by using SID and SOD signals and it can be performed by using SIM & RIM instructions.

INSTRUCTIONS SET OF 8085

DATA TRANSFER GROUP

MOV Rd, Rs.(Move data from Rs to Rd).

Example:

MOV C,B. Move the content of register B to C.

Initially	After execution
B=10H.	B=10H.
C=20H.	C=10H.

Flags Affected :No flags affected.

Addressing mode: Register.

DATA TRANSFER GROUP

MOV Rd, M (Move data from Memory to Rd).

Example:

MOV C,M. Move the content of Memory i.e. "H or L" to C.

Suppose the Data at memory pointed By HL pair at C200H is 10H.

Initially	After execution
H=C2,L=00,C=30H	H=C2,L=00,C=10H.

Flags Affected :No flags affected.

Addressing mode: Indirect.

DATA TRANSFER GROUP

MVI R, Data.(Move Immediate data to Register).

Example:

MVI B, 30H. (Move the data 30 H to Register B)

Initially | After execution

B=40H | B=30H

Flags Affected :No flags affected.

Addressing mode: Immediate.

DATA TRANSFER GROUP

LXI Rp,16 bit .(Load 16 bit data to Register pair Immediate).

Example:

LXI SP, C200H. (Load Stack pointer with C200H).

Initially	After execution
SP=C800H	SP=C200H.

Flags Affected :No flags affected.

Addressing mode: Immediate.

DATA TRANSFER GROUP

STA address.(Store Acc data to address).

Example:

STA C200H. (Move the data from Acc to C200H).

Suppose in Acc the data is 10H.

Initially After execution

A=10H, C200=20H C200=10H , A=10H

Flags Affected :No flags affected.

Addressing mode: Direct.

DATA TRANSFER GROUP

LHLD address.(Load HL pair with data from address).
Example:
LHLD C200H. (Move the data from C200 to HL pair).
Suppose at C200 the data is 20H,30H .

Initially	After execution
H=10H,L=20H	H=20H,L=30H.
C2=20H,00=30H	C2=20H,00=30H

Flags Affected :No flags affected.
Addressing mode: Direct.

DATA TRANSFER GROUP

- XCHG (Exchange the data from HL pair to DE pair)

Example : XCHG

Initially execution	After
H=20H,L=30H,	H=40H,L=70H.
D=40H,E=70H.	D=20H,E=30H.

Flags Affected :No flags affected.

Addressing mode: Register.

DATA TRANSFER GROUP

IN 8 bit address (Move the data from address to Acc)

Example: IN 80H

Move the data from 80H port address to Accumulator.

Suppose data at 80H is 39H.

Initially execution — After

A=20H. — A=39H

Flags Affected :No flags affected.

Addressing mode: Direct.

DATA TRANSFER GROUP

OUT 8 bit address (Move the data from Acc to address)

Example: OUT 80H

Move the data from Acc to port address 80H.

Suppose data at Acc is 39H.

Initially After execution

A=39H. 80=10H.
A=39H,80=39H.

Flags Affected :No flags affected.

Addressing mode: Direct.

DATA TRANSFER GROUP

- Example:Write a program to exchange contents of memory location D000H to D001H

LDA D000H	Load Acc with data from D000
MOV B,A	Move the data to B
LDA D0001H	Load Acc with data from D001
STA 2000H	Store Acc data at D000
MOV A,B	Move B's data to A
STA 2001H	Store data from D000 to

ARITHMETIC GROUP

ADD R (ADD register content with Acc and result in A).

Example:

ADD C. (ADD the content of C with A).

Suppose the Data at C register is 10H.

Initially After execution

. C= 10H ,A=10H A=20H,C=10H.

Flags Affected :All flags are modified.

Addressing mode: Register

ARITHMEIC GROUP

ADD M(ADD H or L Reg content with Acc and result in A).

Example:

ADD M. (ADD the content of HL with A).

- Suppose the Data at memory pointed by HL register 1020H is 10H.

Initially	After execution
. H= 10H ,L=20H .	H=10H,L=20H.
A=20H,C=10H.	A=30H.

Flags Affected :All flags are modified.

Addressing mode: Register Indirect.

ARITHMETIC GROUP

ADI Data(ADD immediate data with Acc and result in A).

Example:

ADI 30H. (ADD 30H with A).

Initially After execution

A=20H, A=50H.

Flags Affected :All flags are modified.

Addressing mode: Immediate.

ARITHMETIC GROUP

ADC R (ADD register content with Acc and carry and result in A).

Example:

ADC C. (ADD the content of C with A with carry).

Suppose the Data at C register is 10H and carry is 01H.

Initially After execution

. C= 10H ,A=10H A=21H,C=10H.

Flags Affected :All flags are modified.

Addressing mode: Register

ARITHMETIC GROUP

Example: Write a program to perform 16 bit addition of 1234H& 4321H. Store answer at H & L registers.

MVI B,21H	B=21H
MVI A,34H	A=34H
MVI C,43H	C=43H
MVI D,12H	D=12H
ADD B	A=34+21H
MOV L,A	L=55H
MOV A,C	A=43H
ADC D	A=43+12H
MOV H,A	H=55H
RST1	STOP.

ARITHMETIC GROUP

SUB R (Subtract register content from Acc and result in A).

Example:

SUB B. (Subtract the content of B from A).

Suppose the Data at B register is 10H .

Initially After execution

. B= 10H ,A=20H A=10H,B=10H.

Flags Affected :All flags are modified.

Addressing mode: Register

ARITHMETIC GROUP

SBB R (Subtract register content from Acc with borrow and result in A).

Example:

SBB B. (Subtract the content of B from A with borrow).

Suppose the Data at B register is 10H and borrow is 01H .

Initially After execution

. B= 0FH ,A=20H A=10H,B=0FH.

Flags Affected :All flags are modified.

Addressing mode: Register

ARITHMETIC GROUP

SUI Data(Subtract immediate data from Acc and result in A).

Example:

SUI 30H. (Subtract 30H from A).

Initially After execution

A=80H, A=50H.

Flags Affected :All flags are modified.

Addressing mode: Immediate

ARITHMETIC GROUP

Example: Subtract data of C800 H from C200H.Store the result at 2C00.

```
LDA C800H
MOV B,A
LDA C200H
SUB B
STA 2C00H
RST1
```

ARITHMETIC GROUP

DAD Rp (Add specified register pair with HL pair)

Example:DAD D.(Add the content of E with L and that of D with H register and result in HL pair)

- Suppose the content of HL pair is H=20H ,L=40H and DE pair is D=30H, E=10H.

Initially	After execution
H=20H ,L=40H	H=50H ,L=50H
D=30H, E=10H	D=30H, E=10H

Flags Affected :Only carry flag is modified.

Addressing mode: Register.

ARITHMETIC GROUP

DAA (Decimal adjust accumulator)

Example:

MVI A,12H

ADI 39H

DAA .

- This instruction is used to store result in BCD form.If lower nibble is greater than 9 ,6 is added while if upper nibble is greater than 9,6 is added to it to get BCD result.

Initially After execution

12+39=4B 12+39=51 in BCD form.

Flags Affected :All flags are modified.

Addressing mode: Register

ARITHMETIC GROUP

INR R (Increment register content by 1).

Example:

INR C. (Increment the content of C by 1).

Suppose the Data at C register is 10H.

Initially	After execution
C= 10H	C=11H.

Flags Affected :All flags are modified except carry flag.

Addressing mode: Register.

ARITHMETIC GROUP

DCR R (Decrement register content by 1).

Example:

DCR C. (Decrement the content of C by 1).

Suppose the Data at C register is 10H.

Initially	After execution
C= 10H	C=0FH.

Flags Affected :All flags are modified except carry flag.

Addressing mode: Register.

ARITHMETIC GROUP

INX Rp (Increment register pair content by 1).

Example:

INX SP (Increment the content of Stack pointer pair by 1).

INX B. (Increment the content of BC pair by 1).

Suppose the Data at BC register is 1010H and SP is C200H

Initially	After execution
BC= 1010H	BC=1011H.
SP=C200H	SP=C201H.

Flags Affected :No flags are modified.

Addressing mode: Register.

LOGICAL GROUP

ANA R (Logically AND register content with Acc and result in A).

Example:

ANA C (AND the content of C with A).

Suppose the Data at C register is 10H.

Initially After execution

C= 10H ,A=10H A=10H,C=10H.

Flags Affected :S,Z,P are modified Cy=reset,AC=set.

Addressing mode:Register.

LOGICAL GROUP

ANI Data (Logically AND immediate data with Acc and result in A).

Example:

ANI 10H (AND 10H with A).

Initially	After execution
A=10H	A=10H

Flags Affected :S,Z,P are modified Cy=reset,AC=set.

Addressing mode: Immediate.

LOGICAL GROUP

ORA R (Logically OR register content with Acc and result in A5).

Example:

ORA C (OR the content of C with A).

Suppose the Data at C register is 17H.

Initially After execution

C= 17H ,A=10H A=17H,C=17H.

Flags Affected :S,Z,P are modified Cy=reset,AC=reset.

Addressing mode:Register.

LOGICAL GROUP

ORI Data (Logically OR immediate data with Acc and result in A).

Example:

ORI 10H (OR 10H with A).

Initially	After execution
A=30H	A=30H

Flags Affected :S,Z,P are modified Cy=reset,AC=set.

Addressing mode: Immediate.

LOGICAL GROUP

XRA R (Logically XOR register content with Acc and result in A).

Example:

XRA C (XOR the content of C with A).

Suppose the Data at C register is 17H.

Initially After execution

C= 17H ,A=10H A=07H,C=17H.

Flags Affected :S,Z,P are modified Cy=reset,AC=reset.

Addressing mode:Register.

LOGICAL GROUP

CMP R (Compare register content with Acc and result in A).

Example:

CMP C (Compare the content of C with A).

Suppose the Data at C register is 17H.

Initially	After execution
C= 10H ,A=17H	A=17H,C=17H.

Flags Affected :S=0,Z=0,P=0, Cy=reset,AC=reset.

Addressing mode:Register.

LOGICAL GROUP

CPI Data (Compare immediate data with Acc).

Example:

CPI 10H (Compare the content of C with A).

Initially After execution

A=17H A=17H.

Flags Affected :S=0,Z=0,P=0, Cy=reset,AC=reset.

Addressing mode:Immediate.

LOGICAL GROUP

RLC (Rotate accumulator left).

Example:

MOV A,03H.

RLC (Rotate accumulator left).

Initially After execution

A=03H A=06H.

Flags Affected :Only carry flag is affected.

Addressing mode:Implied.

LOGICAL GROUP

RAL (Rotate accumulator left with carry).

Example:

MOV A,03H.

RAL (Rotate accumulator left with carry).

Initially After execution

A=03H , carry =01H A=07H.

Flags Affected :Only carry flag is affected.

Addressing mode:Implied.

LOGICAL GROUP

RRC (Rotate accumulator right).

Example:

MOV A,03H.

RRC (Rotate accumulator right).

Initially After execution

A=03H , A=81H.

Flags Affected :Only carry flag is affected.

Addressing mode:Implied.

LOGICAL GROUP

Write a program to reset last 4 bits of the number 32H
Store result at C200H.

MVI A, 32H	A=32H
ANI F0H 1111000	00110010 AND
	=00110000=30H
STA C200H.	C200=30H
RST1	Stop

BRANCH GROUP

JMP address(Unconditional jump to address)

Example:

JMP C200H.

- After this instruction the Program Counter is loaded with this location and starts executing and the contents of PC are loaded on Stack.

Flags Affected :No Flags are affected.

Addressing mode:Immediate.

CALL address(Unconditional CALL from address)

Example:

CALL C200H.

- After this instruction the Program Counter is loaded with this location and starts executing and the contents of PC are loaded on Stack.

Flags Affected :No Flags are affected.

Addressing mode:Immediate

BRANCH GROUP

Conditional Jump Instructions.

- JC (Jump if Carry flag is set)
- JNC (Jump if Carry flag is reset)
- JZ (Jump if zero flag set)
- JNZ (Jump if zero flag is reset)
- JPE (Jump if parity flag is set)
- JPO (Jump if parity odd or P flag is reset)
- JP (Jump if sign flag reset)
- JM (Jump if sign flag is set or minus)

BRANCH GROUP

Conditional Call Instructions.

- CC (Call if Carry flag is set)
- CNC (Call if Carry flag is reset)
- CZ (Call if zero flag set)
- CNZ (Call if zero flag is reset)
- CPE (Call if parity flag is set)
- CPO (Call if parity odd or P flag is reset)
- CP (Call if sign flag reset)
- CM (Call if sign flag is set or minus)

BRANCH GROUP

RET (Return from subroutine)

Example:

MOV A,C

RET

- After this instruction the Program Counter POPS PUSHED contents from stack and starts executing from that address .

Flags Affected :No Flags are affected.

Addressing mode:Register indirect .

BRANCH GROUP

RST (Restart instruction)

Example:

MOV A,C

RST 1.

- After this instruction the Program Counter goes to address 0008H and starts executing from that address .

Flags Affected :No Flags are affected.

Addressing mode:Register indirect.

BRANCH GROUP

The addresses of the respective RST commands are:

Instruction	Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

STACK AND MACHINE CONTROL

PUSH Rp.(Push register pair contents on stack).

Example:LXI SP FFFFH.

PUSH H. (Move the content of HL pair on Stack).

- Suppose at HL pair the data is H= 20H,L= 30H & SP is initialized at FFFFH

Initially

H=20H,L=30H

SP=FFFF H

FFFD=30H,FFFE=20H

After execution

H=20H,L=30H.

Flags Affected :No flags affected.

Addressing mode: Register indirect.

STACK AND MACHINE CONTROL

POP Rp.(Pop register pair contents from stack).

Example:POP D(POP the content of DE pair from Stack).

- Suppose at DE pair the data is H= 20H,L= 30H SP was initialized at FFFFH

Initially

D=20H,E=30H

FFFD=80H,FFFE=10H

After execution

D=10H,E=80H.

Flags Affected :No flags affected.

Addressing mode: Register indirect

STACK AND MACHINE CONTROL

XTHL (Exchange HL register pair contents with top of stack).

Example:XTHL(Exchange top with HL pair).

- Suppose at HL pair the data is H= 20H,L= 30H & SP =FFFFH

& at locations FFFF=10H and at FFFE= 80H.

Initially	After execution
H=20H,L=30H	H=10H,L=80H.
SP=FFFF =10H,FFFE=80H	FFFD=20H,FFFE=30H

Flags Affected :No flags affected.

Addressing mode: Register indirect.

ADDRESSING MODES OF 8085

Immediate addressing:

Immediate data is transferred to address or register.

Example:

MVI A,20H. Transfer immediate data 20H to accumulator.

Number of bytes:

Either 2 or 3 bytes long.

1st byte is opcode.

2nd byte 8 bit data .

3rd byte higher byte data of 16 bytes.

ADDRESSING MODES OF 8085

Register addressing:

Data is transferred from one register to other.

Example:

MOV A, C :Transfer data from C register to accumulator.

Number of bytes:

Only 1 byte long.

One byte is opcode.

ADDRESSING MODES OF 8085

Direct addressing:

- Data is transferred from direct address to other register or vice-versa.

Example:

LDA C200H .Transfer contents from C200H to Acc.

Number of bytes:

These are 3 bytes long.

1st byte is opcode.

2nd byte lower address.

3rd byte higher address.

ADDRESSING MODES OF 8085

Indirect addressing:

- Data is transferred from address pointed by the data in a register to other register or vice-versa.

Example:

MOV A, M: Move contents from address pointed by M to Acc.

Number of bytes:

These are 3 bytes long.

1st byte is opcode.

2nd byte lower address.

3rd byte higher address.

ADDRESSING MODES OF 8085

Implied addressing:

- These doesn't require any operand. The data is specified in Opcode itself.

Example: RAL: Rotate left with carry.

No.of Bytes:

These are single byte instruction or Opcode only.

PROGRAM

- Write a program to transfer a block of data from C550H to C55FH. Store the data from C570H to C57FH .

```
    LXI H ,C550H
    LXI B ,C570H
    MVI D,0FH
UP MOV A,M
    STAX B
    INX H
    INX B
    DCR D
    JNZ  UP
     RST1
```

PROGRAM

- Find out errors in the following :
- MVI B,D =Immediate addressing doesn't have register as operand .Therefore, MVI B,80H.
- INX L=Increment operator always acts on the higher memory address in register pair .Thus ,INX H.
- JP 80H = Conditional jump instructions doesn't have any immediate operand .Thus, JP UP.

If Flag contents are AB H, what is flag status

If flag contains AB H then it's values from D_7 to D_0 are 10101011.

By comparing it with flag register we get S=1,Z=0,AC=0, P=0,Cy=1.

PROGRAM

11. What are the instructions for the following actions?

- Load the PC with second and third byte of instruction.

LXI H, C200H

PCHL Load PC with HL content

Thus PC= L,PC +1=H.

- No change in normal execution except increment the PC.

NOP (No operation)

- This instruction has no effect on code only used to cause delay .

PROGRAM

Write a program to add 10 data bytes. Data is stored from locations C200. Store result at C300H.

```
        LXI H,C200 H
        MVI C, 0A H
UP      MVI A,00 H
        MOV B,M
        ADD B
        INX H
        DCR C
        JNZ UP
        STA C300H
        RST1.
```

TIMING AND STATE DIAGRAM

- The µP operates with reference to clock signal.The rise and fall of the pulse of the clock gives one clock cycle.
- Each clock cycle is called a T state and a collection of several T states gives a machine cycle.
- Important machine cycles are :
- Op-code fetch.
- Memory read.
- Memory write.
- I/Op-read.
- I/O write.

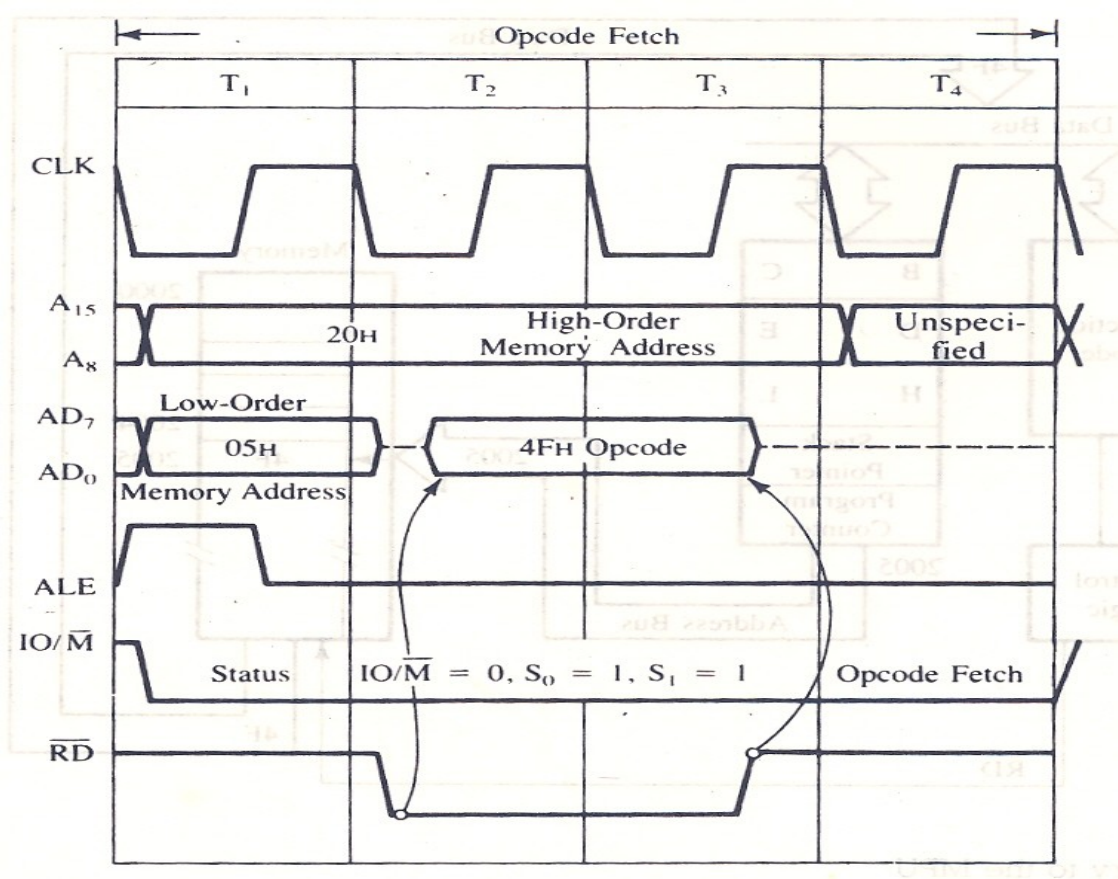
TIMING AND STATE DIAGRAM

Op-code Fetch:It basically requires 4 T states from T_1-T_4

- The ALE pin goes high at first T state always.
- AD_0-AD_7 are used to fetch OP-code and store the lower byte of Program Counter.
- A_8-A_{15} store the higher byte of the Program Counter while IO/$\overline{M}$ will be low since it is memory related operation.
- $\overline{RD}$ will only be low at the Op-code fetching time.
- $\overline{WR}$ will be at HIGH level since no write operation is done.
- S_0=1,S_1=1 for Op-code fetch cycle.

TIMING AND STATE DIAGRAM

Op-code fetch cycle :



TIMING AND STATE DIAGRAM

Memory Read Cycle: It basically requires 3T states from T_1-T_3

- The ALE pin goes high at first T state always.
- AD_0-AD_7 are used to fetch data from memory and store the lower byte of address.
- A_8-A_{15} store the higher byte of the address while IO/$\overline{M}$ will be low since it is memory related operation.
- $\overline{RD}$ will only be low at the data fetching time.
- $\overline{WR}$ will be at HIGH level since no write operation is done.
- S_0=0,S_1=1 for Memory read cycle.

TIMING AND STATE DIAGRAM

Memory write Cycle: It basically requires 3T states from T_1-T_3 .

- The ALE pin goes high at first T state always.
- AD_0-AD_7 are used to fetch data from CPU and store the lower byte of address.
- A_8-A_{15} store the higher byte of the address while $IO/\overline{M}$ will be low since it is memory related operation.
- $\overline{RD}$ will be HIGH since no read operation is done.
- $\overline{WR}$ will be at LOW level only when data fetching is done.
- S_0=1,S_1=0 for Memory write cycle.

SUBROUTINE

Calculation of Delay using 8 bit counter:

- Consider following example:

```
    MVI C, count(8 bit) H                    7 T states

UP DCR C                                     4 T states
    JNZ UP                                   10/7 T
    RET                                10T
```

- Here loop UP is executed (N-1) times.
- Thus delay is

Td=M+[(count)x N)-3.

- Where M= no.of T states outside loop.

  N=no.of T states inside loop.

SUBROUTINE

- Here value of M= 17, N= 14.
- The maximum delay will occur if count is 255 or FF H.
- Thus Td max =17+[255x14]-3= 3584 T states.
- For 0.5 μsec delay for a T state, we get
- Td max=0.5 μsec x 3584= 1792 μsec or 1.792 m sec.

8085 Memory Interfacing

- Generally µP 8085 can address 64 kB of memory .
- Generally EPROMS are used as program memory and RAM as data memory.
- We can interface Multiple RAMs and EPROMS to single µP .
- Memory interfacing includes 3 steps :

5. Select the chip.
6. Identify register.
7. Enable appropriate buffer.

8085 Memory Interfacing

- Example: Interface 2Kbytes of Memory to 8085 with starting address 8000H.

Initially we realize that 2K memory requires 11 address lines

(2^11=2048). So we use A_0-A_{10} .

- Write down A_{15} –A_0

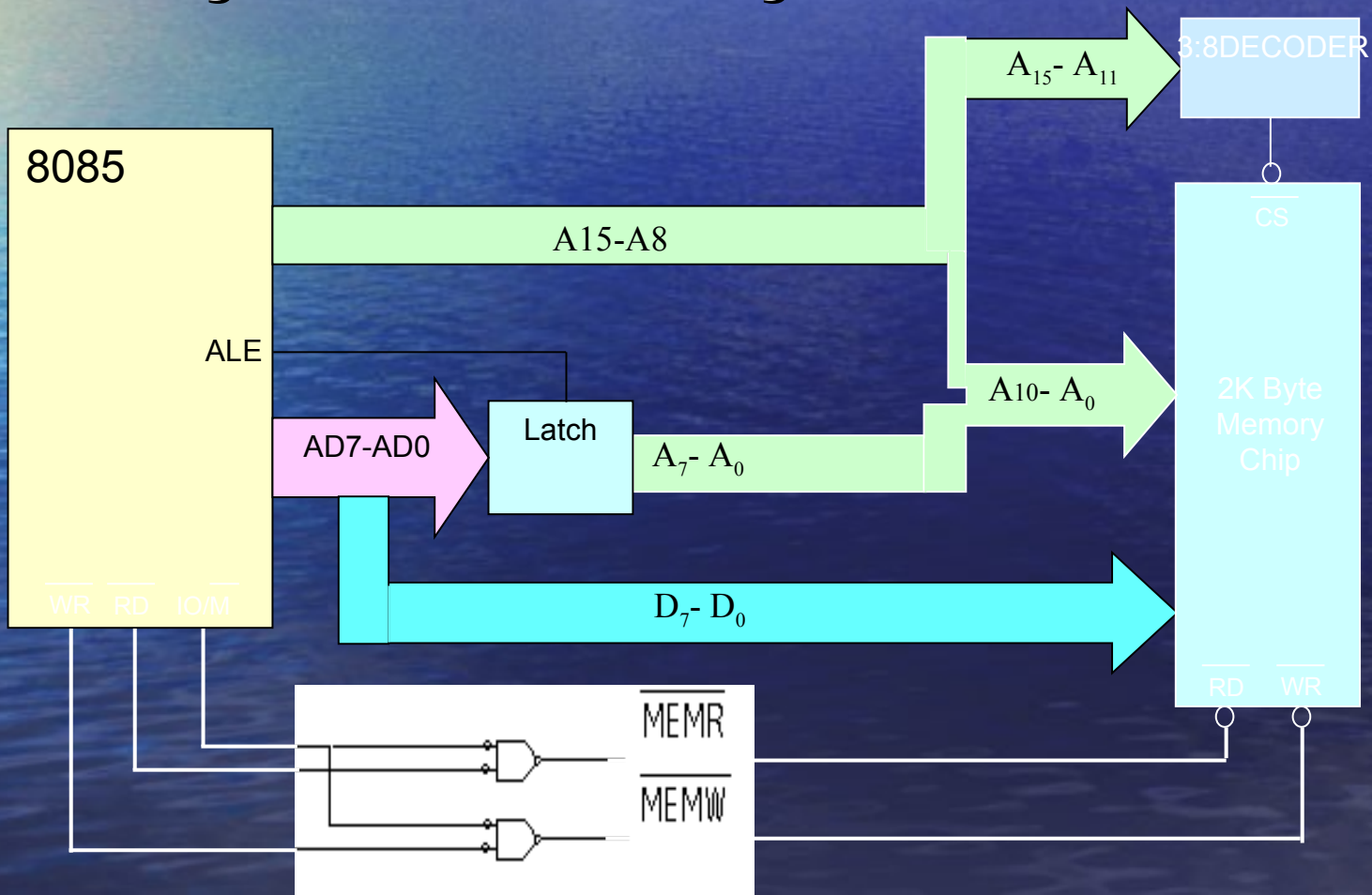
A_{15}	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ADD
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	87FFH

8085 Memory Interfacing

- Address lines A_0-A_{10} are used to interface memory while A_{11},A_{12},A_{13},A_{14},A_{15} are given to 3:8 Decoder to provide an output signal used to select the memory chip $\overline{CS}$ or Chip select input.
- $\overline{MEMR}$ and $\overline{MEMW}$ are given to $\overline{RD}$ and $\overline{WR}$ pins of Memory chip.
- Data lines D_0-D_7 are given to D_0-D_7 pins of the memory chip.
- In this way memory interfacing can be achieved.

8085 Memory Interfacing

- The diagram of 2k interfacing is shown below:

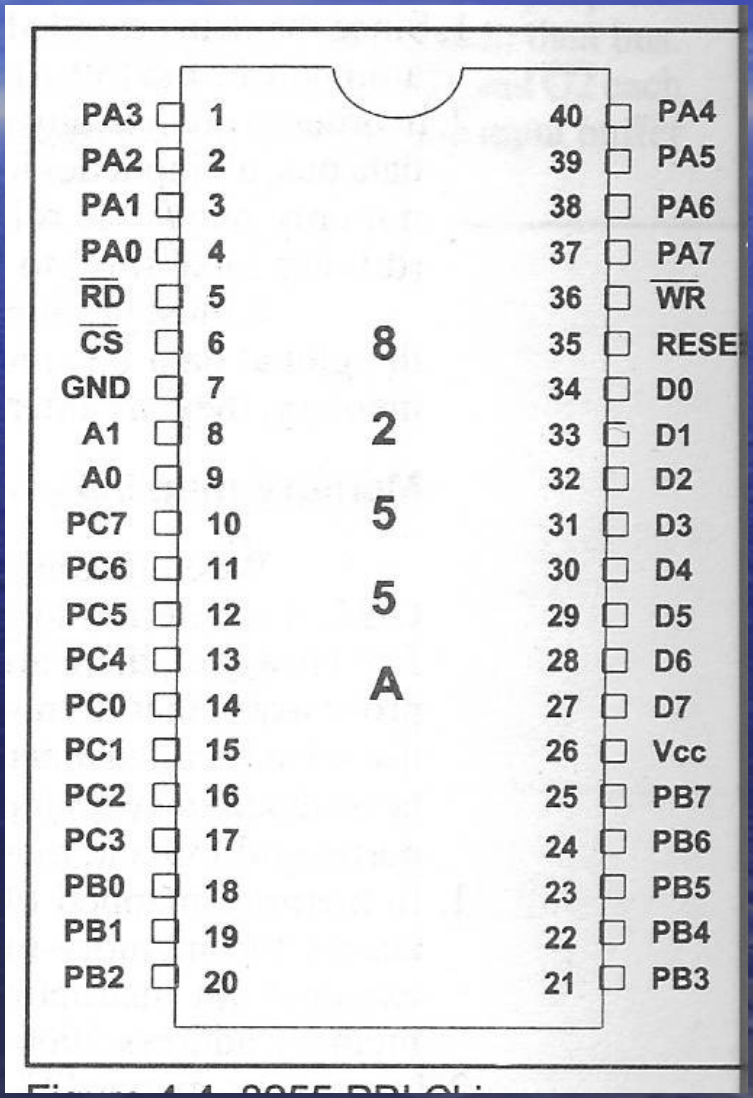


8085 Memory Interfacing

- In this example we saw that some address lines are used for interfacing while others are for decoding.
- It is called absolute decoding.
- We sometimes don't requires that many address lines.So we ignore them.But this may lead to shadowing or multiple address.
- This type of decoding is called linear decoding or partial decoding.
- In partial decoding wastage of address takes place but it requires less hardware and cost is also less as compared with absolute one.

8255 PIN DIAGRAM

PA0-PA7	I/O	Port A Pins
PB0-PB7	I/O	Port B Pins
PC0-PC7	I/O	Port C Pins
D0-D7	I/O	Data Pins
RESET	I	Reset pin
$\overline{RD}$	I	Read input
$\overline{WR}$	I	Write input
A0-A1	I	Address pins
$\overline{CS}$	I	Chip select
Vcc , Gnd	I	+5volt supply



8255 BLOCK DIAGRAM

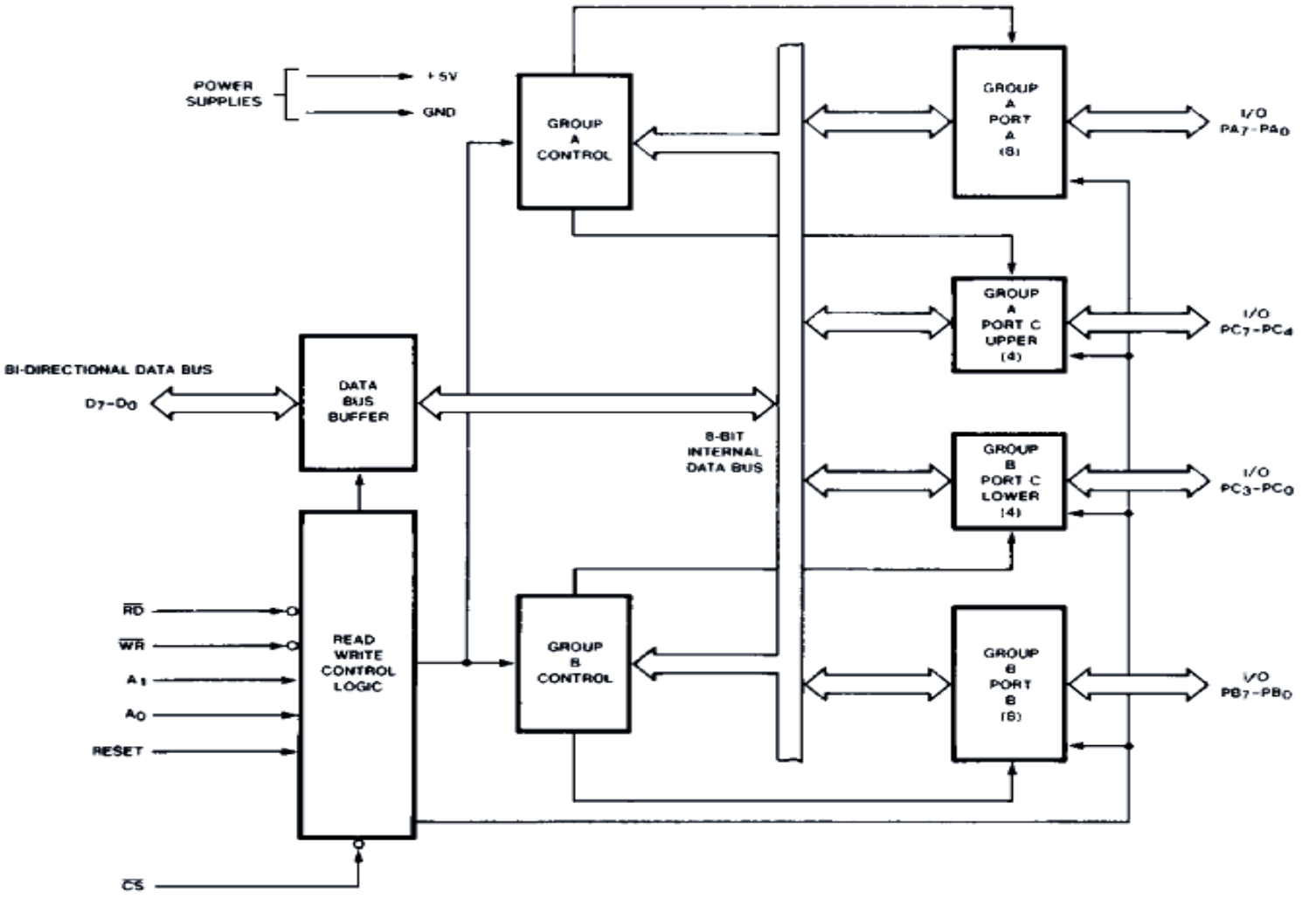


Figure 1. 82C55A Block Diagram

8255 BLOCK DIAGRAM

- Data Bus Buffer: It is an 8 bit data buffer used to interface 8255 with 8085. It is connected to D_0-D_7 bits of 8255.
- Read/write control logic:It consists of inputs $\overline{RD}$,$\overline{WR}$,A0,A1,$\overline{CS}$.
- $\overline{RD}$,$\overline{WR}$ are used for reading and writing on to 8255 and are connected to $\overline{MEMR}$,$\overline{MEMW}$ of 8085 respectively.
- A_0,A_1 are Port select signals used to select the particular port .
- $\overline{CS}$ is used to select the 8255 device .
- It is controlled by the output of the 3:8 decoder used

8255 BLOCK DIAGRAM

A_0,A_1 decide the port to be used in 8255.

A_1	A_0	Selected port
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Register

8255 BLOCK DIAGRAM

- Group A and Group B Control:
- Group A control consists of Port A and Port C upper.
- Group B control consists of Port A and Port C lower.
- Each group is controlled through software.
- They receive commands from the $\overline{RD}$, $\overline{WR}$ pins to allow access to bit pattern of 8085.
- The bit pattern consists of :

7. Information about which group is operated.
8. Information about mode of Operation.

8255 BLOCK DIAGRAM

- PORT A,B:These are bi-directional 8 bit ports each and are used to interface 8255 with CPU or peripherals.
- Port A is controlled by Group A while Port B is controlled by Group B Control.
- PORT C: This is a bi-directional 8 bit port controlled partially by Group A control and partially by Group B control .
- It is divided into two parts Port C upper and Port C lower each of a nibble.
- It is used mainly for control signals and interfacing with peripherals.

8255 MODES

- Mode 0 : Simple I/O
  - Any of A, B, CL and CH can be programmed as input or output
- Mode 1: I/O with Handshake
  - A and B can be used for I/O
  - C provides the handshake signals
- Mode 2: Bi-directional with handshake
  - A is bi-directional with C providing handshake signals
  - B is simple I/O (mode-0) or handshake I/O (mode-1)
- BSR (Bit Set Reset) Mode
  - Only C is available for bit mode access.
  - Allows single bit manipulation for control applications

INTERFACING 8085 & 8255

- Here 8255 is interfaced in Memory Mapped I/O mode.

Initially we write down the addresses and then interface it .

A15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Port
1	0	0	0	0	X	X	X	X	X	X	X	X	X	0	0	A
1	0	0	0	0	X	X	X	X	X	X	X	X	X	0	1	B
1	0	0	0	0	X	X	X	X	X	X	X	X	X	1	0	C
1	0	0	0	0	X	X	X	X	X	X	X	X	X	1	1	CW

INTERFACING 8085 & 8255

- Thus we get addresses ,considering don't cares to be zero as

Port A =8000H

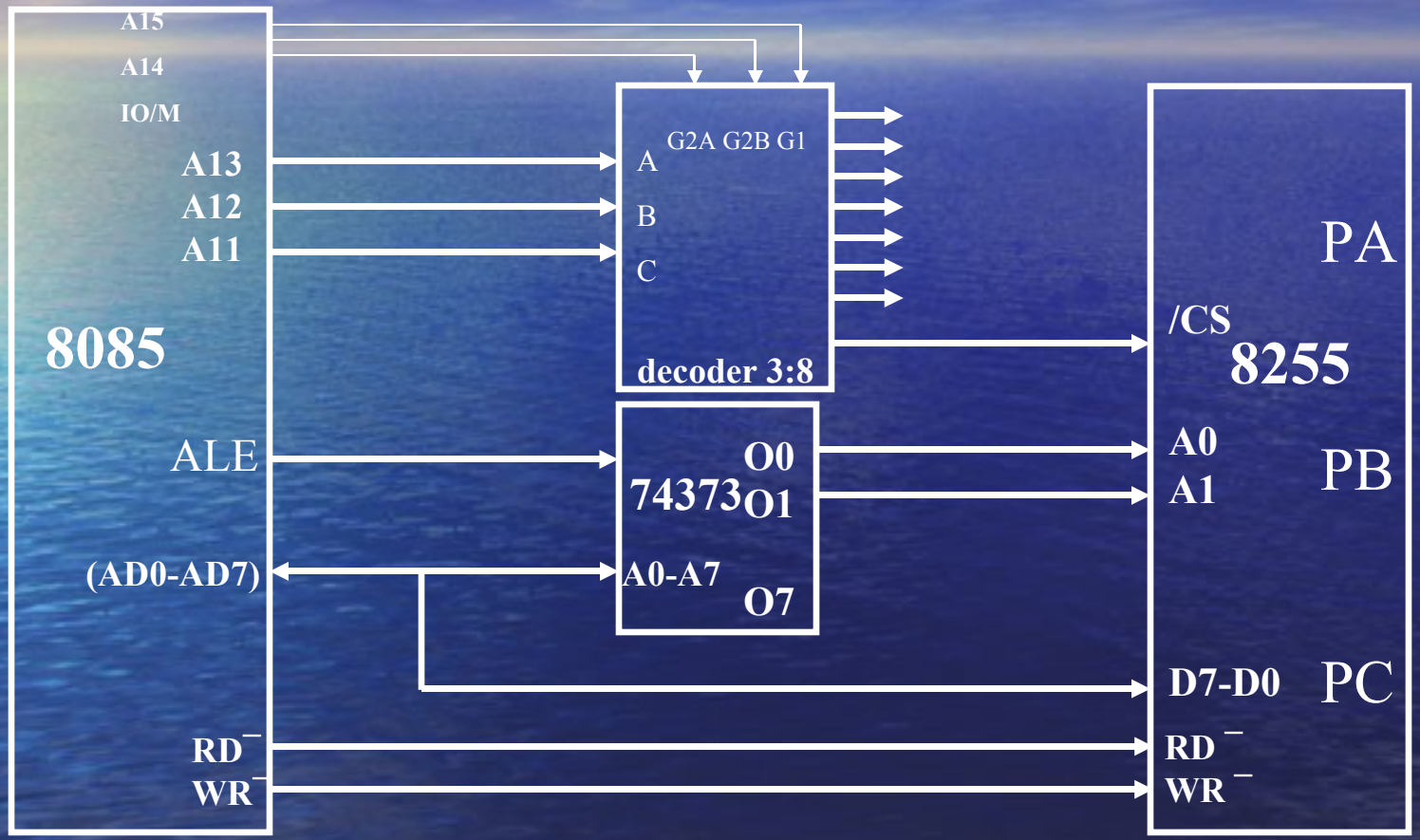
Port B =8001H

Port C =8002H

CWR =8003H

- Then,we give A_{11},A_{12},A_{13} pins to A,B,C inputs of Decoder to enable 8255 or Chip Select.
- A_{15} is logic 1 so it is given to active HIGH G_1 pin& A_{14} ,IO/$\overline{M}$ are given to active low $\overline{G2B}$,$\overline{G2A}$ pins.
- Output from Latch is given as A_0,A_1 pins to 8255 while D_0-D_7 are given as data inputs.

INTERFACING 8085 & 8255



INTERFACING 8085 & 8255

Example:Take data from 8255 port B.Add FF H .Output result to port A.

```
MVI A,82H      Initialize 8255.
OUT 83H
LDA 81H        Take  data from port B
ADI FFH        Add FF H to data
OUT 80H.       OUT Result to port A.
RST1.          STOP.
```

INTERFACING STEPPER MOTOR with 8255

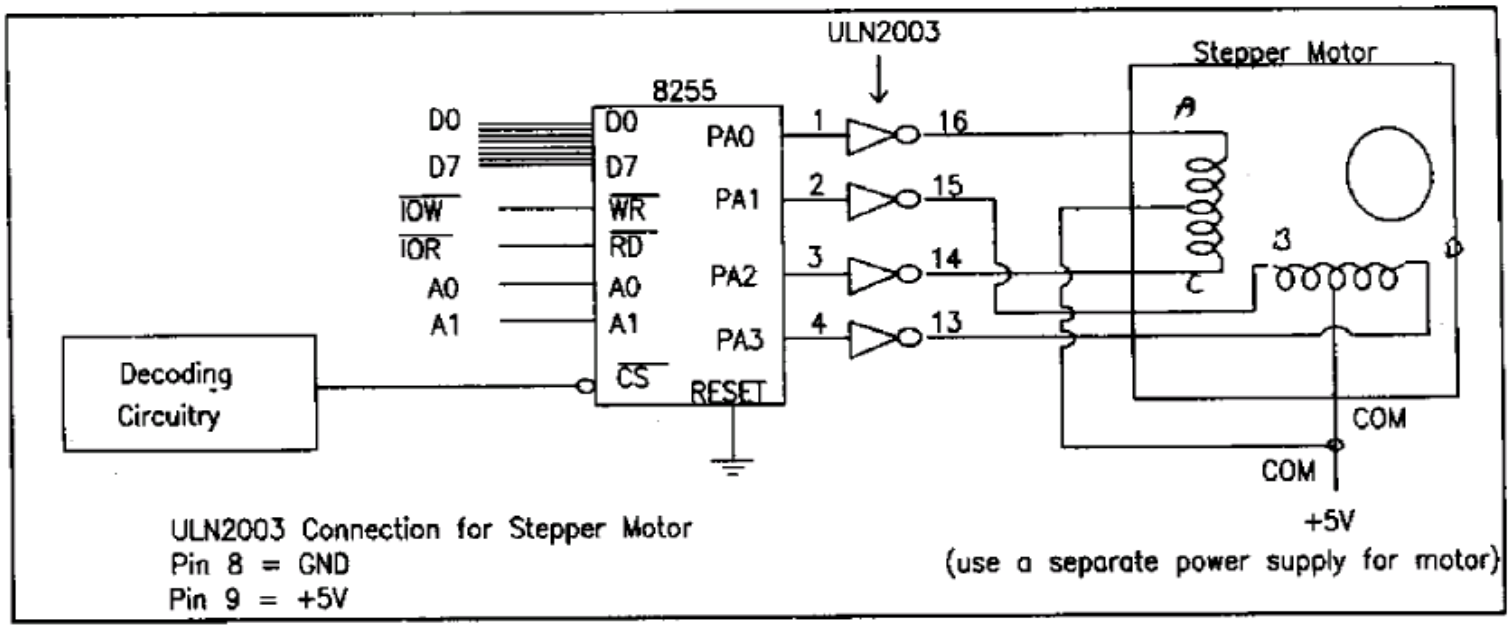


Figure 4-40. 8255 Connection to Stepper Motor

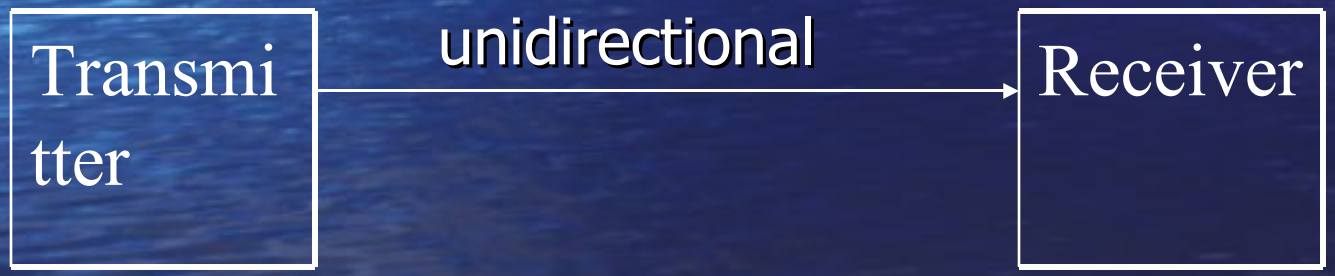
SERIAL COMMUNICATION

Serial Communications systems are of three types:

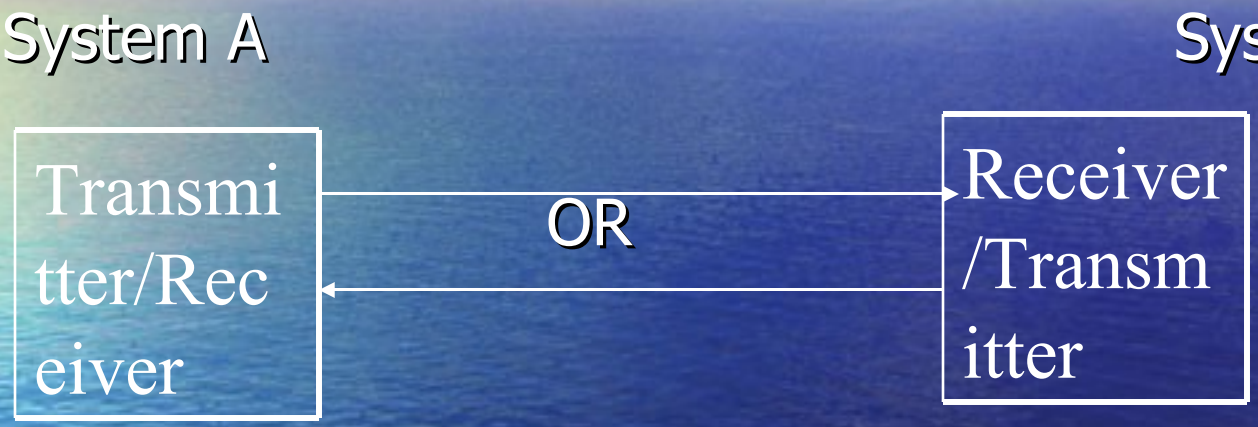
Simplex: This is a one way communication.

- Only one party can speak.
- The other party only hears to the first one but cant communicate.

System A System B



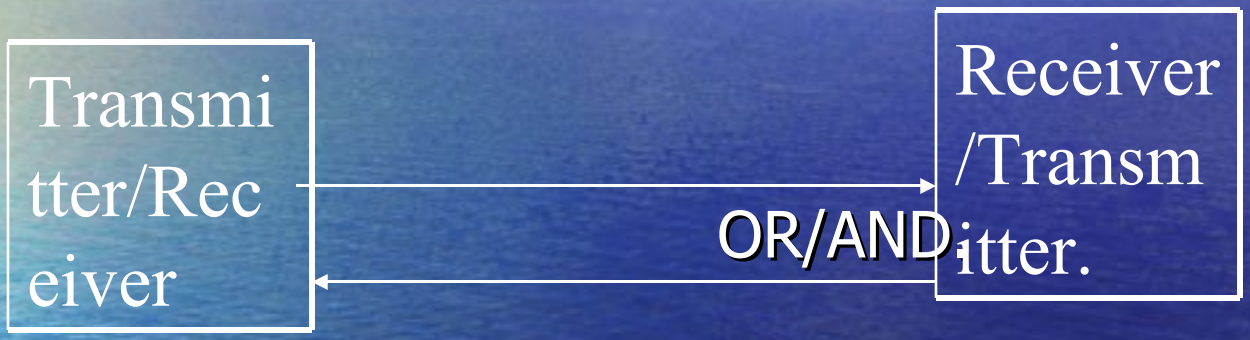
SERIAL COMMUNICATION



Half Duplex: It is a two way communication between two ports provided that only party can communicate at a time.

- When one party stops transmitting the other starts transmitting.
- The first party now acts as a receiver.

SERIAL COMMUNICATION



Full Duplex: It is a two way communication between two ports and both parties can communicate at same time.

- Thus here efficient communication can be established.

TRANSMISSION FORMATS

Asynchronous	Synchronous
1. It transfers one character at a time.	1. It transfers group of characters at a time.
2. Used for transfer data rates <20KBPS	2. Used for transfer data rates >20KBPS
3. Start and stop bit for each character which forms a frame.	3. No start and stop bit for each character.
4. Two Clocks are used for Tx and Rx	4. Single clock is used for both Tx and Rx.

INTERRUPTS IN 8085

- Interrupt is a process where an external device can get the attention of the microprocessor.

  The process starts from the I/O device

  The process is asynchronous.

- Classification of Interrupts

  Interrupts can be classified into two types:

  - Maskable Interrupts (Can be delayed or Rejected)
  - Non-Maskable Interrupts (Can not be delayed or Rejected)

INTERRUPTS IN 8085

Interrupts can also be classified into:

- Vectored (the address of the service routine is hard-wired)
- Non-vectored (the address of the service routine needs to be supplied externally by the device)

- An interrupt is considered to be an emergency signal that may be serviced.
  - The Microprocessor may respond to it as soon as possible.

INTERRUPTS IN 8085

- The 8085 has 5 interrupt inputs.
- The INTR input.

  The INTR input is the only non-vectored interrupt.

  INTR is mask-able using the EI/DI instruction pair.

  RST 5.5, RST 6.5, RST 7.5 are all automatically vectored.
  - RST 5.5, RST 6.5, and RST 7.5 are all mask-able.

  TRAP is the only non-mask-able interrupt in the

INTERRUPTS IN 8085

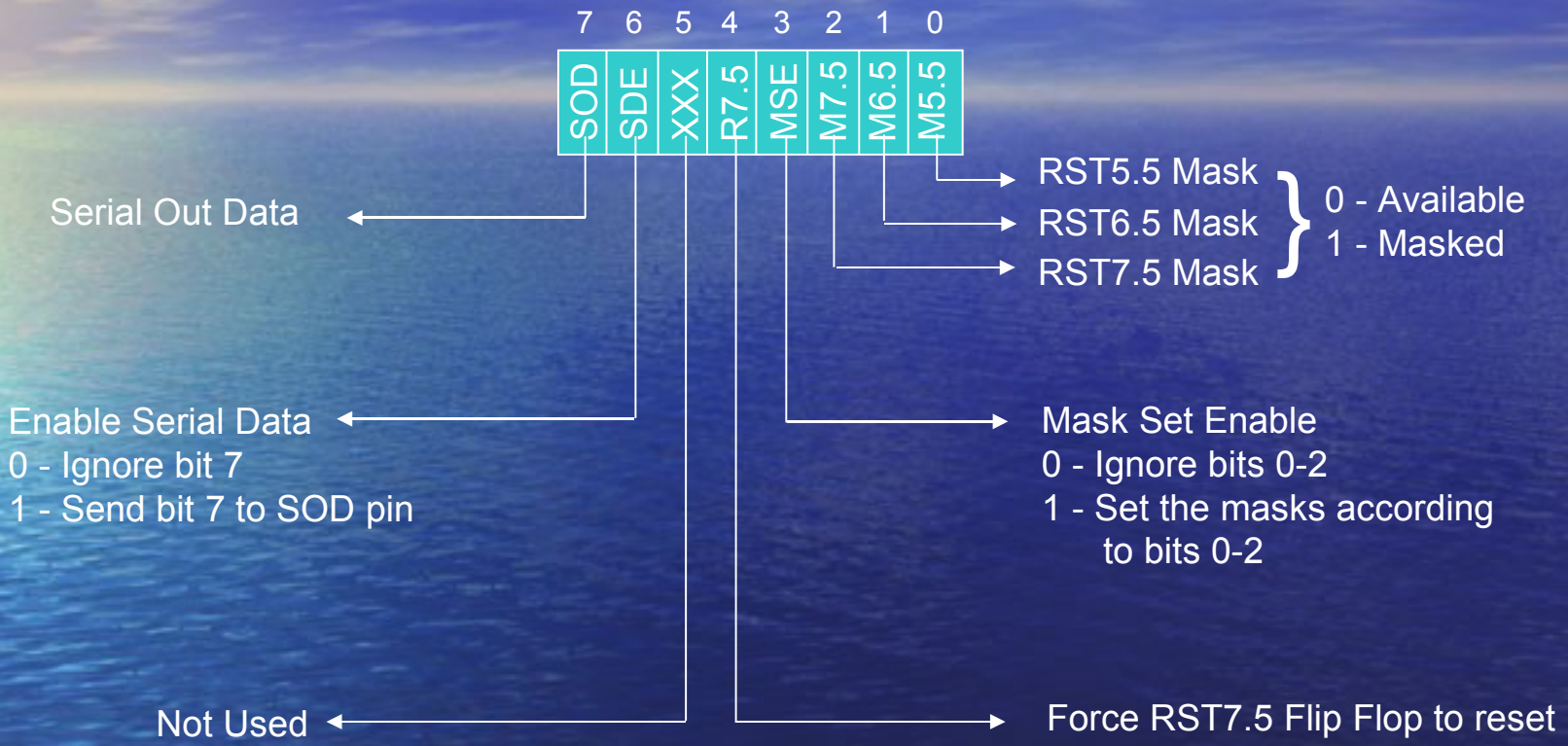
- Non vectored interrupts:
- The 8085 recognizes 8 RESTART instructions: RST0 - RST7 . Each of these would send the execution to a predetermined hard-wired memory location.

Restart Instruction	Equivalent to
RST0	CALL 0000H
RST1	CALL 0008H
RST2	CALL 0010H
RST3	CALL 0018H
RST4	CALL 0020H
RST5	CALL 0028H
RST6	CALL 0030H
RST7	CALL 0038H

INTERRUPT PRIORITY

Interrupt name	Mask-able	Vectored
TRAP	No	Yes
RST 7.5	Yes	Yes
RST 6.5	Yes	Yes
RST 5.5	Yes	Yes
INTR	YES	NO

SIM INSTRUCTION



- SIM Instruction helps activate a particular interrupt.
- It can also mask a maskable interrupt.

SIM INSTRUCTION

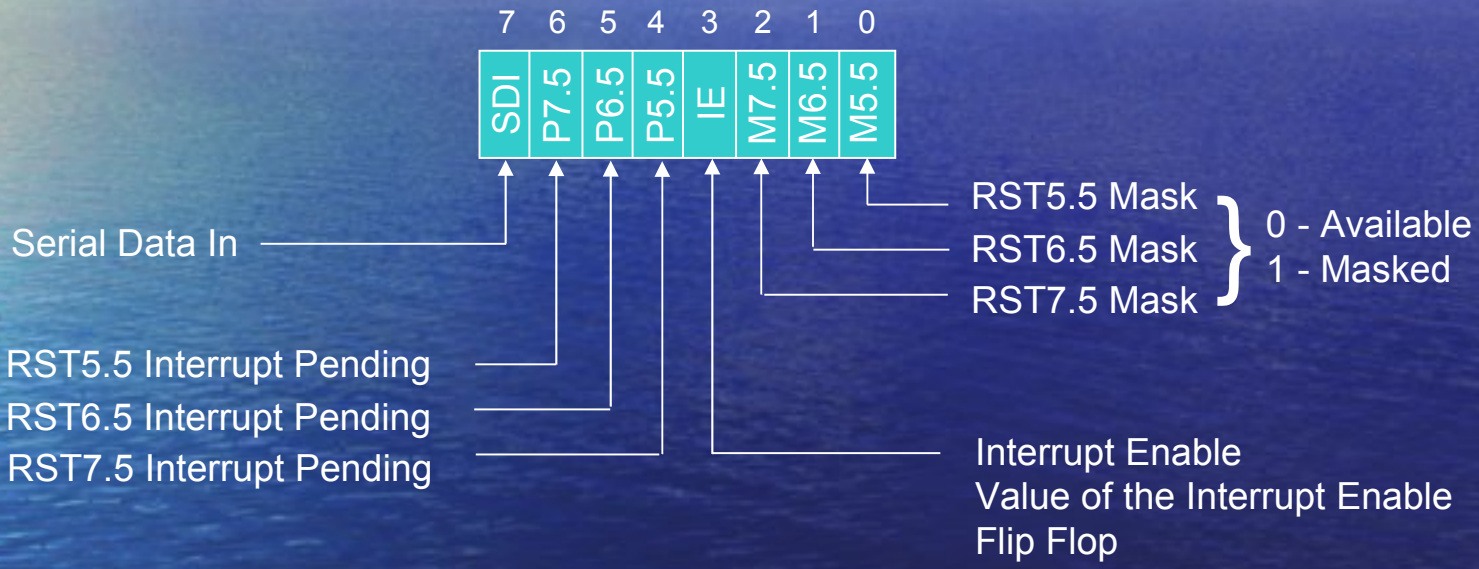
- Example: Set the interrupt masks so that RST5.5 is enabled, RST6.5 is masked, and RST7.5 is enabled.
- First, determine the contents of the accumulator.

- Enable 5.5	bit 0 = 0
- Disable 6.5	bit 1 = 1
- Enable 7.5	bit 2 = 0
- Allow setting the masks	bit 3 = 1
- Don't reset the flip flop	bit 4 = 0
- Bit 5 is not used	bit 5 = 0
- Don't use serial data	bit 6 = 0
- Serial data is ignored	bit 7 = 0

SDO	SDE	XXX	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	1	0	1	0

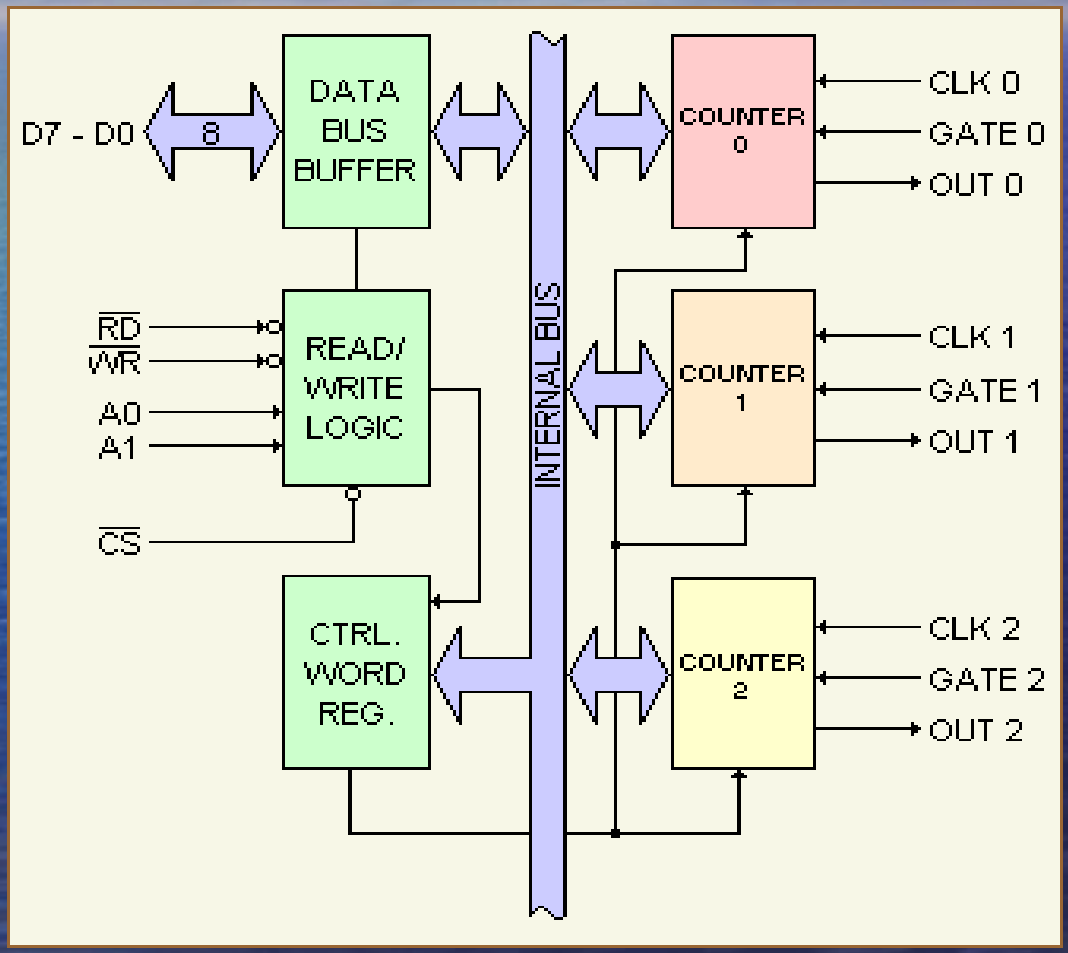
```
EI          ; Enable interrupts including INTR
MVI A, 0A   ; Prepare the mask to enable RST 7.5, and 5.5, disable 6.5
SIM         ; Apply the settings RST masks
```

RIM INSTRUCTION



- Since the 8085 has five interrupt lines, interrupts may occur during an ISR and remain pending.
- Using the RIM instruction, it is possible to can read the status of the interrupt lines and find if there are any pending interrupts.

8253 PIT


D7 - D0
8
DATA BUS BUFFER
INTERNAL BUS
COUNTER 0
CLK 0
GATE 0
OUT 0
RD
WR
A0
A1
CS
READ/ WRITE LOGIC
COUNTER 1
CLK 1
GATE 1
OUT 1
CTRL. WORD REG.
COUNTER 2
CLK 2
GATE 2
OUT 2

8253 Features

- Three independent 16 bit counters.
- 24 pin Dual in line Package.
- Counting facility in Both BCD and Binary modes.
- Dc to 2 MHz operating Frequency.
- Can be used as a clock generator.

CONTROL WORD

D7							D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

SC1	SC0	Select counter
0	0	Counter0
0	1	Counter1
1	0	Counter2
1	1	ILLEGAL

RL1	RL0	Read/Load
0	0	Counter latching
0	1	Read/load LSB
1	0	Read/load MSB
1	1	R/L MSB 1st then LSB.

CONTROL WORD

BCD =0 Binary counter

BCD =1 BCD counter

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

8253 SQUARE WAVE

- Example: Use 8253 as a square wave generator with 1ms period if the input frequency is 1MHz.
- We use counter 0 as a square wave generator and address of counter 0 =10H and control register =13H.
- I/P frequency is 1MHz.So time is 1µsec.
- Count value = Required period /Input period = 1ms/1 µsec
- =1000(Decimal).
- Thus we use 8253 as a decimal counter.

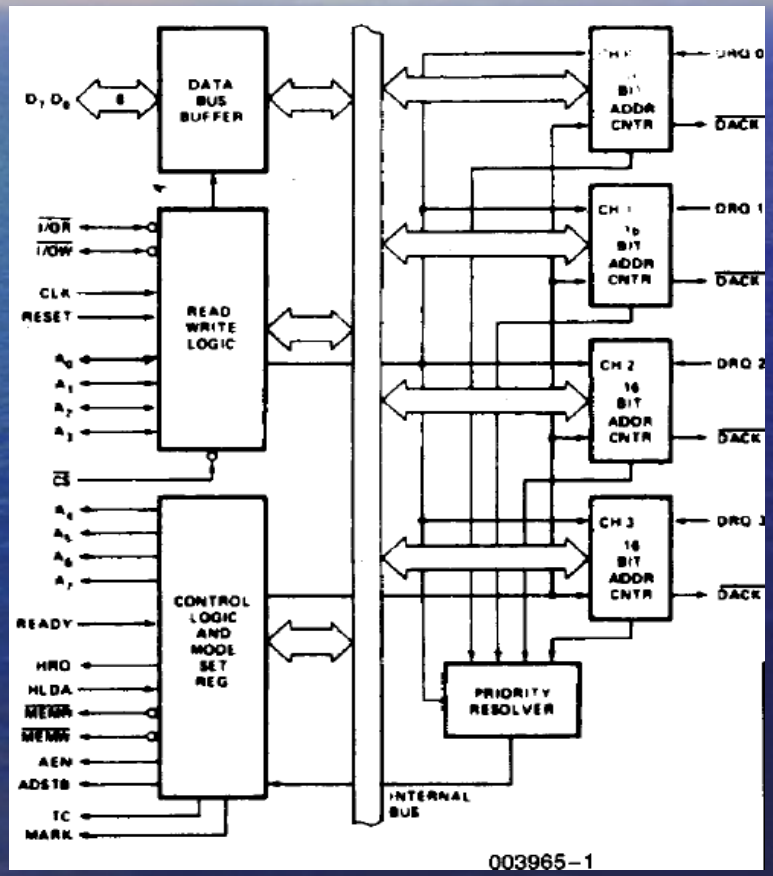
8253 SQUARE WAVE

- Program:

```
MVI A,37H          Initialize counter 0 mode 3
OUT 13H             16 bit count BCD
MVI A,00H          Load LSB count to counter 0
OUT 10H
MVI A,10H           Load MSB count to counter 0
OUT 10H.
```

- Thus, the output will be a square wave.

DMA


DATA BUS BUFFER
READ WRITE LOGIC
CONTROL LOGIC AND MODE SET REG
PRIORITY RESOLVER
INTERNAL BUS
CH 2
16 BIT ADDR CNTR
CH 3
16 BIT ADDR CNTR
CLK
RESET
READY
HRQ
HLDA
AEN
TC
MARK
003965-1

8257 DMA

- It is a 4 Channel DMA containing 4 individual I/P ,O/P Channels.

CH_0,CH_1,CH_2,CH_3

- It is compatible with Intel processors.
- The maximum frequency is 3 MHz.

It executes 3 cycles:

- DMA read
- DMA write.
- DMA verify.
- The external device can terminate DMA Operation

OPERRATING MODES OF 8257

- Rotating priority mode:Each channel has equal priority.
- Priority is shifted from one channel to other.
- Fixed priority mode: Each channel has a fixed priority

and if higher priority channels are busy then smaller priority will get to serve.

- Extended write mode: This mode is used to interface slower devices to the system.
- TC stop mode:If this bit is set the channel whose terminal count is reached is disabled.
- Auto reload mode: If this bit is set data is transferred by channel 2 only.All other channels are not used.

INSTRUCTION SET OF 8085

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code*** or ***Instruction Byte***.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

Data Transfer Instructions

Opcode	Operand	Description
MOV	Rd, Rs Rd, M M, Rs	Copy from source to destination.

- This instruction copies the contents of the source register into the destination register.
- The contents of the source register are not altered.
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.
- **Example:** MOV B, C
- MOV B, M
- MOV M, C

Data Transfer Instructions

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- The 8-bit data is stored in the destination register or memory.
- If the operand is a memory location, its location is specified by the contents of the H-L registers.
- **Example:** MVI A, 57H
- MVI M, 57H

Data Transfer Instructions

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

- This instruction loads 16-bit data in the register pair.
- **Example:** LXI H, 2034 H

Data Transfer Instructions

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- **Example:** LDA 2034H

Data Transfer Instructions

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.
- **Example:** LDAX B

Data Transfer Instructions

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.
- **Example:** LHLD 2040 H

Data Transfer Instructions

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator are copied into the memory location specified by the operand.
- **Example:** STA 2500 H

Data Transfer Instructions

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of accumulator are copied into the memory location specified by the contents of the register pair.
- **Example:** STAX B

Data Transfer Instructions

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- **Example:** SHLD 2550 H

Data Transfer Instructions

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.
- **Example:** XCHG

Arithmetic Instructions

- These instructions perform the operations like:
  - Addition
  - Subtract
  - Increment
  - Decrement

Addition

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.
- **Example:** The contents of register B cannot be added directly to the contents of register C.

Subtraction

- Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
- The result is stored in the accumulator.
- Subtraction is performed in 2’s complement form.
- If the result is negative, it is stored in 2’s complement form.
- No two other 8-bit registers can be subtracted directly.

Increment / Decrement

- The 8-bit contents of a register or a memory location can be incremented or decremented by 1.
- The 16-bit contents of a register pair can be incremented or decremented by 1.
- Increment or decrement can be performed on any register or a memory location.

Arithmetic Instructions

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

- The contents of register or memory are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADD B or ADD M

Arithmetic Instructions

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADC B or ADC M

Arithmetic Instructions

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ADI 45 H

Arithmetic Instructions

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ACI 45 H

Arithmetic Instructions

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set.
- No other flags are changed.
- **Example:** DAD B

Arithmetic Instructions

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUB B or SUB M

Arithmetic Instructions

Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBB B or SBB M

Arithmetic Instructions

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUI 45 H

Arithmetic Instructions

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBI 45 H

Arithmetic Instructions

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** INR B or INR M

Arithmetic Instructions

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place.
- **Example:** INX H

Arithmetic Instructions

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** DCR B or DCR M

Arithmetic Instructions

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- **Example:** DCX H

Logical Instructions

- These instructions perform logical operations on data stored in registers, memory and status flags.

- The logical operations are:
  - AND
  - OR
  - XOR
  - Rotate
  - Compare
  - Complement

AND, OR, XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
  - AND operation
  - OR operation
  - XOR operation

  with the contents of accumulator.
- The result is stored in accumulator.

Rotate

- Each bit in the accumulator can be shifted either left or right to the next position.

Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
  - Equality
  - Greater Than
  - Less Than

  with the contents of accumulator.
- The result is reflected in status flags.

Complement

- The contents of accumulator can be complemented.
- Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved .
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.
- **Example:** CMP B or CMP M

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The 8-bit data is compared with the contents of accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset
- **Example:** CPI 89H

Logical Instructions

Opcode	Operand	Description
ANA	R M	Logical AND register or memory with accumulator

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset and AC is set.
- **Example:** ANA B or ANA M.

Logical Instructions

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY is reset, AC is set.
- **Example:** ANI 86H.

Logical Instructions

Opcode	Operand	Description
XRA	R M	Exclusive OR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA B or XRA M.

Logical Instructions

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORA B or ORA M.

Logical Instructions

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are logically ORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORI 86H.

Logical Instructions

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA B or XRA M.

Logical Instructions

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** XRI 86H.

Logical Instructions

Opcode	Operand	Description
RLC	None	Rotate accumulator left

- Each binary bit of the accumulator is rotated left by one position.
- Bit D7 is placed in the position of D0 as well as in the Carry flag.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RLC.

Logical Instructions

Opcode	Operand	Description
RRC	None	Rotate accumulator right

- Each binary bit of the accumulator is rotated right by one position.
- Bit D0 is placed in the position of D7 as well as in the Carry flag.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RRC.

Logical Instructions

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RAL.

Logical Instructions

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RAR.

Logical Instructions

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- **Example:** CMA.

Logical Instructions

Opcode	Operand	Description
CMC	None	Complement carry

- The Carry flag is complemented.
- No other flags are affected.
- **Example:** CMC.

Logical Instructions

Opcode	Operand	Description
STC	None	Set carry

- The Carry flag is set to 1.
- No other flags are affected.
- **Example:** STC.

Branching Instructions

- The branching instruction alter the normal sequential flow.
- These instructions alter either unconditionally or conditionally.

Branching Instructions

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- **Example:** JMP 2034 H.

Branching Instructions

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- **Example:** JZ 2034 H.

Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0

Branching Instructions

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
- **Example:** CALL 2034 H.

Branching Instructions

Opcode	Operand	Description
Cx	16-bit address	Call conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
- **Example:** CZ 2034 H.

Call Conditionally

Opcode	Description	Status Flags
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P = 1
CPO	Call if Parity Odd	P = 0

Branching Instructions

Opcode	Operand	Description
RET	None	Return unconditionally

- The program sequence is transferred from the subroutine to the calling program.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example:** RET.

Branching Instructions

Opcode	Operand	Description
Rx	None	Call conditionally

- The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example:** RZ.

Return Conditionally

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1
RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0

Branching Instructions

Opcode	Operand	Description
RST	0 – 7	Restart (Software Interrupts)

- The RST instruction jumps the control to one of eight memory locations depending upon the number.
- These are used as software instructions in a program to transfer program execution to one of the eight locations.
- **Example:** RST 3.

Restart Address Table

Instructions	Restart Address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H

Control Instructions

- The control instructions control the operation of microprocessor.

Control Instructions

Opcode	Operand	Description
NOP	None	No operation

- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.
- **Example:** NOP

Control Instructions

Opcode	Operand	Description
HLT	None	Halt

- The CPU finishes executing the current instruction and halts any further execution.
- An interrupt or reset is necessary to exit from the halt state.
- **Example:** HLT

Control Instructions

Opcode	Operand	Description
DI	None	Disable interrupt

- The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.
- No flags are affected.
- **Example:** DI

Control Instructions

Opcode	Operand	Description
EI	None	Enable interrupt

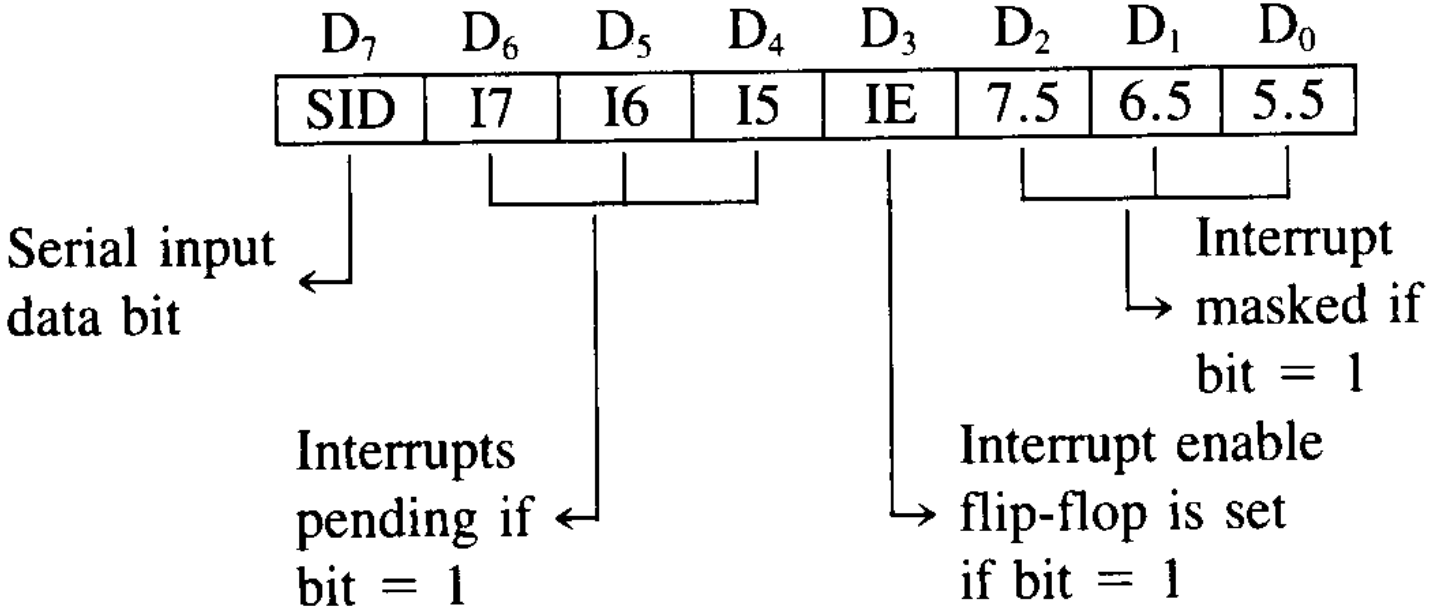
- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags are affected.
- This instruction is necessary to re-enable the interrupts (except TRAP).
- **Example:** EI

Control Instructions

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

- This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
- The instruction loads eight bits in the accumulator with the following interpretations.
- **Example:** RIM

RIM Instruction

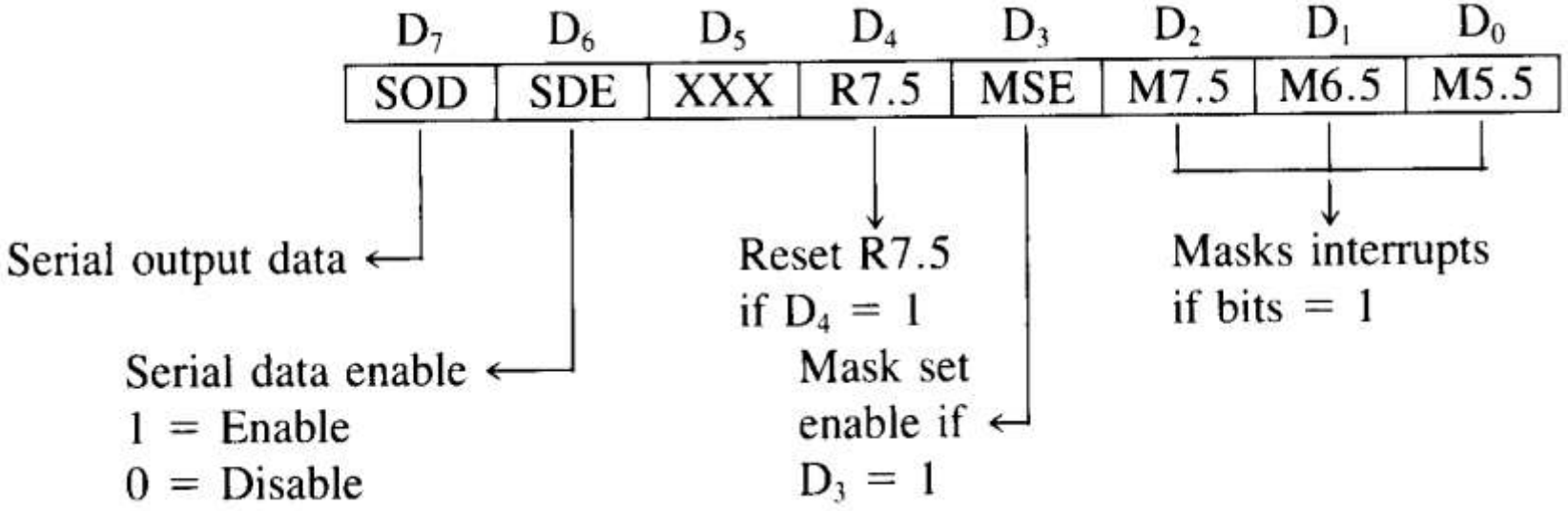


Control Instructions

Opcode	Operand	Description
SIM	None	Set Interrupt Mask

- This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
- The instruction interprets the accumulator contents as follows.
- **Example:** SIM

SIM Instruction



Addressing modes in 8085 microprocessor

(by Garima Rohela)

The way of specifying data to be operated by an instruction is called addressing mode.

Types of addressing modes –
In 8085 microprocessor there are 5 types of addressing modes:

1. **Immediate Addressing Mode –**
   In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

   Examples:
   MVI B 45 (move the data 45H immediately to register B)
   LXI H 3050 (load the H-L pair with the operand 3050H immediately)
   JMP address (jump to the operand address immediately)

2. **Register Addressing Mode –**
   In register addressing mode, the data to be operated is available inside the register(s) and register(s) is(are) operands. Therefore the operation is performed within various registers of the microprocessor.
   Examples:
   MOV A, B (move the contents of register B to register A)
   ADD B (add contents of registers A and B and store the result in register A)
   INR A (increment the contents of register A by one)

3. **Direct Addressing Mode –**
   In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself.
   Examples:
   LDA 2050 (load the contents of memory location into accumulator A)
   LHLD address (load contents of 16-bit memory location into H-L register pair)
   IN 35 (read the data from port whose address is 01)

4. **Register Indirect Addressing Mode –**
   IN register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified b a register pair.
   Examples:
   MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)

LDAX B (move contains of B-C register to the accumulator)
LXIH 9570 (load immediate the H-L pair with the address of the location 9570)

5. **Implied/Implicit Addressing Mode –**
   In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.
   Examples:
   CMA (finds and stores the 1's complement of the contains of accumultor A in A)
   RRC (rotate accumulator A right by one bit)
   RLC (rotate accumulator A left by one bit)

Notes on

Interrupts in 8085 microprocessor

(By Garima Rohela, Lecturer-GP Sonipat)

When microprocessor receives any interrupt signal from peripheral(s) which are requesting its services, it stops its current execution and program control is transferred to a sub-routine by generating **CALL** signal and after executing sub-routine by generating **RET** signal again program control is transferred to main program from where it had stopped.
When microprocessor receives interrupt signals, it sends an acknowledgement (INTA) to the peripheral which is requesting for its service.

Interrupts can be classified into various categories based on different parameters:

1. **Hardware and Software Interrupts –**

   When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as *Hardware Interrupts*. There are 5 Hardware Interrupts in 8085 microprocessor. They are – *INTR, RST 7.5, RST 6.5, RST 5.5, TRAP*

   Software Interrupts are those which are inserted in between the program which means these are mnemonics of microprocessor. There are 8 software interrupts in 8085 microprocessor. They are – *RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7*.

2. **Vectored and Non-Vectored Interrupts –**
   Vectored Interrupts are those which have fixed vector address (starting address of sub-routine) and after executing these, program control is transferred to that address.

Vector Addresses are calculated by the formula 8 * TYPE

INTERRUPT	VECTOR ADDRESS
TRAP (RST 4.5)	24 H
RST 5.5	2C H

INTERRUPT	VECTOR ADDRESS
RST 6.5	34 H
RST 7.5	3C H

For Software interrupts vector addresses are given by:

INTERRUPT	VECTOR ADDRESS
RST 0	00 H
RST 1	08 H
RST 2	10 H
RST 3	18 H
RST 4	20 H
RST 5	28 H
RST 6	30 H
RST 7	38 H

Non-Vectored Interrupts are those in which vector address is not predefined. The interrupting device gives the address of sub-routine for these interrupts. *INTR* is the only non-vectored interrupt in 8085 microprocessor.

3. **Maskable and Non-Maskable Interrupts –**

   Maskable Interrupts are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled. *INTR, RST 7.5, RST 6.5, RST 5.5* are maskable interrupts in 8085 microprocessor.

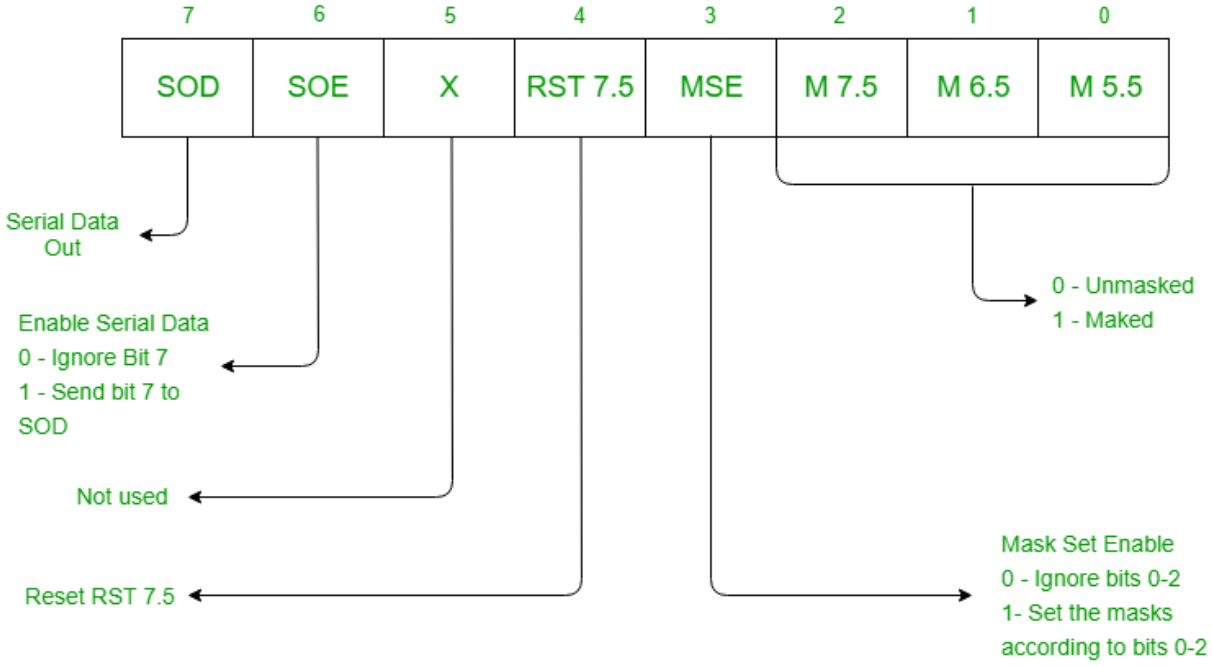
Non-Maskable Interrupts are those which cannot be disabled or ignored by microprocessor. *TRAP* is a non-maskable interrupt. It consists of both level as well as edge triggering and is used in critical power failure conditions.

Priority of Interrupts –

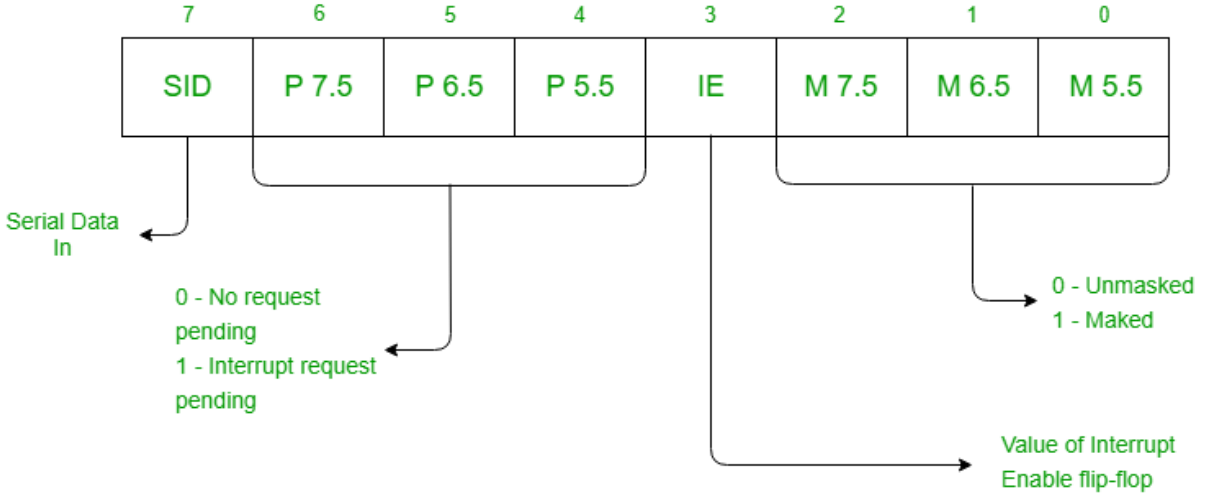
When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.

Instruction for Interrupts –

1. **Enable Interrupt (EI) –** The interrupt enable flip-flop is set and all interrupts are enabled following the execution of next instruction followed by EI. No flags are affected. After a system reset, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to enable the interrupts again (except TRAP).
2. **Disable Interrupt (DI) –** This instruction is used to reset the value of enable flip-flop hence disabling all the interrupts. No flags are affected by this instruction.
3. **Set Interrupt Mask (SIM) –** It is used to implement the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by setting various bits to form masks or generate output data via the Serial Output Data (SOD) line. First the required value is loaded in accumulator then SIM will take the bit pattern from it.



4. **Read Interrupt Mask (RIM) –** This instruction is used to read the status of the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by loading into the A register a byte which defines the condition of the mask bits for the interrupts. It also reads the condition of SID (Serial Input Data) bit on the microprocessor.



Subroutine in 8085

In computers, a subroutine is a sequence of program instructions that perform a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task have to be performed. A subroutine is often coded so that it can be started (called) several times and from several places during one execution of the

program, including from other subroutines, and then branch back (return) to the next instruction after the call, once the subroutine's task is done. It is implemented by using Call and Return instructions. The different types of subroutine instructions are

Unconditional Call instruction –
CALL address is the format for unconditional call instruction. After execution of this instruction program control is transferred to a sub-routine whose starting address is specified in the instruction. Value of PC (Program Counter) is transferred to the memory stack and value of SP (Stack Pointer) is decremented by 2.

Conditional Call instruction –
In these instructions program control is transferred to subroutine and value of PC is pushed into stack only if condition is satisfied.

INSTRUCTION	PARAMETER	COMMENT
CC	16-bit address	Call at address if cy (carry flag) = 1
CNC	16-bit address	Call at address if cy (carry flag) = 0
CZ	16-bit address	Call at address if ZF (zero flag) = 1
CNZ	16-bit address	Call at address if ZF (zero flag) = 0
CPE	16-bit address	Call at address if PF (parity flag) = 1
CPO	16-bit address	Call at address if PF (parity flag) = 0
CN	16-bit address	Call at address if SF (signed flag) = 1
CP	16-bit address	Call at address if SF (signed flag) = 0

Unconditional Return instruction –

RET is the instruction used to mark the end of sub-routine. It has no parameter. After execution of this instruction program control is transferred back to main program from where it had stopped. Value of PC (Program Counter) is retrieved from the memory stack and value of SP (Stack Pointer) is incremented by 2.

Conditional Return instruction –

By these instructions program control is transferred back to main program and value of PC is popped from stack only if condition is satisfied. There is no parameter for return instruction.

INSTRUCTION	COMMENT
RC	Return from subroutine if cy (carry flag) = 1
RNC	Return from subroutine if cy (carry flag) = 0
RZ	Return from subroutine if ZF (zero flag) = 1
RNZ	Return from subroutine if ZF (zero flag) = 0
RPE	Return from subroutine if PF (parity flag) = 1
RPO	Return from subroutine if PF (parity flag) = 0
RN	Return from subroutine if SF (signed flag) = 1
RP	Return from subroutine if SF (signed flag) = 0

Advantages of Subroutine –

1. Decomposing a complex programming task into simpler steps.
2. Reducing duplicate code within a program.
3. Enabling reuse of code across multiple programs.
4. Improving tractability or makes debugging of a program easy.

MEMORY MAPPING(BY GARIMA ROHELA)

Memory Mapping is a method to expand the memory of the microprocessor. Microprocessor have a limited amount of memory. Many-a-times it calls for more memory space. Being limited in memory resource, microprocessor needs to be connected to external memory devices like RAM/ROM/EEPROM.

8085 can access 64kB of external memory. It can be explained as- total number of address lines in 8085 are 16, therefore it can access 2^16 = 65535 locations i.e. 64kB.

2^n=number of memory locations. Where, n = number of address lines

Some of the RAM IC's are given as:

1. IC 2114 -> 1k x 4bits
2. IC 6116 -> 2k x 8bits
3. IC 6264 -> 8k x 8bits

Some of the ROM IC's are given as:

1. IC 2708 -> 1k x 8bits
2. IC 2716 -> 2k x 8bits
3. IC 2732 -> 4k x 8bits
4. IC 2764 -> 8k x 8bits
5. IC 27128 -> 16k x 8bits
6. IC 27256 -> 32k x 8bits
7. IC 2708 -> 64k x 8bits

Spacing	Memory	No. of Address lines	used address lines	Unused address lines
03FFH	1kB	10	A0-A9	A10-A15
07FFH	2Kb	11	A0-A10	A11-A15
0FFFH	4kB	12	A0-A11	A12-A15
1FFFH	8kB	13	A0-A12	A13-A15
3FFFH	16kB	14	A0-A13	A14-A15
7FFFH	32kB	15	A0-A14	A15
FFFFH	64kB	16	A0-A15	-------

Q. Design a minimum system to interface the following specification:
1. 32kB of RAM using 2 x 16kB RAM IC
2. 32kB of ROM using 2 x 16kB ROM IC

- Therefore the spacing for all the 4 ICs are same due to same size of memory ie. 3FFFH.

- The number of address lines required by each IC's is 14.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	spacing	IC details
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H	ROM IC -1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H	ROM IC- 2
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H	RAM IC- 3
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFFH	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H	RAM IC- 4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH	

MEMORY MAPPED I/O AND I/O MAPPED I/O (ISOLATED I/O)

Memory mapped I/O and Isolated I/O

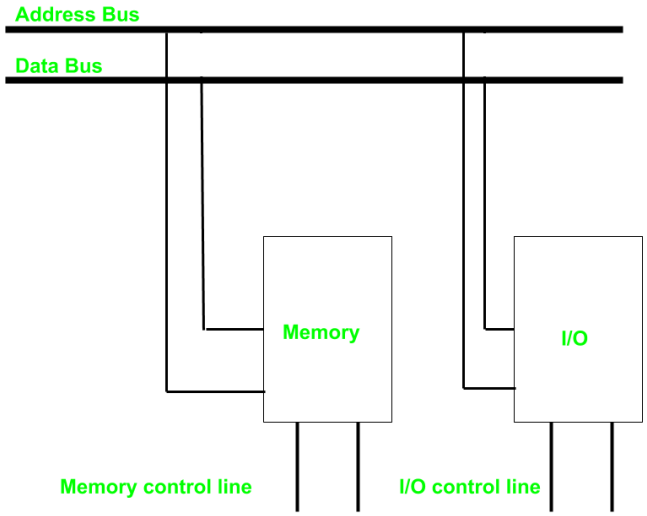
As a CPU needs to communicate with the various memory and input-output devices (I/O) as we know data between the processor and these devices flow with the help of the system bus. There are three ways in which system bus can be allotted to them :

1. Separate set of address, control and data bus to I/O and memory.

2. Have common bus (data and address) for I/O and memory but separate control lines.
3. Have common bus (data, address, and control) for I/O and memory.

In first case it is simple because both have different set of address space and instruction but require more buses.

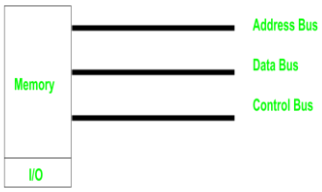
Isolated I/O –Then we have Isolated I/O in which we Have common bus(data and address) for I/O and memory but separate read and write control lines for I/O. So when CPU decode instruction then if data is for I/O then it places the address on the address line and set I/O read or write control line on due to which data transfer occurs between CPU and I/O. As the address space of memory and I/O is isolated and the name is so. The address for I/O here is called ports. Here we have different read-write instruction for



both I/O and memory.

Memory Mapped I/O –

In this case every bus in common due to which the same set of instructions work for memory and I/O. Hence we manipulate I/O same as memory and both have same address space, due to which addressing capability of memory become less because some part is occupied by the I/O.



Differences between memory mapped I/O and isolated I/O –

ISOLATED I/O	MEMORY MAPPED I/O
Memory and I/O have separate address space	Both have same address space
All address can be used by the memory	Due to addition of I/O addressable memory become less for memory
Separate instruction control read and write operation in I/O and Memory	Same instructions can control both I/O and Memory
In this I/O address are called ports.	Normal memory address are for both
More efficient due to separate buses	Lesser efficient
Larger in size due to more buses	Smaller in size
It is complex due to separate separate logic is used to control both.	Simpler logic is used as I/O is also treated as memory only.

8255 (programmable peripheral interface)PPI

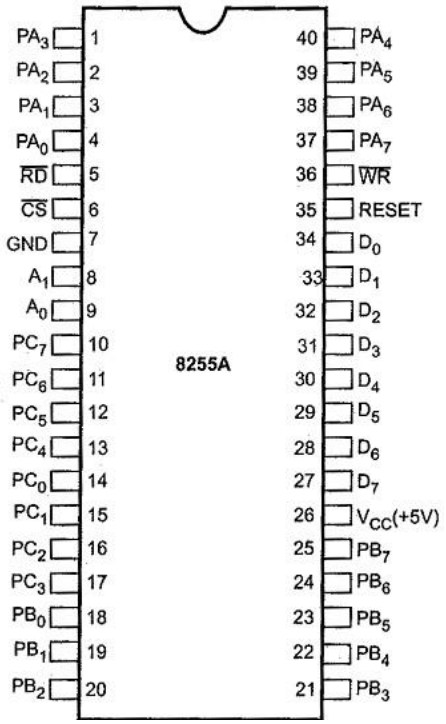
(by Garima Rohela)

8255 is a popularly used parallel, programmable input-output device. It can be used to transfer data under various condition from simple input-output to interrupt input-output. This is economical, functional, flexible but is a little complex and general purpose i/o device that can be used with almost any microprocessor.

8255 PIN DIAGRAM

It has 40 pin architecture and operates in +5v regulated power supply. It has 24 pins that can be grouped in two 8-bit parallel ports: A and B called Port A(PA) and Port B(PB) with the remaining eight known as Port C(PC). Port C can be further divided into groups of 4-bits ports named Cupper(Cu) and Clower(Cl).

8255 Pin Diagram of Microprocessor.



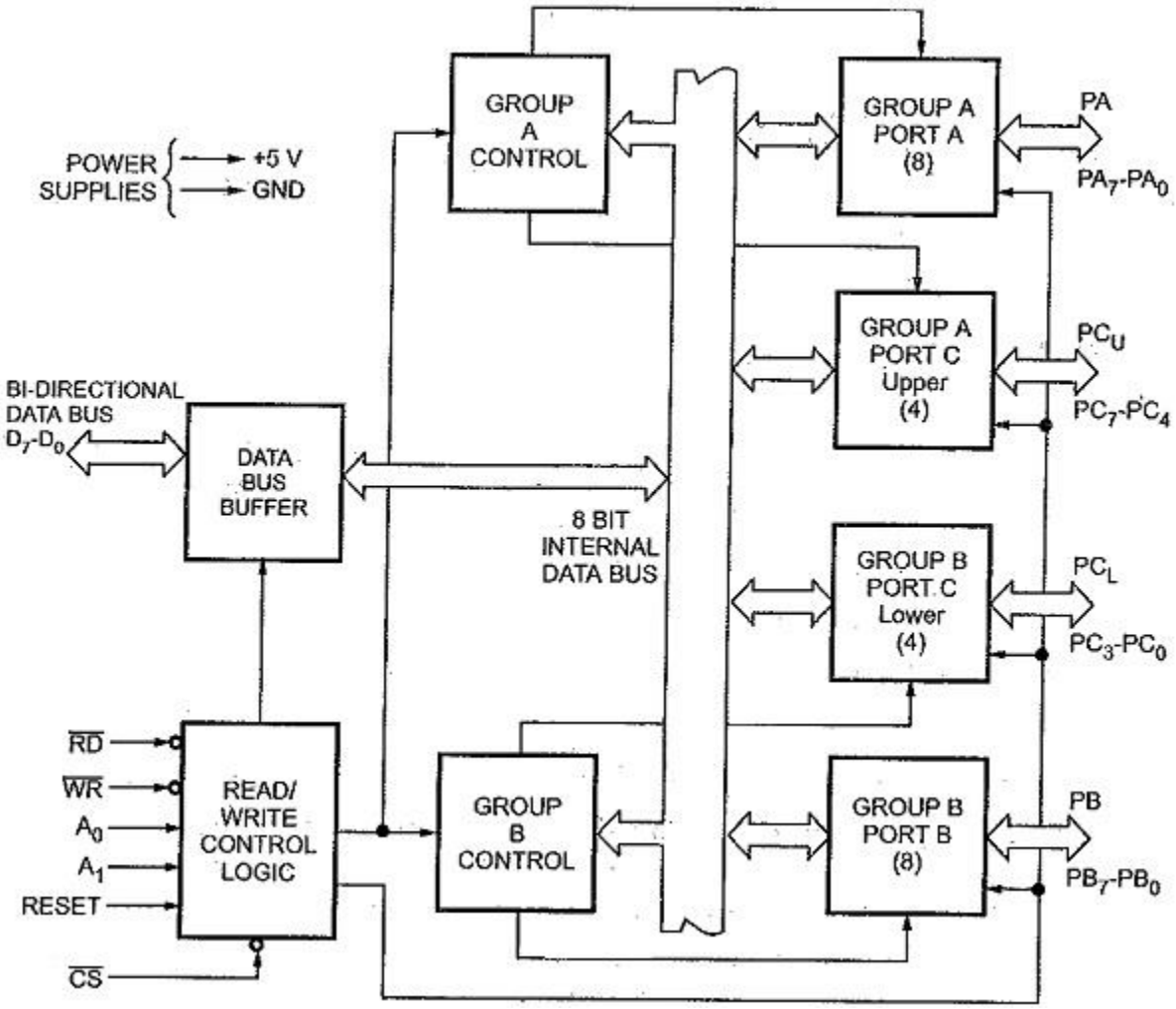
Pin Symbols	Function
D_0-D_7 (Data Bus)	These bi-directional, tri-state data bus lines are connected to the system data bus. They are used to transfer data and control word from microprocessor (8085) to 8255 or to receive data or status word from 8255 to the 8085.
PA_0-PA_7 (Port A)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data output latch/buffer, when used in output mode and an 8-bit data input buffer, when used in input mode.
PB_0-PB_7 (Port B)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data, output latch/buffer when used in output mode and an 8-bit data input buffer, when used in input mode.
D_0-D_7 (Data Bus)	These bi-directional, tri-state data bus lines are connected to the system data bus. They are used to transfer data and control word from microprocessor (8085) to 8255 or to receive data or status word from 8255 to the 8085.
PA_0-PA_7 (Port A)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data output latch/buffer, when used in output mode and an 8-bit data input buffer, when used in input mode.
PB_0-PB_7 (Port B)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data, output latch/buffer when used in output mode and an 8-bit data input buffer, when used in input mode.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operations
					Input (Read) Operation
0	0	0	1	0	Port A to Data Bus
0	1	0	1	0	Port B to Data Bus
1	0	0	1	0	Port C to Data Bus
					Output (Write) Operation
0	0	1	0	0	Data Bus to Port A
0	1	1	0	0	Data Bus to Port B
1	0	1	0	0	Data Bus to Port C
1	1	1	0	0	Data Bus to Control Register
					Disable Function
X	X	X	X	1	Data Bus Tri-stated
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus Tri-stated

Port and register select signals summer

8255 Block Diagram:

Fig. shows the internal block diagram of 8255. It consists of data bus buffer, control logic and Group A and Group B controls.



Block Diagram of 8255

Data Bus Buffer:

This tri-state bi-directional buffer is used to interface the internal data bus of 8255 Pin Diagram to the system data bus. Input or Output instructions executed by the CPU either Read data from, or Write data into the buffer. Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

Control Logic:

The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control blocks (Group A control and Group B control). It issues appropriate enabling signals to access the required data/control words or status word. The input pins for the control logic section are described here.

Group A and Group B Controls:

Each of the Group A and Group B control blocks receives control words from the CPU and issues appropriate commands to the ports associated with it. The Group A control block controls Port A and PC_7-PC_4 while the Group B control block controls Port B and PC_3-PC_0.

Port A :

This has an 8-bit latched and buffered output and an 8-bit input latch. It can be programmed in three modes: mode 0, mode 1 and mode 2.

Port B :

This has an 8-bit data I/O latch/ buffer and an 8-bit data input buffer. It can be programmed in mode 0 and mode 1.

Port C :

This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. Port C can be splitted into two parts and each can be used as control signals for ports A and B in the handshake mode. It can be programmed for bit set/reset operation.

Modes of Operation of 8255 Microprocessor:

It works in two modes:

1. Bit set reset (BSR) mode
2. Input/output (I/O) mode

Bit Set-Reset (BSR) Mode:

The individual bits of Port C can be set or reset by sending out a single OUT instruction to the control register. When Port C is used for control/status operation, this feature can be used to set or reset individual bits.

I/O Modes:

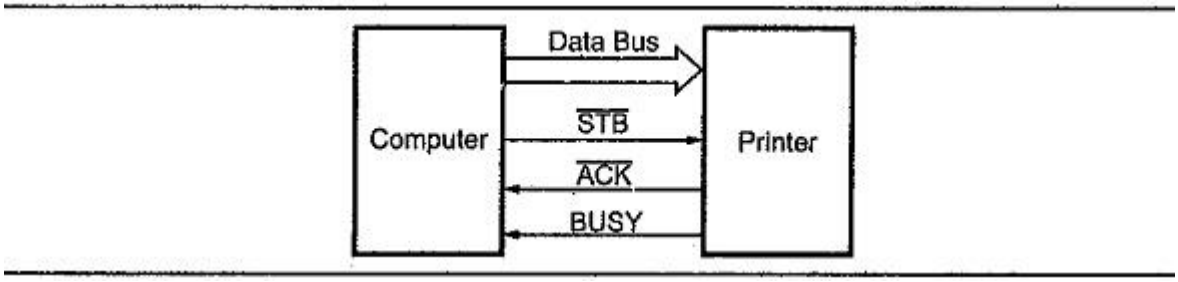
Mode 0: Simple input/output:

In this mode, ports A and B are used as two simple 8-bit I/O ports and Port C as two 4-bit ports. Each port (or half – port, in case of C) can be programmed to function as simply an input port or an output port. The input/output features in Mode 0 are as follows:

1. **Outputs are latched.**
2. **2. Inputs are buffered, not latched.**
3. **Ports do not have handshake or interrupt capability.**

Mode 1: Input/Output with handshake:

In this mode, input or output data transfer is controlled by handshaking signals. Handshaking signals are used to transfer data between devices. whose data transfer speeds are not same. For example, computer can send data .to the printer with large speed but printer can't accept data and print data with this rate. So computer has to send data with the speed with which printer can accept. This type of data transfer is achieved by using handshaking signals along-with data signals. Fig. shows data transfer between computer and printer using handshaking signals.



Data transfer between computer and printer using handshake signals

These handshaking signals are used to tell computer whether printer is ready to accept the data or not. If printer is ready to accept the data then after sending data on data bus, computer uses another handshaking signal (STB) to tell printer that valid data is available on the data bus.

The 8255 Pin Diagram mode 1 which supports handshaking has following features.

1. **Two ports (A and B) function as 8-bit I/O ports. They, can be configured either as input or output ports.**
2. **Each port uses three lines from Port C as handshake signals. The remaining two lines of. Port C can be used for simple I/O functions.**
3. **Input and output data are latched.**
4. **Interrupt logic is supported.**
5. **Mode 2 : Bi-directional I/O data transfer:**

This mode allows bi-directional data transfer (transmission and reception) over a single 8-bit data bus using handshaking signals. This feature is available only in Group A with Port A as the 8-bit bidirectional data bus; and PC_3 – PC_7 are used for handshaking purpose. In this mode, both inputs and outputs are latched. Due to use of a single 8-bit data bus for bi-directional data transfer, the data sent out by the CPU through Port A appears on the bus connecting it to the peripheral, only when the peripheral requests it. The remaining lines of Port C i.e. PC_0-PC_2 can be used for simple I/O functions. The Port B can be programmed in mode 0 or in mode 1. When Port B is programmed in mode 1, PC_0-PC_2 lines of Port C are used as handshaking signals.

Control Word Formats:

A high on the RESET pin causes all 24 lines of the three 8-bit ports to be in the input mode. All flip-flops are cleared and the interrupts are reset. This condition is maintained even after the RESET goes low. The ports of the 8255 Pin Diagram can then be programmed

for any other mode by writing a single control word into the control register, when required.

For Bit Set/Reset Mode:

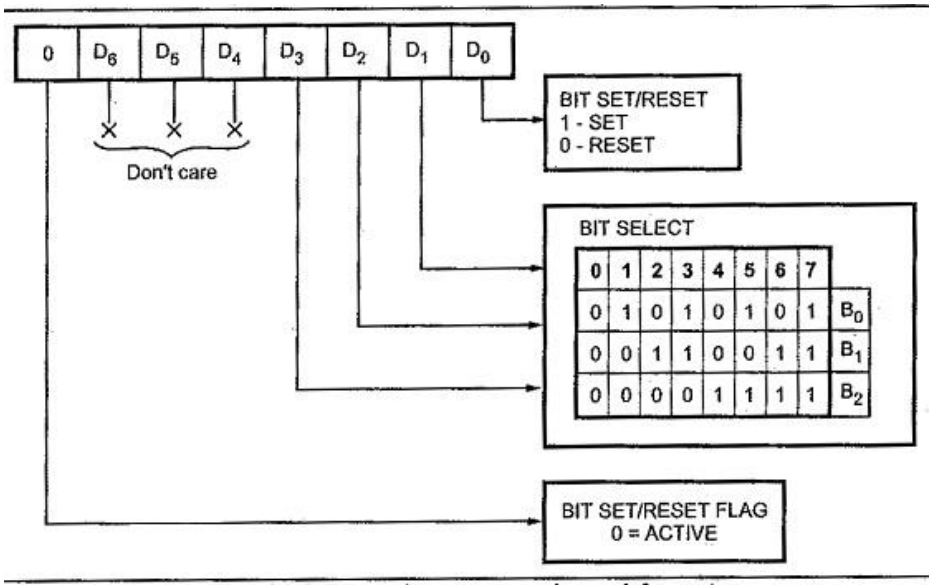


Fig. 14.4 Bit set/reset control word format

The eight possible combinations of the states of bits D_3 – D_1 (B_2 B_1 B_0) in the Bit Set-Reset format (BSR) determine particular bit in PC_0 – PC_7 being set or reset as per the status of bit D_0. A BSR word is to be written for each bit that is to be set or reset. For example, if bit PC_3 is to be set and bit PC_4 is to be reset, the appropriate BSR words that will have to be loaded into the control register will be, 0XXX0111 and 0XXX1000, respectively, where x is don't care.

The BSR word can also be used for enabling or disabling interrupt signals generated by Port C when the 8255 Pin Diagram is programmed for Mode 1 or 2 operation. This is done by setting or resetting the associated bits of the interrupts. This is described in detail in next section.

For I/O Mode:

The control words for both, mode definition and Bit Set – Reset are loaded into the same control register, with bit D_7 used for specifying whether the word loaded into the control register is

a mode definition word or Bit Set-Reset word. If D_7 is high, the word is taken as a mode definition word, and if it is low, it is taken as a Bit Set-Reset word. The appropriate bits are set or reset depending on the type of operation desired, and loaded into the control register.

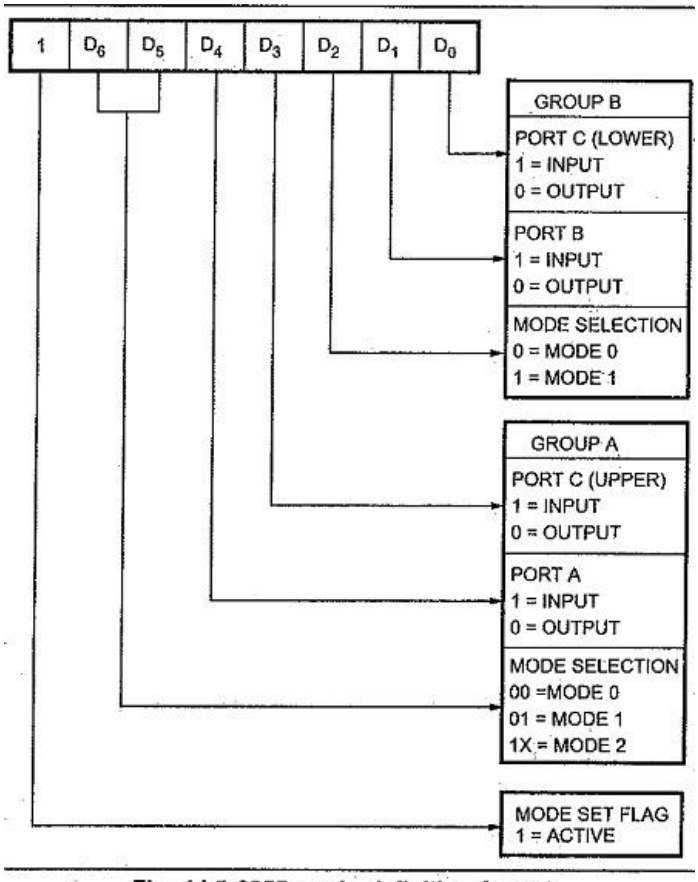


Fig. 14.5 8255 mode definition format

DMA CONTROLLER 8237/8257

(by Garima Rohela)

Introduction of 8237

Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU).

- It is also a fast way of transferring data within (and sometimes between) computer.
- The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled.
- The DMA controller temporarily borrows the address bus, data bus and control bus from the microprocessor and transfers the data directly from the external devices to a series of memory locations (and vice versa).

Basic DMA Operation:

- Two control signals are used to request and acknowledge a direct memory access (DMA) transfer in the microprocessor-based system.
    1. The HOLD signal as an input (to the processor) is used to request a DMA action.
    2. The HLDA signal as an output that acknowledges the DMA action.
- When the processor recognizes the hold, it stops its execution and enters hold cycles.

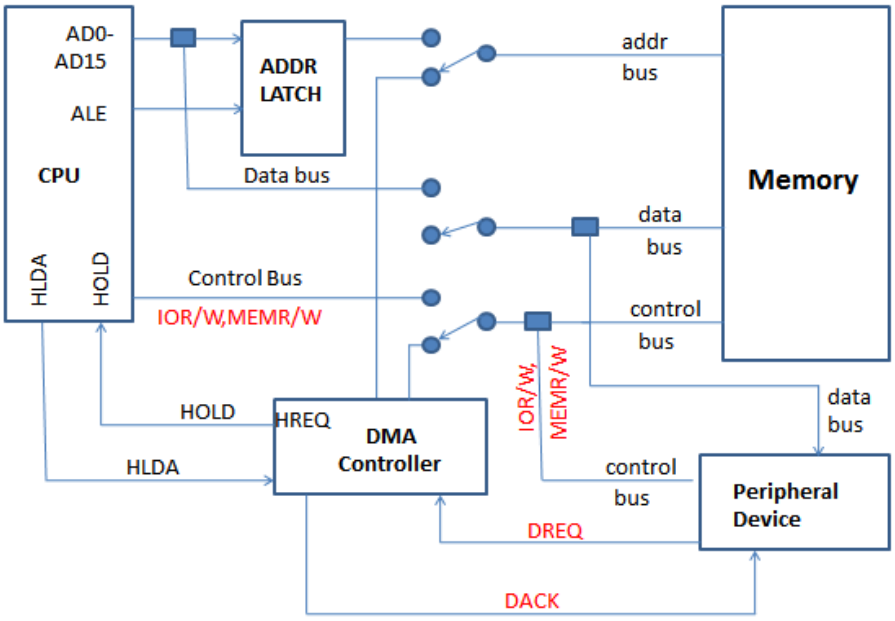
- HOLD input has higher priority than INTR or NMI.
- The only microprocessor pin that has a higher priority than a HOLD is the RESET pin.
- HLDA becomes active to indicate that the processor has placed its buses at high-impedance state.

Basic DMA Definitions

- Direct memory accesses normally occur between an I/O device and memory without the use of the microprocessor.
    1. A DMA read transfers data from the memory to the I/O device.
    2. A DMA write transfers data from an I/O device to memory.
- The system contains separate memory and I/O control signals.
- Hence the Memory & the I/O are controlled simultaneously
- The DMA controller provides memory with its address, and the controller signal selects the I/O device during the transfer.
- Data transfer speed is determined by speed of the memory device or a DMA controller.
- In many cases, the DMA controller slows the speed of the system when transfers occur.
- The serial PCI (Peripheral Component Interface) Express bus transfers data at rates exceeding DMA transfers.
- This in modern systems has made DMA is less important.

CPU having the control over the bus

When DMA operates



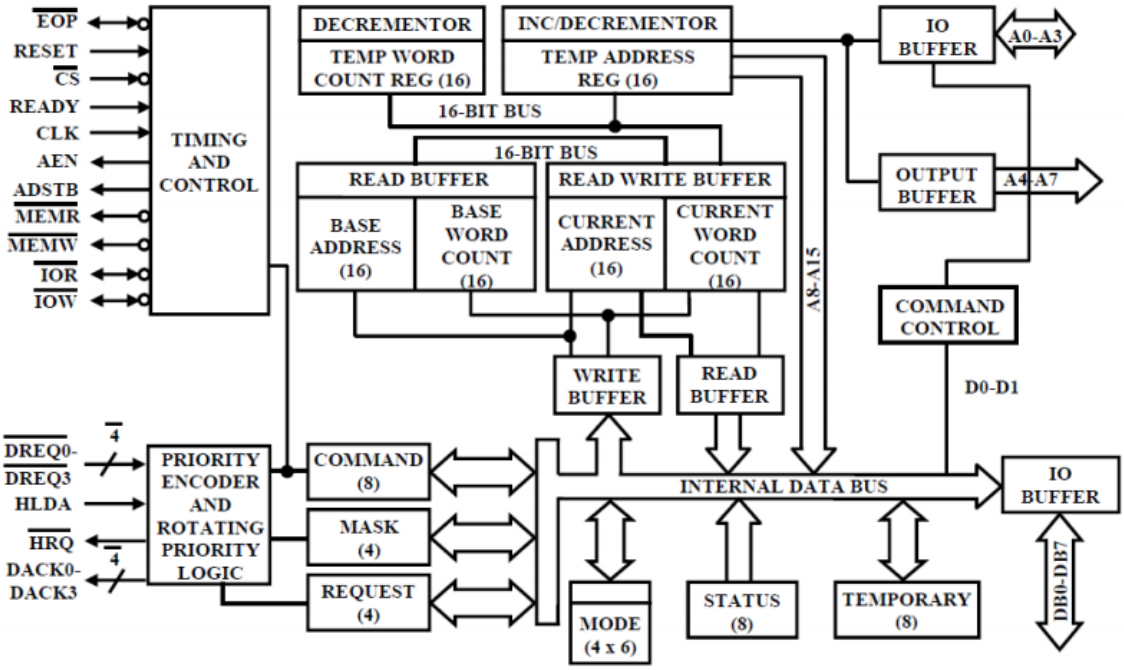
The 8237 DMA Controller

- The 8237 supplies memory & I/O with control signals and memory address information during the DMA transfer.
- It is actually a special-purpose microprocessor whose job is high-speed data transfer between memory and I/O

- 8237 is not a discrete component in modern microprocessor-based systems.
- It appears within many system controller chip sets
- 8237 is a four-channel device compatible with 8086/8088, adequate for small systems.
- Expandable to any number of DMA channel inputs
- 8237 is capable of DMA transfers at rates up to 1.6MB per second.

- Each channel is capable of addressing a full 64K-byte section of memory.

Block Diagram of 8237



8237 Internal Registers

CAR

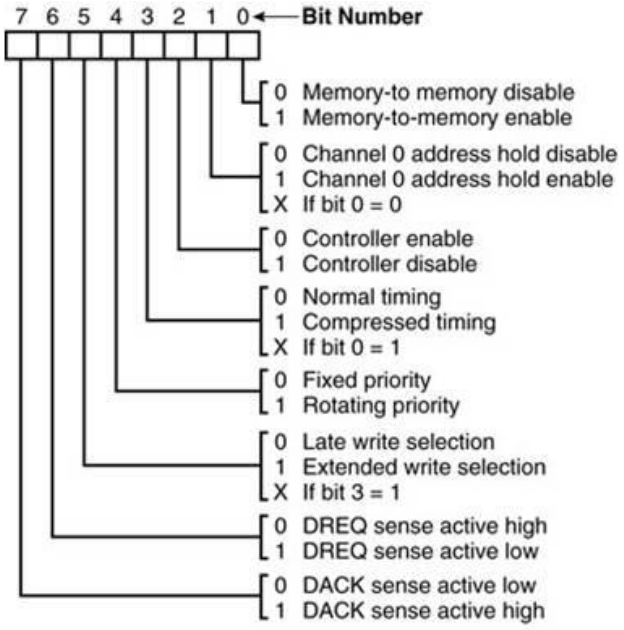
- The `current address register` holds a 16-bit memory address used for the DMA transfer.
- each channel has its own current address register for this purpose.
- When a byte of data is transferred during a DMA operation, CAR is either incremented or decremented. depending on how it is programmed

CWCR

- The current word count register programs a channel for the number of bytes to transferred during a DMA action.

CR

- The `command register` programs the operation of the 8237 DMA controller.
- The register uses bit position 0 to select the memory-to-memory DMA transfer mode.
    1. Memory-to-memory DMA transfers use DMA channel
    2. DMA channel 0 to hold the source address
    3. DMA channel 1 holds the destination address

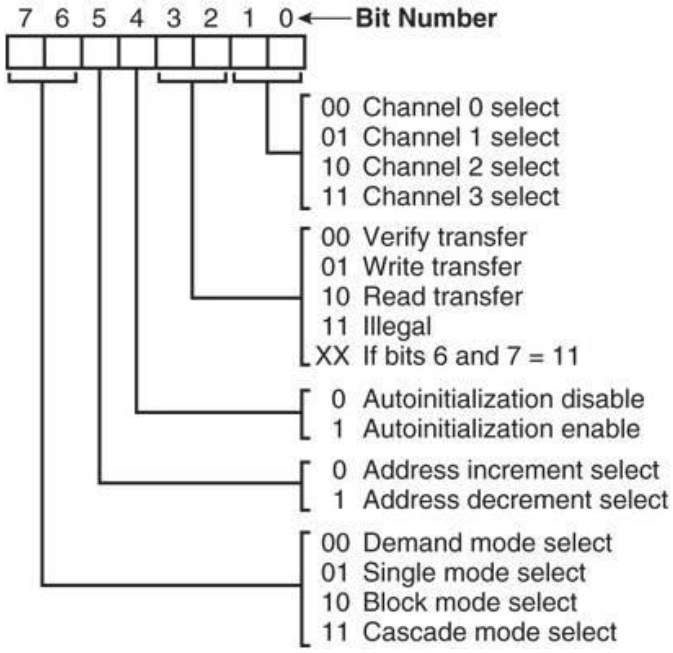


BA and BWC

- `The base address` (BA) and `base word count` (BWC) registers are used when auto-initialization is selected for a channel.
- In auto-initialization mode, these registers are used to reload the CAR and CWCR after the DMA action is completed.

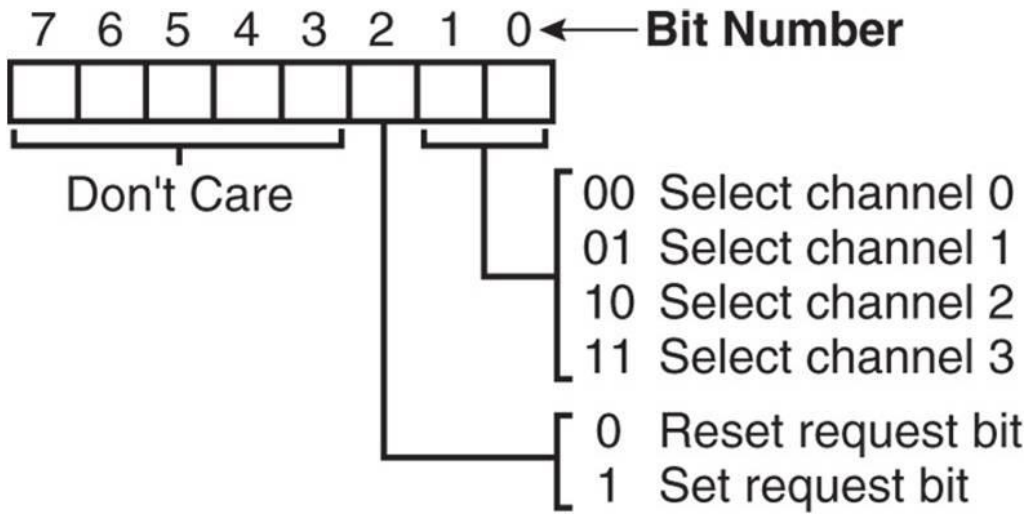
MR

- The `mode register` programs the mode of operation for a channel.
- Each channel has its own mode register as selected by bit positions 1 and 0.
    1. Remaining bits of the mode register select operation, auto-initialization, increment/decrement, and mode for the channel



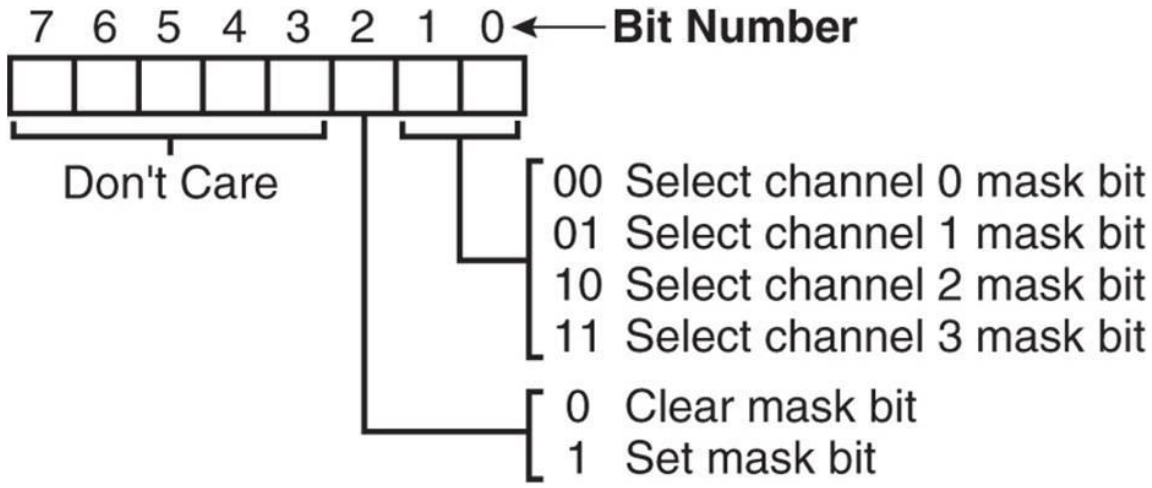
BR

- The `bus request register` is used to request a DMA transfer via software.
    1. very useful in memory-to-memory transfers, where an external signal is not available to begin the DMA transfer



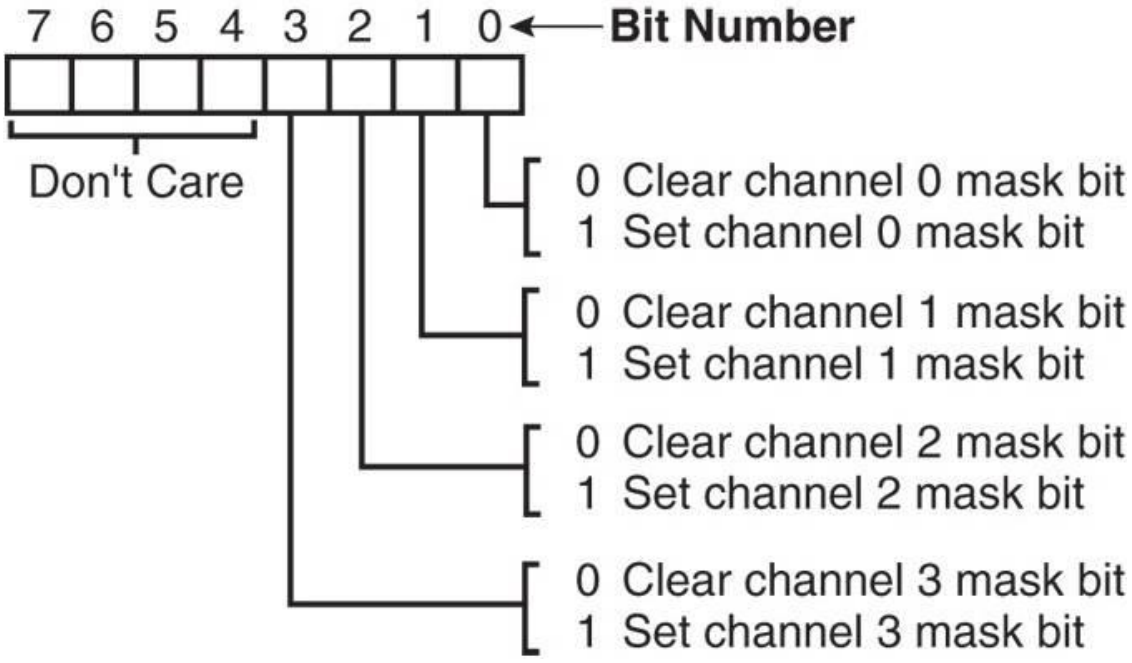
MRSR

- The `mask register set/reset` sets or clears the channel mask.
    1. if the mask is set, the channel is disabled
    2. the RESET signal sets all channel masks to disable them



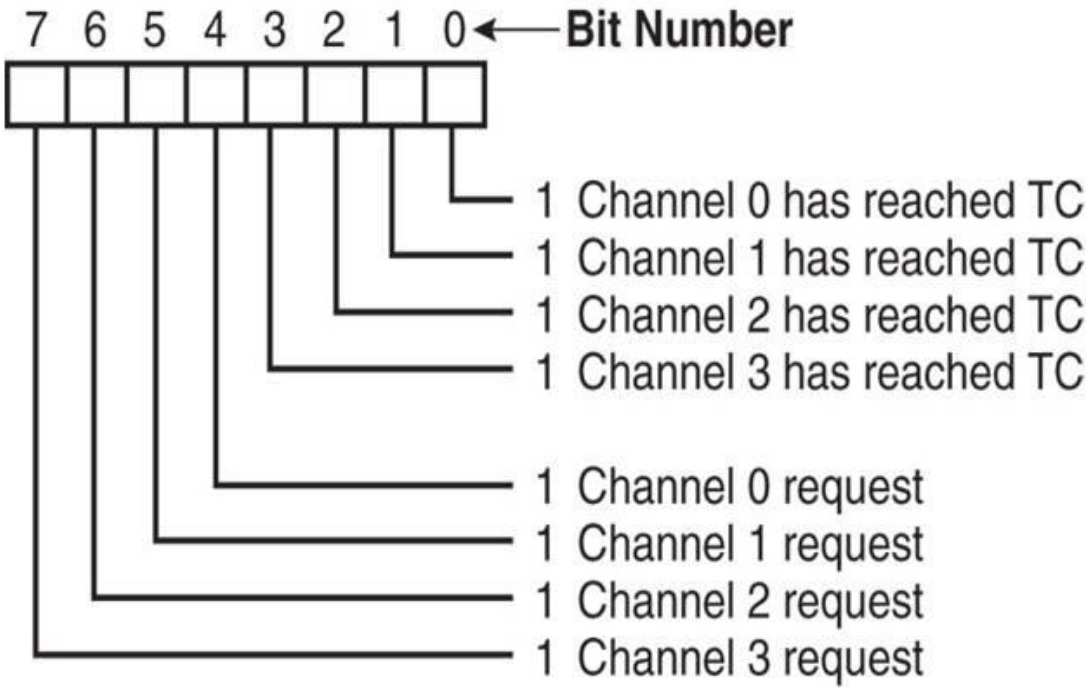
MSR

- The `mask register` clears or sets all of the masks with one command instead of individual channels, as with the MRSR.



SR

- The `status register` shows status of each DMA channel. The TC bits indicate if the channel has reached its terminal count (transferred all its bytes).
- When the terminal count is reached, the DMA transfer is terminated for most modes
  of operation.
- The request bits indicate whether the DREQ input for a given channel is active.



8237 Software Commands

Master clear

Acts exactly the same as the RESET signal to the 8237. As with the RESET signal, this command disables all channels

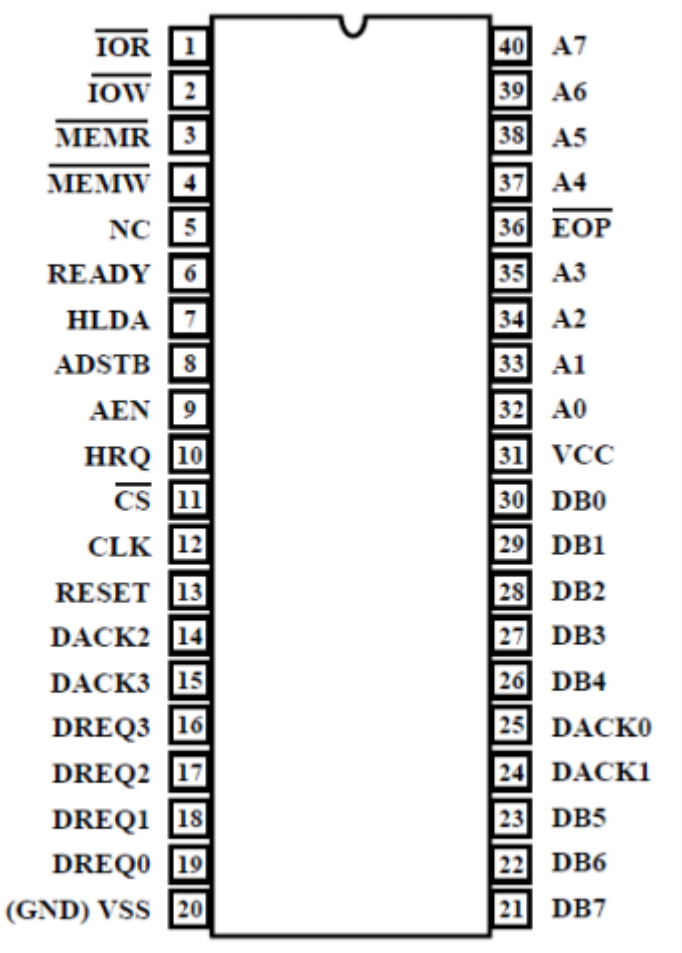
Clear mask register

Enables all four DMA channels.

Clear the first/last flip-flop

Clears the first/last (F/L) flip-flop within 8237. The F/L flip-flop selects which byte (low or high order) is read/written in the current address and current count registers. if F/L = 0, the low-order byte is selected if F/L = 1, the high-order byte is selected Any read or write to the address or count register automatically toggles the F/L flip-flop.

Pin Diagram and Pin description of 8237



VCC: POWER: a5V supply

VSS: GROUND: Ground.

CLK Input: CLOCK INPUT : Clock Input controls the internal operations of the 8237A and its rate of data transfers. The input may be driven at up to 5 MHz for the 8237A-5.

CS Input:

CHIP SELECT: Chip Select is an active low input used to select the 8237A as an I/O device during the Idle cycle. This allows CPU communication on the data bus.

RESET Input:

RESET: Reset is an active high input which clears the Command, Status, Request and Temporary registers. It also clears the first/ last flip/flop and sets the Mask register. Following a Reset the device is in the Idle cycle.

READY Input:

READY: Ready is an input used to extend the memory read and write pulses from the 8237A to accommodate slow memories or I/O peripheral devices. Ready must not make transitions during its specified setup/hold time.

HLDA Input:

HOLD ACKNOWLEDGE: The active high Hold Acknowledge from the CPU indicates that it has relinquished control of the system busses.

DREQ0 ±DREQ3 Input:

DMA REQUEST: The DMA Request lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In fixed Priority, DREQ0 has the highest priority and DREQ3 has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of DREQ signal. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active.

DB0 ±DB7:

DATA BUS: The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program condition during the I/O Read to output the contents of an Address register, a Status register, the Temporary register or a Word Count register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 8237A control registers. During DMA cycles the most significant 8 bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations, data from the memory comes into the 8237A on the data bus during the read-from-memory transfer. In the

write-to-memory transfer, the data bus outputs place the data into the new memory location.

IOR Input/Output:

I/O READ: I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 8237A to access data from a peripheral during a DMA Write transfer.

IOW Input/Output:

I/O WRITE: I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 8237A. In the Active cycle, it is an output control signal used by the 8237A to load data to the peripheral during a DMA Read transfer.

EOP Input/Output:

END OF PROCESS: End of Process is an active low bidirectional signal. Information concerning the completion of DMA services is available at the bidirectional EOP pin. The 8237A allows an external signal to terminate an active DMA service. This is accomplished by pulling the EOP input low with an external EOP signal. The 8237A also generates a pulse when the terminal count (TC) for any channel is reached. This generates an EOP signal

which is output through the EOP line. The reception of EOP, either internal or external, will cause the 8237A to terminate the service, reset the request, and, if Auto initialize is enabled, to write the base registers to the current registers of that channel. The mask bit and TC bit in the status word will be set for the currently active channel by EOP unless the channel is programmed for Auto initialize. In that case, the mask bit remains unchanged. During memory-to-memory transfers, EOP will be output when the TC for channel 1 occurs. EOP should be tied high with a pull-up resistor if it is not used to prevent erroneous end of process inputs.

A0 ±A3 Input/Output:

ADDRESS: The four least significant address lines are bidirectional three-state signals. In the Idle cycle they are inputs and are used by the CPU to address the register to be loaded or read. In the Active cycle they are outputs and provide the lower 4 bits of the output address.

A4 ±A7 Output:

ADDRESS: The four most significant address lines are three-state outputs and provide 4 bits of address. These lines are enabled only during the DMA service.

HRQ Output:

HOLD REQUEST: This is the Hold Request to the CPU and is used

to request control of the system bus. If the corresponding mask bit is clear, the presence of any valid DREQ causes 8237A to issue the HRQ.

DACK0 ±DACK3 Output:

DMA ACKNOWLEDGE: DMA Acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The sense of these lines is programmable. Reset initializes them to active low.

AEN Output:

ADDRESS ENABLE: Address Enable enables the 8-bit latch containing the upper 8 address bits onto the system address bus. AEN can also be used to disable other system bus drivers during DMA transfers. AEN is active HIGH.

ADSTB Output:

ADDRESS STROBE: The active high, Address Strobe is used to strobe the upper address byte into an external latch.

MEMR Output

MEMORY READ: The Memory Read signal is an active low three-state output used to access data from the selected memory

location during a DMA Read or a memory-to-memory transfer.

MEMW Output:

MEMORY WRITE: The Memory Write is an active low three-state output used to write data to the selected memory location during a DMA Write or a memory-to-memory transfer.

PIN5 Input:

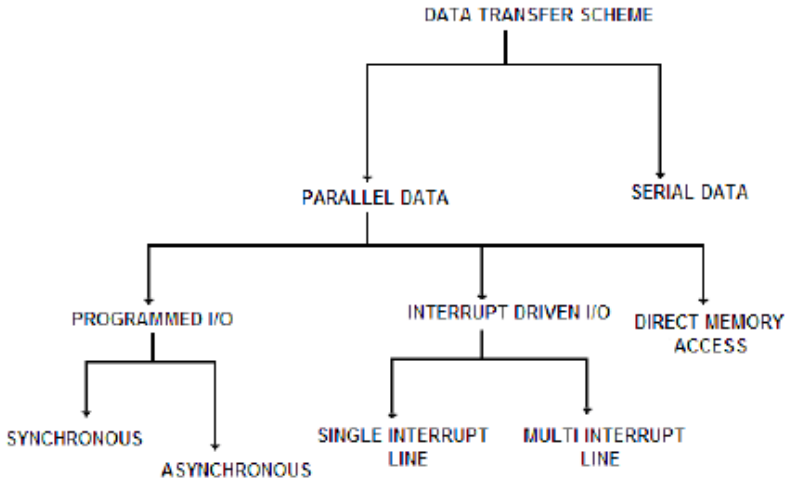
PIN5: This pin should always be at a logic HIGH level. An internal pull-up resistor will establish a logic high when the pin is left floating. It is recommended however, that PIN5 be connected to VCC

DATA TRANSFER TECHNIQUES

(by Garima Rohela)

Data transfer schemes of 8085 microprocessor

In 8085 microprocessor based systems several input and output devices are connected. We know that data transfer may take place between microprocessor and memory, microprocessor and I/O devices and memory & I/O devices. As we know not much of the problems arise for the data communication between microprocessor and memory as same technology is used in the manufacturing of memory and microprocessor.
The main reason for that the speed of the memory is almost compatible with the speed of 8085 microprocessor. Now the main concern is for the data transfer between the microprocessor and I/O devices. The main problems arise due to mismatch of the speed of the I/O devices and the speed of microprocessor or memory. To overcome this problem of speed mismatch between the microprocessor and I/O devices we have to do something. For that reason only we introduce data transfer schemes of 8085 microprocessor. So following data transfer schemes may be considered for smooth data transfer process. The data transfer schemes of 8085 microprocessor were categorised depending upon the capabilities of I/O devices for accepting serial or parallel data.



The 8085 microprocessor is a parallel device. That means it transfers eight bits of data simultaneously over eight data lines (parallel I/O mode). However in many situations, the parallel I/O mode is either impractical or impossible. For example, parallel data communication over a long distance becomes very

expensive. Similarly, parallel data communication is not possible with devices such as CRT terminal or Cassette tape etc.

Serial I/O mode transfer

For these devices and for these reasons serial I/O mode is used. In serial I/O mode transfer a single bit of data on a single line at a time. For serial I/O data transmission mode, 8-bit parallel word is converted to a stream of eight serial bit using parallel-to-serial converter. Similarly, in serial reception of data, the microprocessor receives a stream of 8-bit one by one which are then converted to 8- bit parallel word using serial-to-parallel converter. For this purpose data transfer schemes of 8085 microprocessor are introduced.

Parallel data transfer scheme

Parallel data transfer scheme is faster than serial I/O transfer. in parallel data transfer 8-bit data send all together with 8 parallel wire. In 8085 microprocessor mainly three types of parallel data transfer scheme we observed. Those are

- **Programmed I/O Data Transfer**
- **Interrupt Driven I/O Data Transfer**
- **Direct Memory Access (DMA) Data Transfer**

Programmed I/O Data Transfer scheme of 8085 microprocessor

Programmed I/O Data Transfer scheme of 8085 microprocessor is a simple parallel data transfer scheme. This method of data transfer is generally used in the simple microprocessor systems. It is obvious that where speed is unimportant. This method uses instructions to get the data into or out of the microprocessor. Programmed I/O Data Transfer scheme of 8085 microprocessor can be work on synchronous or asynchronous mode. The data

transfer can be synchronous or asynchronous it completely depends upon the type and the speed of the I/O devices.

Synchronous type of data transfer

Synchronous type of data transfer can be used when the speed of the I/O devices matches with the speed of the 8085 microprocessor. So for synchronization established between I/O device and microprocessor we need common clock pulse. This common clock pulse synchronizes the microprocessor and the I/O devices. Synchronous type of data transfer scheme because of the matching of the speed, the microprocessor does not have to wait for the availability of the data. The microprocessor immediately sends data for the transfer as soon as the microprocessor issues a signal.

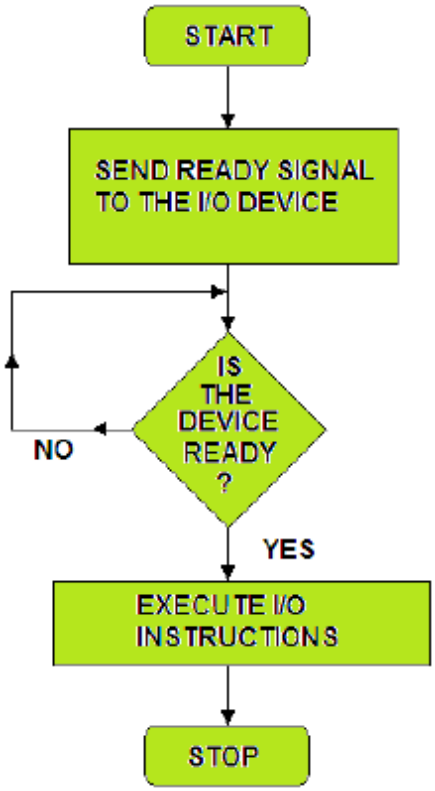
The asynchronous data transfer

The asynchronous data transfer method is used when the speed of the I/O devices is slower than the speed of the microprocessor. Because of the mismatch of the speed, the internal timing of the I/O device is independent from the microprocessor. That is why two units are said to be asynchronous to each other. The asynchronous data transfer is normally implemented using **'handshaking'** mode. Now question is what is handshaking mode? In the handshaking mode some signals are exchanged between the I/O device and microprocessor before the data transfer takes place.

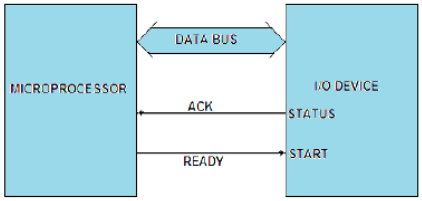
By this handshaking the microprocessor has to check the status to the input/output device. Now if the device is ready for the data transfer or not.

- First step of microprocessor is initiates the I/O device to get ready.
- Then status of the I/O device is continuously checked by the microprocessor.

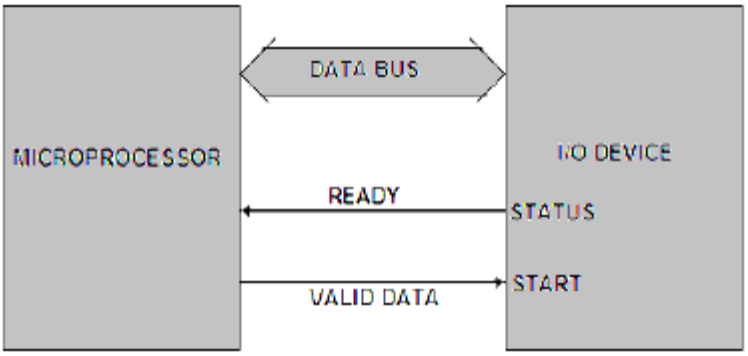
- This process remain continues until the I/O device becomes ready.
- After that microprocessor sends instructions to transfer the data.



Now form this bellow figure, the microprocessor sends a ready signal to I/O device. When the device is ready to accept the data, the I/O device sends an ‘ACK’ (Acknowledge) signal to microprocessor. By sending ACK, it indicating that the I/O device has acknowledged the ‘Ready’ signal. Now finally it is ready for the transfer of data.



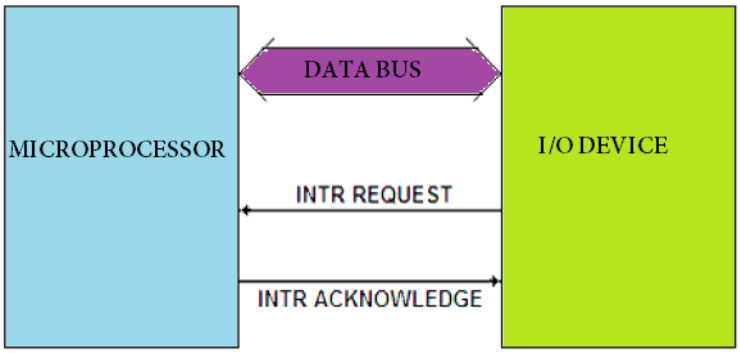
Again in bellow figure shows the asynchronous handshaking process to transfer the data from the I/O device to microprocessor. In this case I/O device issues the ready signal to microprocessor indicating that I/O device is ready to send the data to microprocessor. In response to this signal, valid data signal is sent by the microprocessor to I/O device and then the valid data is put on the data bus for the transfer.



Interrupt Driven I/O Data Transfer

As we saw that in the **programmed I/O data transfer method**, microprocessor is busy all the time in checking for the availability of data from the slower I/O devices. And it also busy in checking if I/O device is ready for the data transfer or not. In other words, in this data transfer scheme, some of the microprocessor time is wasted in waiting while an I/O device is getting ready. To overcome this problem interrupt driven I/O data transfer introduced.

The interrupt driven I/O data transfer method is very efficient because no microprocessor time is wasted in waiting for an I/O device to be ready. In this interrupt driven I/O data transfer method the I/O device informs the microprocessor for the data transfer whenever the I/O device is ready. This is achieved by interrupting the microprocessor. As we know that the interrupt is hardware facilities provided on the microprocessor.



Now come to the working process of interrupt driven I/O data transfer. So the beginning the microprocessor initiates data transfer by requesting the I/O device ‘to get ready’ and then continue executing its original program rather wasting its time by checking the status of I/O device. Whenever the device is ready to accept or supply data, it informs the processor through a control signal. This control signal known as interrupt (INTR) signal. In response to this interrupt signal, the microprocessor sends back an interrupt acknowledge signal to the I/O device. By sending acknowledgement it indicating that it received the request. It then suspends its job after executing the current instruction. It saves the contents and status of program counter to stack and jumps to the subroutine program.

This subroutine program is called **Interrupt Service Subroutine** (ISS) program. The ISS saves the processor status into stack; and after executing the instruction for the data transfer, it restores the processor status and then returns to main program.
several input/output devices may be connected to microprocessor using Interrupt Driven Data Transfer Scheme. Following interrupt request configuration may arise while interfacing the I/O devices to microprocessor.

1. Single Interrupt system
2. Multi Interrupt System

Single Interrupt System

When only one interrupt line is available with the microprocessor and several I/O devices are to be connected, then the method is known as Single Interrupt System.

Multi Interrupt System

When the microprocessor has several interrupt terminals and one I/O device is to be connected to each interrupt terminal, then it is known as multi interrupt system. In this scheme, the number of I/O devices to be connected to the interrupt lines should be equal to or less than the number of interrupt terminals. In this way one device is connected to each level of interrupt. So when a device interrupts the microprocessor, it immediately knows which device has interrupted. Such an interrupt scheme is known as vectored interrupt.

Direct Memory Access (DMA) Data Transfer

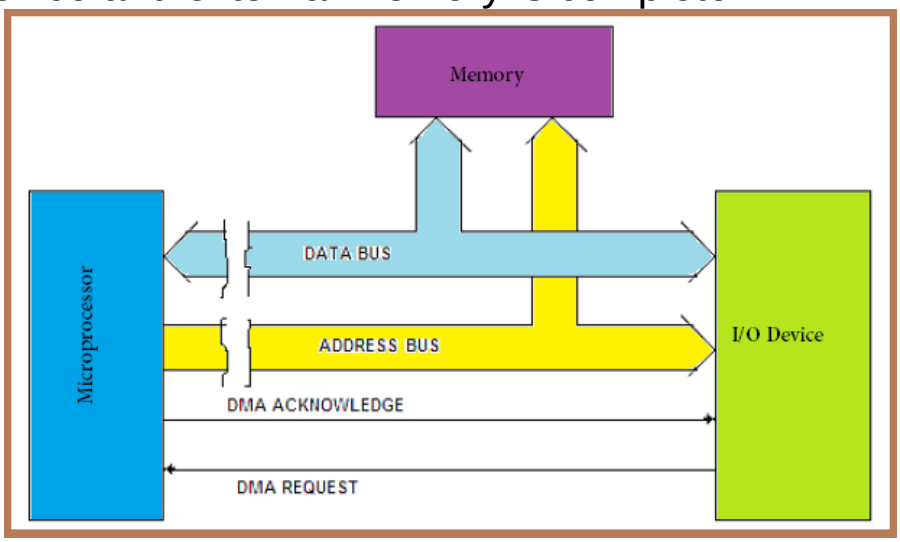
As we discussed earlier that in programmed I/O or interrupt driven I/O methods of data transfer between the I/O devices and external memory is via the accumulator. Now think for bulk data transfer from I/O devices to memory or vice-versa, these two methods discussed above are time consuming and quite uneconomical even though the speed of I/O devices matches with the speed of microprocessor. Because in those methods the data is first transferred to accumulator and then to concerned device.

To overcome those problem direct memory access data transfer method is introduced. The Direct Memory Access (DMA) data transfer method is used for bulk data transfer from I/O devices to microprocessor or vice-versa. In this method I/O devices are allowed to transfer the data directly to the external memory without being routed through accumulator. For this reason the

microprocessor relinquishes the control over the data bus and address bus, so that these can be used for transfer of data between the devices.

Working principle of direct memory access data transfer

So now come to working principle of direct memory access data transfer. For the data transfer using DMA process, a request to the microprocessor in form of HOLD signal, by the I/O device is sent. When microprocessor receipt of such request, the microprocessor relinquishes the address and data buses and informs the I/O devices of the situation by sending Acknowledge signal HLDA. The I/O device withdraws the request when the data transfer between the I/O device and external memory is complete.



If we discuss in brief about working principal of DMA controller. Then we should mention that DMA controller is used with the microprocessor that helps to generate the addresses for the data to be transferred from the I/O devices. The peripheral device sends the request signal (DMARQ) to the DMA controller and the DMA controller in turn passes it to the microprocessor (HOLD signal). On receipt of the DMA request the microprocessor sends an acknowledge signal (HLDA) to the DMA controller. On receipt of this signal (HLDA) the DMA controller sends a DMA acknowledge signal (DMACK) to the I/O device. The DMA

controller then takes over the control of the buses of microprocessor and controls the data transfer between RAM and I/O device. When the data transfer is complete, DMA controller returns the control over the buses to the microprocessor by disabling the HOLD and DMACK signals.

Now question is how many way DMA can work? It may be mentioned here that DMA transfer the data of the following types:

- Memory to I/O device
- I/O device to memory
- Memory to memory
- I/O device to I/O device

8086 microprocessor

(by Garima Rohela)

8086 Microprocessor features:

1. It is 16-bit microprocessor
2. It has a 16-bit data bus, so it can read data from or write data to memory and ports either 16-bit or 8-bit at
a time.
3. It has 20 bit address bus and can access up to 220 memory locations (1 MB).
4. It can support up to 64K I/O ports
5. It provides 14, 16-bit registers
6. It has multiplexed address and data bus AD0-AD15 & A16-A19
7. It requires single phase clock with 33% duty cycle to provide internal timing.
8. Prefetches up to 6 instruction bytes from memory and queues them in order to speed up the processing.
9. 8086 supports 2 modes of operation
a. Minimum mode
b. Maximum mode

Architecture of 8086 microprocessor :

As shown in the figure, the 8086 CPU is divided into two independent functional parts

- Bus Interface Unit(BIU)
- Execution Unit(EU)

Dividing the work between these two units speeds up processing.

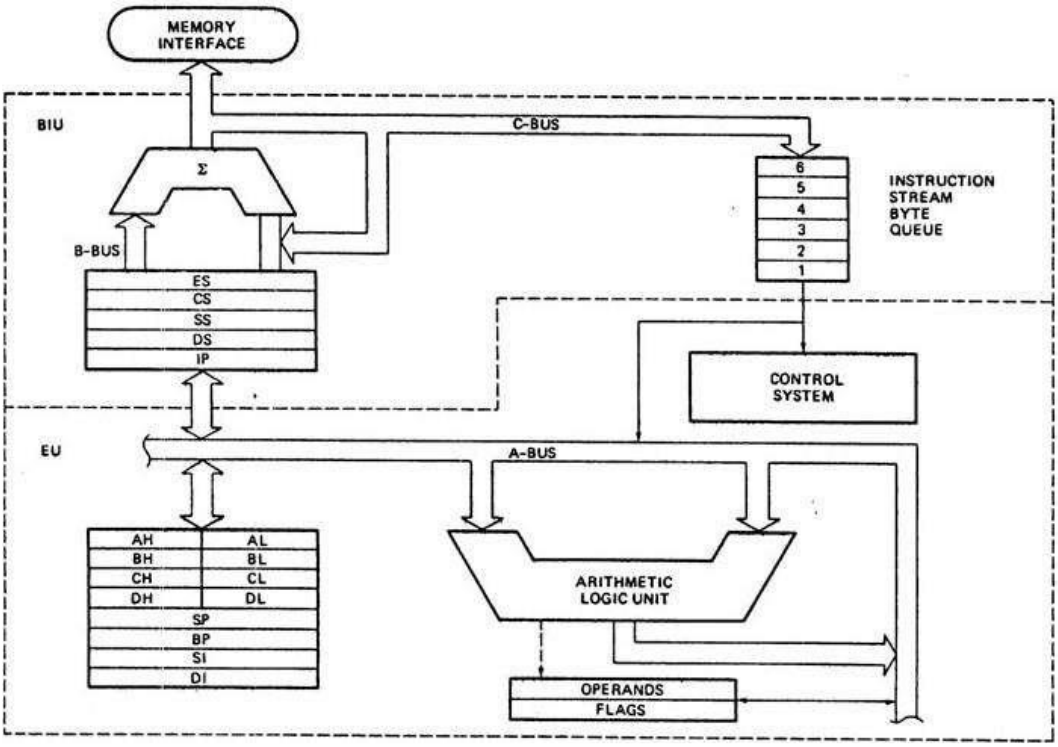
BIU (Bus interface unit):

- It handles all transfers of data and addresses on the buses for the execution unit.
- Sends out addresses

- Fetches instructions from memory.
- Read / write data from/to ports and memory i.e. handles all transfers of data and addresses on the busses

EU (Execution unit):

- Tells BIU where to fetch instructions or data from
- Decodes instructions
- Executes instructions



Functional Block Diagram of 8086 Microprocessor

Instruction Decoder & ALU:

Decoder in the EU translates instructions fetched from the memory into a series of actions which the EU carries out.16-bit ALU in the EU performs actions such as AND, OR, XOR, increment, decrement etc.

FLAG Register:

It is a 16-bit register. 9-bit are used as different flags, remaining bits unused

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Fig: 16-bit flag register

Out of 9-flags, 6 are conditional (status) flags and three are control flags

Conditional flags:

These are set or reset by the EU on the basis of the results of some arithmetic or logic operation. 8086 instructions check these flags to determine which of two alternative actions should be done in executing the instructions.

1. OF (Overflow flag): is set if there is an arithmetic overflow, i.e. the size of the result exceeds the capacity of the destination location.

2. SF (Sign flag): is set if the MSB of the result is 1

3. ZF (Zero flag): is set if the result is zero

4. AF (Auxiliary carry flag): is set if there is carry from lower nibble to upper nibble or from lower byte to upper byte

5. PF (Parity flag): is set if the result has even parity

6. CF (Carry flag): is set if there is carry from addition or borrow from subtraction

Control flags:

They are set using certain instructions. They are used to control certain operations of the processor.

1. **TF (Trap flag):** for single stepping through the program
2. **IF (Interrupt flag):** to allow or prohibit the interruption of a program
3. **DF (Direction flag):** Used with string instructions

General purpose Registers (GPRs):

There are 8 GPRs AH, AL (Accumulator), BH, BL, CH, CL, DH, DL are used to store 8 bit data.

AL register is also called the accumulator

Used individually for the temporary storage of data

GPRs can be used together (as register pair) to store 16-bit data words. Acceptable register pairs are:

AH-AL pair AX register

BH-BL pair BX register (to store the 16-bit data as well as the base address of the memory location)

CH-CL pair CX register (to store 16-bit data and can be used as counter register for some instructions like loop)

DH-DL pair DX register (to store 16-bit data and also used to hold the result of 16-bit data multiplication and division operation)

Pointer and Index registers:

SP (Stack Pointer), BP (Base pointer), SI (Source Index), DI (Destination index)

Pointer Registers:

The two pointer registers, SP and BP are used to access data in the stack segment. The SP is used as offset from current Stack Segment during execution of instruction that involve stack. SP is automatically updated. BP contains offset address and is utilized in based addressing mode. Overall, these are used to hold the offset address of the stack address.

Index Registers:

EU also contains a 16-bit source index (SI) register and 16-bit destination index (DI) register. These registers can be used for temporary storage of data similarly as the general purpose registers. However they are specially to hold the 16-bit offset of the data word. SI and DI are used to hold the offset address of the data segment and extra segment memory respectively.

Bus Interface Unit:

The QUEUE:

When EU is decoding or executing an instruction, bus will be free at that time. BIU pre-fetches up to 6-instructions bytes to be executed and places them in QUEUE. This improves the overall speed because in each time of execution of new instruction, instead of sending address of next instruction to be executed to the system memory and waiting from the memory to send back the instruction byte, EU just picks up the fetched instruction byte from the QUEUE.

The BIU stores these pre-fetched bytes in a first-in-first-out (FIFO) register set called a queue. Fetching the next instruction while the current instruction executes is called pipelining.

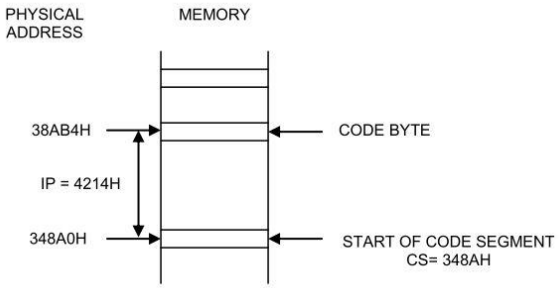
Segment Registers:

The BIU contains a dedicated address, which is used to produce the 20 bit address. The bus control logic of the BIU generates all the bus control signals, such as the READ and WRITE signals, for memory and I/O. The BIU also has four 16 bit segments registers namely:

1. **Code segment:** holds the upper 16-bits of the starting addresses of the segment from which BIU is currently fetching instruction code bytes.
2. **Stack segment:** store addresses and data while subprogram executes
3. **Extra segment:** store upper 16-bits of starting addresses of two memory segments that are used for data.
4. **Data segment:** store upper 16-bits of starting addresses of two memory segments that are used for data.

Code Segment Register (CS) and Instruction Pointer (IP)

All program instructions located in memory are pointed using 16 bits of segment register CS and 16 bits offset contained in the 16 bit instruction pointer (IP). The BIU computes the 20 bit physical address internally using the logical address that is the contents of CS and IP. 16 bit contents of CS will be shifted 4 bits to the left and then adding the 16 bit contents of IP. Thus, all instructions of the program are relative contents of IP. Simply stated, CS contains the base or start of the current code segment, and IP contains the distance or offset from this address to the next instruction byte to be fetched.

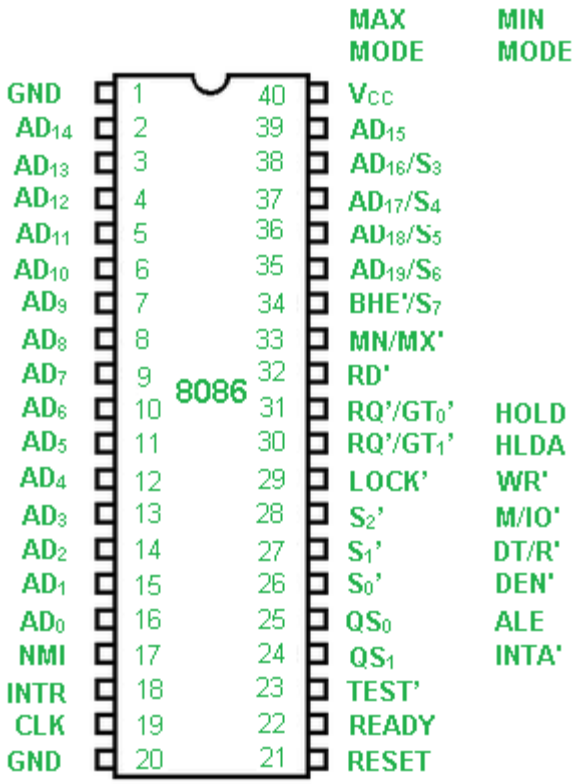


Above fig shows addition of IP to CS to produce the physical address of code byte Stack Segment Register (SS) and Stack Pointer (SP)

The stack segment registers points to the current stack. The 20 bit physical stack address is calculated from the SS and SP. The programmer can also use Base Pointer (BP) instead of SP for addressing. In this case, the A stack is a section of memory to store addresses and data while a subprogram is in progress. 20 bit physical address is calculated using SS and BP.

Pin diagram of 8086 microprocessor

Pin diagram of 8086 microprocessor is as given below:



Intel 8086 is a 16-bit HMOS microprocessor. It is available in 40 pin DIP chip. It uses a 5V DC supply for its operation. The 8086 uses 20-line address bus. It has a 16-line data bus. The 20 lines of the address bus operate in multiplexed mode. The 16-low order address bus lines have been multiplexed with data and 4 high-order address bus lines have been multiplexed with status signals.

AD0-AD15 : Address/Data bus. These are low order address bus. They are multiplexed with data. When AD lines are used to transmit memory address the symbol A is used instead of AD, for example A0-A15. When data are transmitted over AD lines the symbol D is used in place of AD, for example D0-D7, D8-D15 or D0-D15.

A16-A19 : High order address bus. These are multiplexed with status signals.

S2, S1, S0 : Status pins. These pins are active during T4, T1 and T2 states and is returned to passive state (1,1,1 during T3 or Tw (when ready is inactive). These are used by the 8288 bus controller for generating all the memory and I/O operation) access control signals. Any change in S2, S1, S0 during T4 indicates the beginning of a bus cycle.

S2	S1	S0	CHARACTERISTICS
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive state

A16/S3, A17/S4, A18/S5, A19/S6 :
The specified address lines are multiplexed with corresponding status signals.

A17/S4	A16/S3	FUNCTION
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

BHE'/S7 : Bus High Enable/Status. During T1 it is low. It is used to enable data onto the most significant half of data bus, D8-D15. 8-bit device connected to upper half of the data bus use BHE' (Active Low) signal. It is multiplexed with status signal S7. S7 signal is available during T2, T3 and T4.
RD': This is used for read operation. It is an output signal. It is active when low.

READY : This is the acknowledgement from the memory or slow device that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the microprocessor. The signal is active high(1).

INTR : Interrupt Request. This is triggered input. This is sampled during the last clock cycles of each instruction for determining the availability of the request. If any interrupt request is found pending, the processor enters the interrupt acknowledge cycle. This can be internally masked after resulting the interrupt enable

flag. This signal is active high(1) and has been synchronized internally.

NMI : Non maskable interrupt. This is an edge triggered input which results in a type II interrupt. A subroutine is then vectored through an interrupt vector lookup table which is located in the system memory. NMI is non-maskable internally by software. A transition made from low(0) to high(1) initiates the interrupt at the end of the current instruction. This input has been synchronized internally.

INTA' : Interrupt acknowledge. It is active low(0) during T2, T3 and Tw of each interrupt acknowledge cycle.

MN/MX' : Minimum/Maximum. This pin signal indicates what mode the processor will operate in.

RQ'/GT1′, RQ'/GT0′ : Request/Grant. These pins are used by local bus masters used to force the microprocessor to release the local bus at the end of the microprocessor's current bus cycle. Each of the pin is bi-directional. RQ'/GT0′ have higher priority than RQ'/GT1′.

LOCK' : Its an active low pin. It indicates that other system bus masters have not been allowed to gain control of the system bus while LOCK' is active low(0). The LOCK signal will be active until the completion of the next instruction.

TEST' : This examined by a 'WAIT' instruction. If the TEST pin goes low(0), execution will continue, else the processor remains in an idle state. The input is internally synchronized during each of the clock cycle on leading edge of the clock.

CLK : Clock Input. The clock input provides the basic timing for processing operation and bus control activity. Its an asymmetric square wave with a 33% duty cycle.

RESET : This pin requires the microprocessor to terminate its present activity immediately. The signal must be active high(1) for at least four clock cycles.

Vcc : Power Supply(+5V D.C.)
GND : Ground

QS1,QS0 : Queue Status. These signals indicate the status of the internal 8086 instruction queue according to the table shown below

QS1	QS0	STATUS
0	0	No operation
0	1	First byte of op code from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

DT/R : Data Transmit/Receive. This pin is required in minimum systems, that want to use an 8286 or 8287 data bus transceiver. The direction of data flow is controlled through the transceiver.

DEN : Data enable. This pin is provided as an output enable for the 8286/8287 in a minimum system which uses transceiver. DEN

is active low(0) during each memory and input-output access and for INTA cycles.

HOLD/HOLDA : HOLD indicates that another master has been requesting a local bus .This is an active high(1). The microprocessor receiving the HOLD request will issue HLDA (high) as an acknowledgement in the middle of a T4 or T1 clock cycle.

ALE : Address Latch Enable. ALE is provided by the microprocessor to latch the address into the 8282 or 8283 address latch. It is an active high(1) pulse during T1 of any bus cycle. ALE signal is never floated, is always integer.

Difference between Minimum Mode and Maximum Mode

Minimum mode	Maximum Mode
In minimum mode there can be only one processor i.e. 8086.	In maximum mode there can be multiple processors with 8086, like 8087 and 8089.
MN/MX is 1 to indicate minimum mode.	MN/MX is 0 to indicate maximum mode.
ALE for the latch is given by 8086 as it is the only processor in the circuit.	ALE for the latch is given by 8288 bus controller as there can be multiple processors in the circuit.
DEN¯and DT/R for the trans-receivers are given by 8086 itself.	DT/R' for the trans-receivers are given by 8288 bus controller.
Direct control signals M/IO', RD' and WR' are given by 8086.	Instead of control signals, each processor generates status signals called S_2', S_1' and S0'
Control signals M/IO' RD' and WR' are decoded by a 3:8 decoder like 74138.	Status signals S_2', S_1' and S_0' are decoded by a bus controller like 8288 to produce control signals.
INTA'is given by 8086 in response to an interrupt on INTR line.	INTA¯ is given by 8288 bus controller in response to an interrupt on INTR line.

Differences between 8085 and 8086 microprocessor

In the changing world of technologies, the devices used are also changing. Let us take a look at the changes between 8085 series of microprocessors and 8086 series of microprocessors.

SERIAL NO.	8085 MICROPROCESSOR	8086 MICROPROCESSOR
1	The data bus is of 8 bits.	The data bus is of 16 bits.
2	The address bus is of 16 bits.	The address bus is of 20 bits.
	3. The memory capacity is 64 KB.Also 8085 Can Perform Operation Upto 2^8 ie. 256 numbers. A number greater than this is to taken multiple times in 8 bit data bus.	The memory capacity is 1 MB.Also 8086 Can Perform Operation upto 2^16 ie. 65,536 numbers.
	4. The input/output port addresses are of 8 bits.	The input/output port addresses are of 8 bits.

SERIAL NO.	8085 MICROPROCESSOR	8086 MICROPROCESSOR
5	The operating frequency is 3.2 MHz.	The operating frequency is 5 MHz, 8MHZ,10MHZ.
		8086 MP has Two Modes Of Operation. 1. Minimum Mode = SingLe CPU PROCESSOR 2. Maximum Mode = Multiple CPU PROCESSOR.
	6.8085 MP has Single Mode Of Operation.	
	7.It not have multiplication and division instructions.	It have multiplication and division instructions.
	8.It does not support pipe-lining.	It supports pipe-lining as it has two independent units Execution Unit (EU) and Bus Interface Unit (BIU).
	9.It does not support instruction queue.	It supports instruction queue.

SERIAL NO.	8085 MICROPROCESSOR	8086 MICROPROCESSOR
	10.Memory space is not segmented.	Memory space is segmented.
	11.It consists of 5 flags(Sign Flag, Zero Flag, Auxiliary Carry Flag, Parity Flag, Carry Flag).	It consists of 9 flags(Overflow Flag, Direction Flag, Interrupt Flag, Trap Flag, Sign Flag, Zero Flag, Auxiliary Carry Flag, Parity Flag, Carry Flag).