MALP_Assignment_I 2
MALP_Assignment_II 34
MALP_Assignment_III 43

MALP

Assignment - I

Page:
Date:

Microprocessor Architecture & Instruction Set

Q.1) Draw the PIN Diagram of 8085 & explain details of each pin.

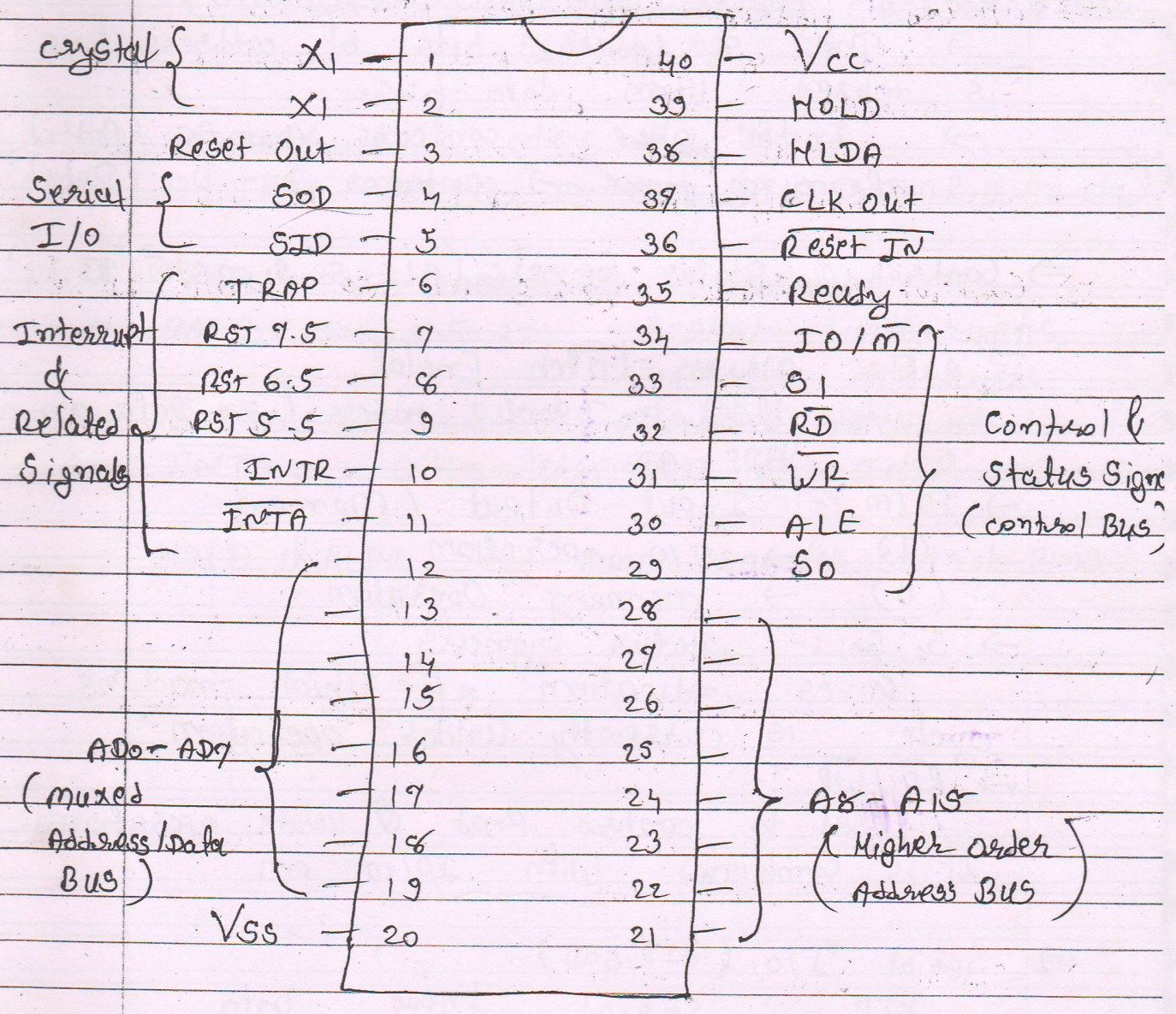


fig 1.1 :- Pin diagram of 8085

⇒ Higher Order Address Bus (A8 - A15) :-

→ The 8085 Address Bus is of 16 Bits.

→ The Upper 8 bits (A15 - A8) are unidirectional.

⇒ Mixed Address / Data Bus (AD7 - AD0)

→ The 8 lower bits of address bus is mixed with data bus.

→ Initial phase → contains A7 - A0 (Address)

Remaining time → contains D7 - D0 (Data)

⇒ Control & Status Signals (ALE, S0, S1, IO/$\overline{M}$, $\overline{RD}$, $\overline{WR}$)

→ ALE : Address latch Enabul

↳ Used to Demux Address from Data on AD7 - AD0

→ IO/$\overline{M}$:- Input - Output / Memory

(1) → I/O operation

(0) → memory Operation

→ S_1, S_0 :- Status Signals

Gives indication of which machine cycle is currently under operation.

→ $\overline{RD}$ / $\overline{WR}$:-

Used to control Read & Write operations. It is combined with IO/$\overline{M}$ pin

⇒ Serial I/O (SID, SOD)

SID → Serial Input Data

SOD → Serial Output Data

Page:
Date:

⇒ Power Supply & clk :- (Vcc, Vss, X_1, X_2)

Vcc → To +5V

Vss → to 0 V (ground

Between

X_1 & X_2 → clock Crystal.

CLK out → pin to output clock to Other Systems.

⇒ Interrupt & other Signals :-

(TRAP, RST 7.5, RST 6.5, RST 5.5, INTR, INTA etc)

TRAP :- Non maskable Interrupt.

RST 7.5, 6.5, 5.5 :- Restart interrupts maskable interrupts

INTR :- General purpose interrupt.

$\overline{INTA}$:- Acknowledgement of INTR.

HOLD & HLDA :- DMA transfer control pins.

Page :
Date :

Q.2 Describe the functions of 8085 μP Pins SOD, SID, HOLD, HLDA, READY, ALE, TRAP ~~PINS.~~

⇒ SOD :- Serial Output Data
It is used to send Data Serially to other peripheral Devices.

⇒ SID :- Serial Input Data
It is used to Receive Data serially from peripheral Devices.

⇒ HOLD :-
It is request of DMA controller to 8085 for Bus access.

⇒ HLDA :- Hold Acknowledgement
It is Acknowledgement of 8085 to DMA controller for HOLD request.

⇒ READY :-
It is used to synchronize the operation with Slower peripheral.
(1) → Device is ready for RD or WR operation.

⇒ ALE :- Address Latch enable.
Used to Demux AD7 - AD0.

ALE { (1) → A7 - A0 : Address
(0) → D7 - D0 : Data

Page :
Date :

⇒ TRAP :- non maskable interrupt
It is used for emergency situations like power failure.

Page:
Date:

Q-3 Draw & explain the internal arch. of 8085

⇒ Accumulator :- 8 bit
It is 8 bit register & hours of ~~all~~ the most of the instruction.

⇒ General Purpose Registers :- (B,C, D,E, H, L)
→ Six 8 bit general Purpose registers in 8085
→ Can also be used d register pairs of 16 bit
BC , DE, HL → 16 bit

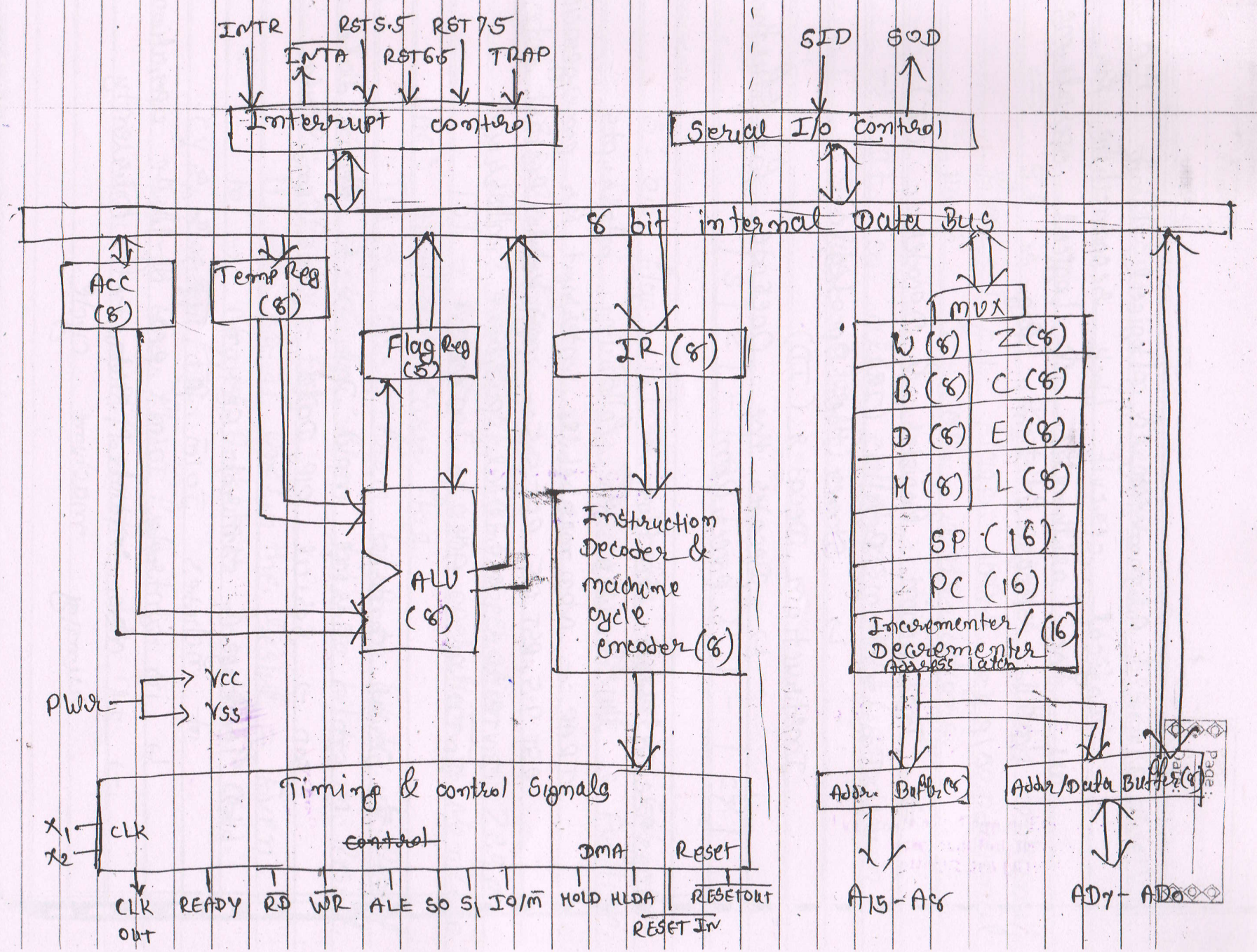
⇒ Program Counter :- 16 bit
→ Stores the address of the next instruction to be executes.

⇒ Stack Pointer :- 16 bit
→ Stores the address of top of the stack.

⇒ Flag Register :-

S	Z	-	AC	-	P	-	CY

S: Sign → (0) → Positive (1) → Negative
Z: Zero → (0) → Result non zero, (1) → Result zero
AC: Auxilliary carry → (0) No BCD overflow, (1) → BCD overflow
P - Parity → (0) → odd parity, (1) even parity
Cy - Carry → (0) → No carry (1) carry.


INTR
RST5.5
RST7.5
INTA
RST6.5
TRAP
Interrupt control
SID
SOD
Serial I/O control
8 bit internal Data Bus
ACC (8)
Temp Reg (8)
Flag Reg (5)
IR (8)
MUX
W (8)
Z (8)
B (8)
C (8)
D (8)
E (8)
H (8)
L (8)
SP (16)
PC (16)
Incrementer / Decrementer (16)
Address latch
ALU (8)
Instruction Decoder & machine cycle encoder (8)
PWR
Vcc
Vss
Timing & control Signals
X1
X2
CLK
~~Control~~
DMA
Reset
CLK OUT
READY
RD
WR
ALE
S0
S1
IO/M
HOLD
HLDA
RESETOUT
RESET IN
Addr. Buffer (8)
Addr/Data Buffer (8)
A15 - A8
AD7 - AD0

(8)

⇒ ALU :- Arithmetic & Logical Unit
Logical circuit block responcible for all the arithmatic & Logical operations
i/ps :- Acc & Temp Reg.
o/p :- Acc

⇒ Instruction Register & Decoder :- (8)
Instruction Register (IR)
↳ Stores the Opcodes
Instruction Decoder (ID)
↳ Decodes the Opcode for further execution.

⇒ Interrupt Section :-
It includes the following interrupts
TRAP :- Non maskable interrupt for emergencies
RST 7.5, RST 6.5, RST 5.5 :- Maskable Restart interrupt
INTR :- General purpose interrupt
$\overline{INTA}$:- ACK to INTR.

⇒ Serial control :-
SID → Serial input Data → to Receive Data
SOD → Serial O/P Data → to Send Data

⇒ Timing & control Section.
it includes $IO/\overline{M}$, $\overline{RD}$, $\overline{WR}$, S_0, S_1
↳ it controls IO/M Read & Write operation
→ It also indicate status of currently running machine cycle.

Page :
Date :

Q.4 Draw flag register of 8085 & explain it.

⇒ It is an 8-bit register which contains five 1-bit flags used by conditional jump & call instructions.

→ It is also called as PSW (Program Status Word).

→

S	Z	-	AC	-	P	-	CY

S : Sign flag :- [1] → indicates Results is Negative
[0] → indicates Result is Positive.

Z : Zero flag :-
[1] → Result of Last operation is Zero
[0] → Result of Last operation is NonZero

AC : Auxilliary Carry flag
[1] → Carry from bit 3 to bit 4 in Result
[0] → No Carry from bit 3 to bit 4 in Result.

P : Parity flag.
[1] → No. of One's in the result = Even
[0] → No. of One's in th result = Odd.

CY : Carry flag.
[1] → overflow from bit 7
[0] → no overflow from bit 7.

Q.5 Draw & Explain μP Bus Organisation.

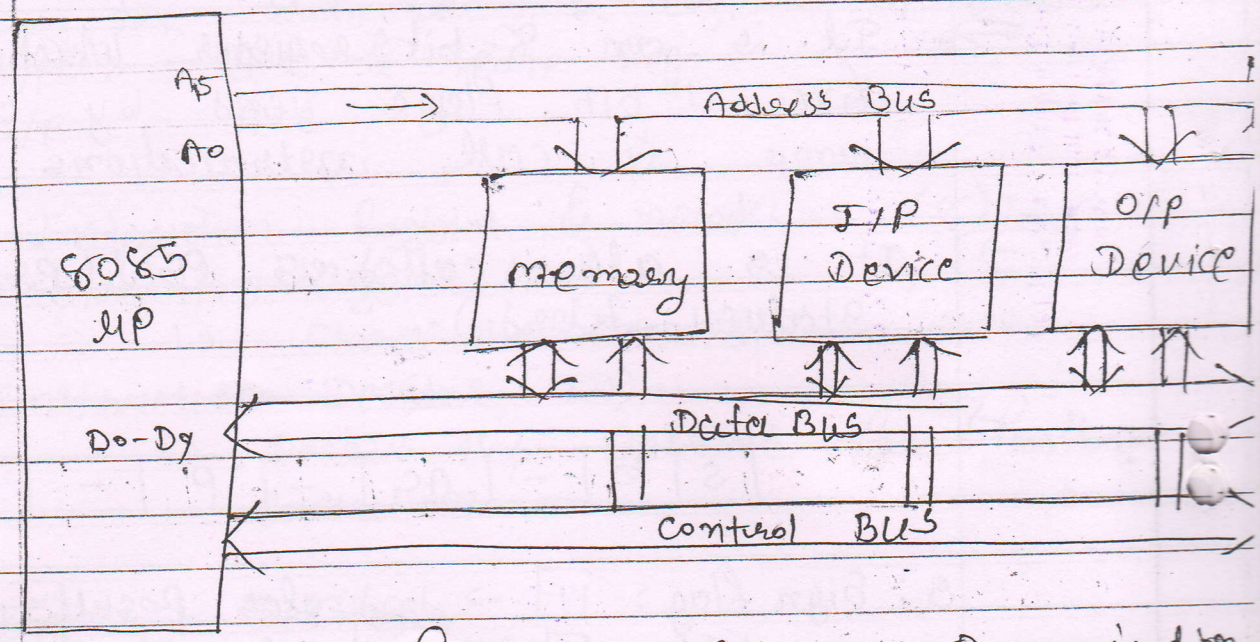


Fig. 5-1 8085 Bus Organisation.

⇒ Bus is a group of wires used to transfer information from one source to another Dest device.

⇒ In 8085 Bus transfers bits from μP to peripheral devices & vice versa.

⇒ The 8085 Bus is composed of 3 diff types of Buses.

i) Data Bus
ii) Address Bus
iii) Control Bus.

Page :
Date :

i) Data Bus :-

→ It is 8 bit Bidirectional Bus.

→ It is used to send & receive Data from 8085. to peripheral.

ii) Address Bus :-

→ It is 16 bit unidirectional Bus.

→ The No. of lines in the address bus also determines the total amount of memory that can be accessed by 8085.

→ 16 address lines ⇒ 2^{16} = 64 KB mem.

iii) Control Bus :-

→ It consists of various ~~sig~~ control & status signals.

→ The Signals – ALE, S_0, S_1, $\overline{RD}$, $\overline{WR}$, $IO/\overline{M}$.

ALE – Address Latch Enable used to demux address from Data

$IO/\overline{M}$ – I/P / memory

S_1, S_0 – Status Signals gives indication of which machine cycle is currently under operation

$\overline{RD}/\overline{WR}$ –

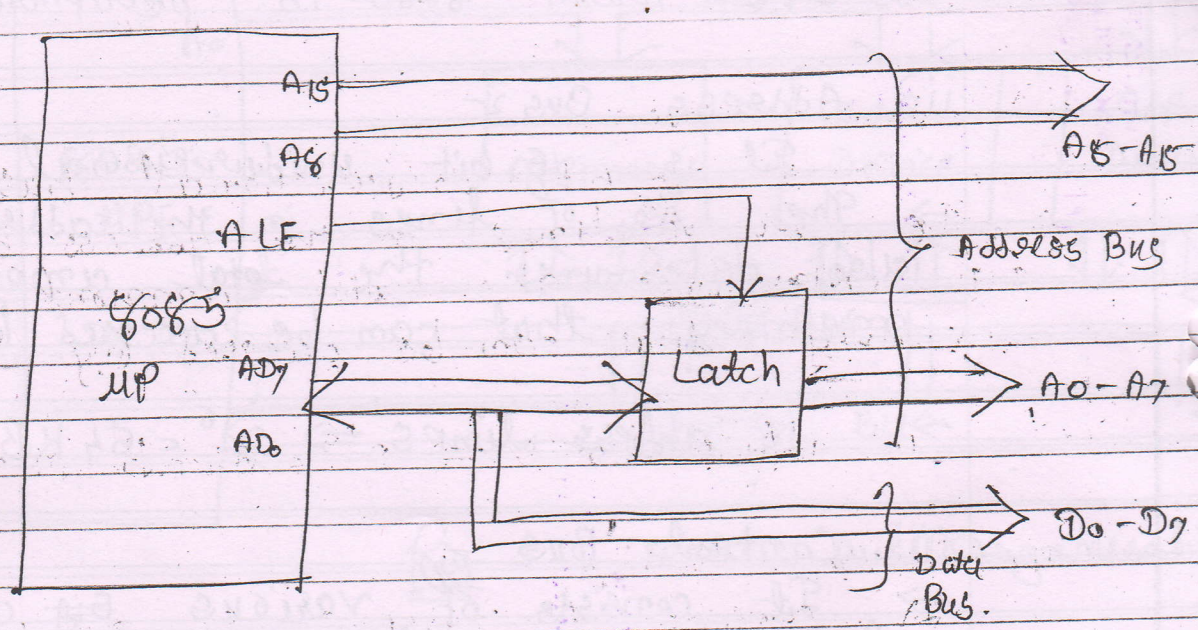
Used to control Read & Write operation

It is combined with $IO/\overline{M}$ pin.

Page :
Date :

Q.6 Explain With Diagram the the Demultiplexing of Muxed Address / Data Bus of 8085 µP

⇒



⇒ The Muxed Address / Data bus AD7-AD0 is used to transfer data & address Both.

→ Address Bus is of Width 16 bit, of which A15-A8 (Higher order address Bus) & AD7-AD0 (Muxed Address / Data BUS)

→ Lower order address Bus AD7-AD0 is muxed.

→ The ALE Signal is used to Demux AD7-AD0

Page :
Date :

ALE → (1) → Address is latched into latch (A_7-A_0)
(0) → Data is available on (D_7-D_0)

Q.7 Define following
→

① Bus :-
→ It is group of wires used to carry Information.

② Opcode :- Operation Code
→ An opcode is 1st byte of an instruction in machine language which tells the H/W what operation to be performed with this instruction.

③ Machine Cycle :-
It is time required by μP to complete the operation of accessing memory or I/O devices is called machine cycle.

④ Memory Map :-
A memory map is a structure of Data that indicates how memory is laid out.

⑤ Instruction Cycle :-
The time period during which one instruction is fetched from memory &

Page :
Date :

executed ~~from~~ When a computer is given an instruction in machine ~~cycle~~ language.

(6) Operand:
An operand is the part of computer instruction that specifies data that is to be operating on.

(7) Architecture:
Computer processor architecture is set of rules & methods that describes the functionality, organisation & implementation

Q.8 Classifiy instruction set of 8085 µP on the basis of their operation. Given one example of each one

⇒ The 8085 instruction set are classified into five functional groups.

1) Data transfer Group
2) Arithmetic group
3) Logic Operation group
4) Branch group
5) Stack, I/O, & machine control group.

Page :
Date :

① Data transfer Group :-
→ This group of instructions move the data between Registers or between memory & registers.

→ e.g MOV, MVI, LXI, LDA, STA, LHLD, SHLD.

② Arithmetic instructions group :-
→ This group of instructions perform the arithmetic operations like addition subtraction, increment, Decrement on contents of registers or memory location

→ e.g ADD, ADI, ADC, ACI, INR, DCR, INX, DCX etc

③ Logical instruction Group :-
→ This group of instructions perform the various logical operations.

→ e.g. ANA, ANI, ORA, ORI, XRA, XRI, CMP, CMA, CMC etc

④ Branch Group :-
→ This group of instruction allow the transfer of control unconditionally or conditionally on the fullfillment of certain conditions on execution.

→ e.g. JUMP, CALL, RET etc.

Page:
Date:

(5) Stack I/O & machine control Group

⇒ This group of instructions for stack operations like push & pop, instru to read & write the I/O ports. Setting & reading the interrupt masks & setting & clearing the flags.

→ eg PUSH, POP, IN, OUT, NOP, HLT, RIM, SIM

Page :
Date :

Q.9 Explain below instruction with suitable example.

① LDA :- Load Accumulator Direct

⇒ format :- LDA <16 bit Address>

⇒ e.g. LDA 2050H

⇒ Description :- It will load the Accumulator with the contents of the memory Location Directly specified (e.g. 2050) in the example

② ⇒ Addressing mode :- Direct Addressing mode.

② LDAX :- Load Accumulator indirect (extended)

⇒ Format :- LDAX Rp

⇒ e.g :- LDAX B

⇒ Description :- It will load the accumulator with the contents of the memory location Indirectly specified in instruction.
Here Register pair Rp (e.g. BC) is containing Address of the Data (e.g. 2050 stored in BC).

⇒ Addressing mode :- Indirect Addressing mode

Page:
Date:

③ LXI :- Load Register pair Immediatly.

⇒ Format :- LXI Rp, <16 bit Data>
∴ Addressing mode :- Immidiate Addressing Mode
⇒ e.g. :- LXI B, 2050H

⇒ Description :- It loads 16 bit data specified immeadiatly (e.g. 2050H) in instruction to RP Register pair (e.g. ~~2050~~ BC)

④ LHLD :- Load H & L Register Direct

⇒ Format :- LHLD <16 bit address>
→ Addressing Mode :- ~~Indirect~~ Direct Addressing mode
⇒ e.g. LHLD 2050H

⇒ Description :- This instruction copies the contents of memory location pointed out by the 16 bit address in register L & copies the contents of next memory location in register H.

Page :
Date :

(5) SHLD :- Store H & L Registers Direct

=> Format :- SHLD <16 bit addr>
Addressing mode :- Direct Addressing mode
=> e.g. :- SHLD 2050H

=> Desc :- The contents of L are stored in the memory location specified by the 16 bit address in the operand, & the contents of H reg are stored in the next memory location.

(6) XTHL :- Exchange H & L with top of the stack

=> Formcat :- XTHL
=> Addressing mode :- Implied Addressing mode
=> e.g. : XTHL

=> Desc :- Contents of H are exchanges with top of the stack. & contents of L are exchanged with the next location

H <-> (SP)
L <-> (SP+1)

Page :
Date :

(7) XCHG :- Exchange H & L with D & E.

⇒ format :- XCHG (No operand)
⇒ Addressing mode :- Implied Addressing
⇒ e.g. : XCHG

⇒ Description :-
The contents of H are exchanged with the contents of D Reg. & the contents of L are exchanged with the contents of Reg. E.

H ⟷ D
L ⟷ E

(8) PCHL :- Load Program Counter with HL content
⇒ Format :- PCHL (No operand)
⇒ Addressing mode :- Implied Addressing.
⇒ e.g. :- PCHL

⇒ Desc :- The contents of H are placed in to PCH (higher byte of PC) & L are placed in PCL (lower byte of PC)

PCH ⟷ H ,
PCL ⟷ L .

Page:
Date:

(9) SPHL :- Copy H & L Reg to the Stack Pointer.

=> Format :- SPHL (no operand)
=> Addressing mode :- Implied addressing.
=> e.g :- SPHL

=> Desc :- Copies the contents of ~~SP~~ H & L registers into SP.

SPH ← H
SPL ← L

(10) DAD :- Add register pair to HL Reg. pair

=> Faormcuit :- DAD Rp
=> Addressing mode :- Register Addressing
=> e.g. :- DAD D

=> Desc :-
The 16 bit content of specified register pair (e.g DE) are added to the contents of the HL reg. pair. The Result is stored in HL Reg pair.

Page :
Date :

(11) DAA :- Decimal Adjust accumulator.

⇒ Format :- DAA (No operand)
⇒ Addressing mode :- Implied Addressing.
⇒ e.g. :- DAA

⇒ Desc :- The contents of the accumulator are changed from a binary value to two 4 bit binary coded Decimal (BCD) digits.

(12) CMA :- Complement Accumulator

⇒ Format :- CMA (No operand)
e.g. :- CMA
→ Addressing :- Implied Addressing
⇒ Desc :- The contents of Accumulator are complemented.

(13) CMC :- Complement Carry :-
→ Format :- CMC (No operand)
e.g. :- CMC

⇒ Desc :- The carry flag is complemented

→ Addressing mode :- Implied Addressing

Page :
Date :

(14) STC :- Set Carry

⇒ Format :- STC (No operand)
e.g :- STC
→ Addressing mode :- Implied Addressing
⇒ DESC :- The carry flag is set to 1.

(15) RLC :- Rotate Acc. Left

⇒ Format :- RLC (No operand)
⇒ eg. :- RLC
→ Addressing mode :- Implied addressing.

⇒ Description :-
Each binary bit of ACC are rotated left by one position.
Bit D_7 is placed into D_0 as well as carry

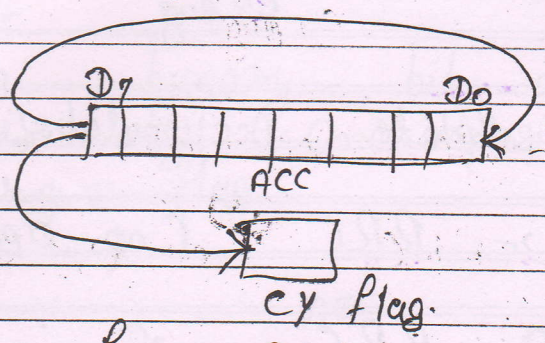


fig. RLC

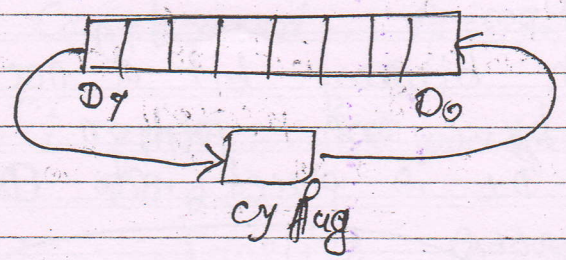
(16) RAL :- Rotate Acc Left through Carry.

⇒ Format :- RAL (no operand)

⇒ e.g. :- RAL

⇒ Addressing Mode :- Implied Addressing Mode

⇒ Desc :- Each binary bit of the Acc is rotated left by one position through the carry flag.



(17) RRC :- Rotate Accumulator Right

⇒ Format :- RRC (no operand)

⇒ e.g :- RRC

⇒ Addressing Mode :- Implied Addressing

⇒

Page :
Date :

⇒ Desc :- Each binary bit of the acc. is rotated right by one position.

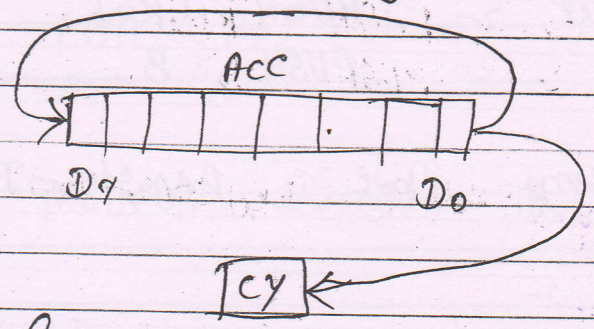


fig. RRC

(16) RAR :- Rotate Acc Right through the Carry

⇒ Format :- RAR No Operand
e.g. :- RAR

⇒ Addressing mode :- Implied Addressing mode

⇒ Desc :-
Each binary bit of the acc is rotated by one position through the carry flag.

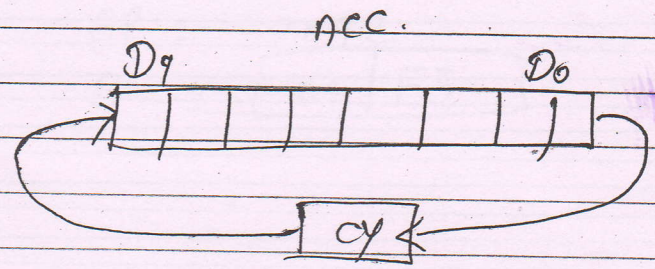


fig RAR

Page :
Date :

(19) PUSH :- Push Register pairs onto Stack.

⇒ Format :- PUSH (Rp)
e.g. :- PUSH B

⇒ Addressing Mode :- Register, Indirect, Combined

⇒ Desc :-
Decrement SP by One SP → (SP-1)
Copy High byte to Loc. pointed by (SP-1)
Decrement (SP-1) by one SP → (SP-2)
Copy Low byte to loc pointed by (SP-2)

→ e.g. before :- SP = 2099 H
BC = 3257 H

After
→ SP = 2099 H
BC = 3257 H

mem

XX	2099H
32	2098H
57	2097H
XX	

Page:
Date:

(20) POP :- Pop off Stack to Register pair.

⇒ Format :- POP (Rp)
e.g. :- POP B

⇒ Addressing mode :- Register - Indirect - Combined

⇒ Desc :-
copy Data from Loc (SP) to low byte Reg. (eg: C)
Increment SP by One SP → SP+1
copy Data from Loc pointed by (SP+1) to High byte
Increment (SP+1) by one SP → SP+2

e.g. before SP = 2090
BC = | XX | XX |
mem

2090H	F5
2091H	01
2092H	XX

after POP B

SP = 2092
BC = | 01 | F5 |

Page:
Date:

(21) CALL :- Unconditional Subroutine CALL

format :- CALL (16 bit Address)

e.g. :- CALL 2050H

Addressing Mode :- Combined Addressing mode

Desc :-
The program sequence is transfer to the address specified by the operand.
Before the transfer the address of next instruction to the call is pushed to the stack.

(22) RET :- Return from Subroutine unconditio

⇒ Format :- RET none
e.g. :- RET

Addressing mode :- Implied Addressing mode

⇒ Desc :-
The program sequency is transferred from the subroutine to the calling function.
Two bytes from the top of the stack is copied to PC & program execution begins.

Page :
Date :

Q.(23) SIM :- Set Interrupt Mask :-

Format :- SIM (No operand)
e.g. SIM
Addressing mode :- Implied :

Desc :- It is multipurpose instruction & used to implement the 8085 interrupts & Serial Data O/P.

→ The instruction interprates Acc content as follows :-

SOD	SDE	XXX	R7.5	MSE	M7.5	M6.5	M5.5

→ SOD :- Serial Output Data
SDE :- Serial Data Enable
↳ if 1, Enables serial output ~~& vice versa~~
XXX :- Dont' care
R7.5 :- Reset RST 7.5
MSE :- mask Set Enable
↳ Global enable for M7.5, M6.5, M5.5
M7.5 :- if 1, RST 7.5 is masked
M 6.5 :- if 1, RST 6.5 is masked
M.5.5 :- if 1, RST 5.5 is marked

Page :
Date :

(24) RIM :- Read Interrupt Mask

⇒ Format :- RIM (No operand)
e.g. :- RIM
Addressing Mode :- Implied.

⇒ Desc :- This is multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 & to read serial data i/p bit.

This instruction loads eight bits in the accumulator as follows:

SID	I7	I6	I5	IE	7.5	6.5	5.5

SID → Serial i/p Data Bit

IE → Interrupt Enable Bit

I7, I6, I5 → Interrupts pending if bit = 1

7.5, 6.5, 5.5 → Interrupts masked if bit = 1

Page :
Date :

(25) EI: Enable Interrupts

⇒ Format :- EI (No operands)
e.g. :- EI
Addressing Mode :- Implied

⇒ Desc :- The Interrupt Enable bit (IE) is set, & all interrupts are enabled. After reset this instruction is necessary.

(26) DI :- Disable interrupts :-

⇒ Format :- DI (No operands)
e.g. : DI
Addressing mode :- Implied

Desc :- The interrupt enable flipflop is reset & all the interrupts except TRAP are disabled.

(27) IN :- Input Data to Acc. from a port with 8-bit address.

⇒ Format :- IN 8 bit port address
eg. :- IN 07H
Addressing Mode :- Direct addressing mode.

→ Desc :- Contents from 8 bit address port are Loaded into Acc.

Page:
Date:

(28) OUT : Output Data from acc to a port with 8-bit address

⇒ Format :- OUT <8 bit port addr>
e.g. OUT 07H
Addressing Mode :- Direct addressing mode

Desc :- The contents of Acc are written to 8 bit address port.

(29) NOP :- No Operation

⇒ Format :- NOP (no-operand)
e.g. : NOP
Addressing mode :- Implied

⇒ Desc :- The instruction is fetched & Decoded However no operation is executed

(30) HLT : Halt & Enter Wait state

⇒ Format :- HLT (no operand)
e.g. :- HLT
Addressing Mode :- Implied

⇒ Desc :- ~~Spec~~ Stop execution from next instruction onwards.

MALP

Page:
Date:

Assignment - II

Q.1 Write a program (WAP) to add two numbers 08H & 02H & store result in A.

⇒ MVI A, 08H
; get the 1st Number in ACC
ADI 02H
; Add 02H to ACC content (to 08H). Result in A.
HLT
; Stop execution.

Q.2 WAP to Subtract Numbers 08H & 02H & Store result in A.

⇒ MVI A, 08H
; get the 1st Number in ACC.
SUI 02H
; Subtract 02H from ACC (from 08H). Result in A.
HLT
; Stop Execution.

Q.3 WAP to add Numbers 18H & 02H & store result in C Register.

⇒ MVI A, 18H
; get 1'st Number in ACC.
ADI 02H
; Add 02H to the ACC content (to 18H). Result in A.
MOV C, A
; move Result from A to C.
HLT
; Stop Execution.

Q.4 WAP to Subtract 02H from 18H & Store Result in C Register.

⇒ MVI A, 18H
; get 1'st Number in ACC.
SUI 02H
; Subtract 02H from ACC content (from 18H). Result in A
MOV C, A
; move Result from A to C.
HLT
; Stop execution.

Page:
Date:

Q.5 WAP to multiply 06H & 07H.

⇒

```
        MVI D, 00H
; Clear D Register for Storing MSB of Result if Any
        MVI A, 00H
; Clear ACC for Storing LSB of Result.
        MVI B, 06H
; Get 1st Number 06H in Reg B.
        MVI C, 07H
; Get 2nd Num 07H in Reg C.
LOOP:   ADD B
; Start of Loop for Repititive Addition. Add B till C exhaust
        JNC NEXT
; if No carry goto Next
        INR D
; if carry → increment MSB Reg D.
NEXT:   DCR C
; Decrement C.
        JNZ LOOP
; Jump to Loop untill C is zero.
        STA 4152H
; Store the LSB Result from ACC to memory
        MOV A, D
; Get the Result MSB from D to ACC.
        STA 4153H
; Store the Result MSB to Memory
        HLT
; Stop execution.
```

Q.6 WAP to Add two 16 bit Numbers using DAD.

⇒ LHLD 4150H
; Load 1st 16 bit Number in HL.
XCHG
; move 1st Number from HL to DE Reg. Pair
LHLD 4152H
; Load 2nd 16 bit Number in HL.
DAD D
; Perform 16 bit addition (HL + DE), Result in HL.
SHLD 4154H
; Store Result in memory from HL to 4154H, 4155H
HLT
; Stop Execution.

Q.7 WAP to add two 16 bit numbers Stored in given memory as below:

2000H → 14H (LSB) 2001H → 6AH (MSB)
3000H → 52H (LSB) 3001H → 4CH (MSB)
4000H → Result (LSB) 4001H → Result (MSB)

⇒ MVI A, 14H
; Get the LSB of 1st number in Acc
STA 2000H
; move it to 2000H memory loc.
MVI A, 6AH
; get the MSB of 1st Num in Acc.
STA 2001H
; move it to 2001H memory loc.

Page:
Date:

```
        MVI  A, 52H
; Get LSB of 2nd Number in ACC.
        STA  3000H
; move it to memory loc 3000H.
        MVI  A, 4CH
; Get MSB. of 2nd Number in ACC
        STA  3001H
; move it to mem loc 3001H
        LHLD 2000H
; Get the 1st Number in HL Reg from mem
        XCHG
; move the 1st Number to DE Reg. Pair.
        LHLD 3000H
; Load the 2nd Number in HL Reg. Pair
        DAD  D
; Do Addition (HL + DE). Result in HL Pair.
        SHLD 4000H
; Store the Result in memory at 4000H & 4001H
        HLT
; Stop execution.
```

2.8 WAP to transfer bytes of source block to Dest block. Size of block is 29 bytes, address of Source block: 5000h & Dest Block: 6000H.

```
        MVI  C, 09H
; Load C Reg with Source block length (09H)
        LXI  H, 5000H
; Load HL with source block starting byte address (5000H)
        LXI  D, 6000H
; Load DE with Dest block starting byte Addr (6000H)
BACK:   MOV  A, M
; get 1st byte of source in to Acc ~~from 500~~
        STAX D
; move the byte to Dest mem pointed by DE
        INX  H
; Increment HL to point to the next source byte
        INX  D
; Increment DE to point to the next Dest loc.
        DCR  C
; Decrement C each time we complet one move
        JNZ  BACK
; Keep on repeating till C=0.
        HLT
; Stop execution.
```

Page:
Date:

Q.9 WAP to transfer memory block of sixteen data bytes stored from starting location 3000H to starting location 4000H in reverse order.

⇒ MVI C, 10H
; Load C Reg with source block length (16 = 10H)
LXI H, 300FH
; load HL with source block Ending byte Address (300F)
LXI D, 4000H
; Load DE with Dest block Starting byte Addr (4000H)
BACK: MOV A, M
; Load the byte (last) from source mem pointed by HL
STAX D
; Store th byte to Dest mem Loc pointed by DE
DCX H
; Decrement HL to Point to previous source byte
INX D
; Increment DE to point to the next Dest loc
DCR C
; Decrement C each time we complete one byte copy
JNZ BACK
; Keep on repeating till C=0
HLT
; Stop execution.

Q.10) WAP to Complement content of mem Location 4000H & Store the result in mem Loc 4001H.

```
LDA 4000H
; Get the Number from mem to ACC
CMA
; Complement the Number in ACC. Result in ACC
STA 4001H
; Store Result to 4001H mem Loc
HLT
; Stop execution.
```

Q.11) WAP to count Number of 1's in given 8 bit Number stored at mem loc C050H Store the result in mem Loc. C051H.

⇒

```
MVI B,00H
; Clear B for using as Temporary Reg for Result
MVI C,08H
; One byte = 8 bit. Use C as bit counter.
LDA 0C050H
; Load Number in Acc from mem loc C050H
BACK: RAR
; get one bit at a time in [CY] using RAR Rotation
JNC SKIP
; if C≠1, skip incrementing & goto [SKIP] Lable
INR B
; Increment B each time we flow a (1) in a Bit
~~SKIP:~~
```

Page :
Date :

```
SKIP :  DCR C
; Decrement bit counter each time a bit is checked
        JNZ BACK
; Repeat till C ≠ 0. Jump back to [BACK] Label
        MOV A,B
; Move the Result from Reg B to ACC.
        STA 0C051H
; Store the Result to mem Location C051H.
        HLT
; Stop Execution.
```

MALP

Page :
Date :

Assignment - III

Q.1 Define addressing modes. Explain addressing modes of 8085 µP with one example for each.

⇒ ~~Instruc~~ Instruction contains opecode & Operands

→ The Operands are data values on which operation is to be performed.

→ The various ways of specifying operands for instructions are called addressing modes

→ There are mainly five different addressing modes.

1) ~~Regis~~ Implied addressing
2) Register addressing
3) Immediate addressing
4) Direct addressing mode
5) Indirect addressing.

(1) Implied Addressing

→ The addressing mode of an instruction is implied by the function of instruction itself.

e.g. STC, CMC, etc.

Page :
Date :

(2) Register addressing :-

→ Here the Operands are specified by registers only.

→ They are always Single byte instruction & are faster to execute.

→ e.g. MOV A, B → Source & Dest both operands are Registers.

(3) Immediate addressing :-

→ Immediate instructions stores the data immediatly after the opecode in the memory

→ e.g. MVI A, 32H → Two byte instruction
e.g. LXI H, 3050H → Three " "

(4) Direct addressing :-

→ Here the address of an operand is directly given within instruction itself.

→ These instructions are always 3-bytes long
→ 1st byte → opecode
2nd byte → Lower byte of Address.
3rd byte → Higher byte of Address.

e.g. LDA 3050H

(5) Indirect addressing :-
Here it specifies the address of the operand indirectly into the register pair.

Eg. mov A, M
→ it copied the contents of the memory location specified by M register pair into the accumulator

Q.2 Explain Looping Counting & indexing with example.

⇒ many applications require that some actions or part of the application

→ The repeatition can be accomplished by Looping. This Looping is same as Looping statement like while in higher level languages.

```
While (condition)
  {
   Block of Statements,
  }
```

→ In the above case, as long as condition is true, block of statement also called body of loop is executed.

→ We can form a loop in 8085 as follows:

```
        MVI  C, 05h.
NEXT    - - -
        - - -
        - - -
        DCR  C
        JNZ  NEXT
```

Body

→ In the above example, count is initialized to 05H. At the end of body of loop, count is Decremented by 1, & if it ~~is over~~ becomes 0, the loop is over otherwise the body is repeated again. Hence Body is repeated 5 times. Here Reg C acts as counter.

→ While writing the programs containing loops, care must be taken to avoid infinite loops.

e.g.

```
Next:   - - -
        - - -
        - - -
        JMP Next
```

The above is infinite loop.

Page:
Date:

→ Apart from using counting in forming loops counters can also be used to count something.

→ Keeping track of position of value in continuous block of data is called indexing.

→ Indexing is especially used in array applications where array or block of values are processed with one value at a time in sequence.

Q.3 Explain T-States, machine Cycle & Instruction Cycle.

⇒ T-State :-
Each operation of the microprocessor is performed in synchronisation with the internal clock.

→ One Clock pulse is called a T-State.

⇒ machine Cycle :-

→ It is defined as time required to perform a memory or an I/O operation.

→ It consists of 3 to 6 T-states

→ $\overline{MEMRD}$, $\overline{MEMWR}$, $\overline{IORD}$, $\overline{IOWR}$ each machine cycle consists of 3 T-states

⇒ Instruction Cycle :-

→ It is defined as time required to executed an instruction.

It consists of One to five machine Cycles.

Q.4 Draw & explain the timing Diagram of MVI C, 15H.

⇒ Structure of the instruction :-

Memory Loc.	Instruction	machine Code
2000H	MVI C, 15H	0E H
2001 H		15 H

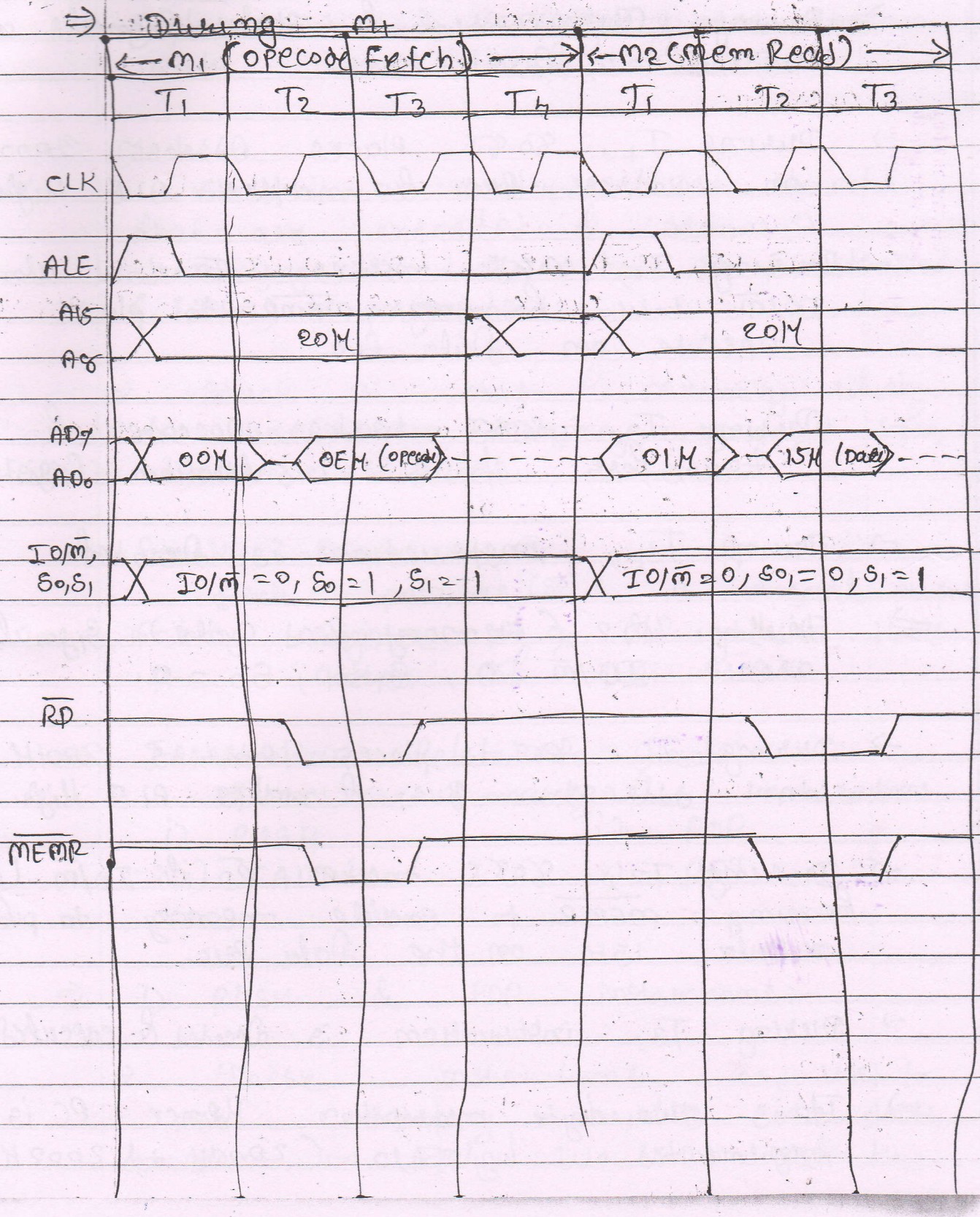
⇒ The Instruction Cycle of consists of two machine Cycles

i) Opcode fetch Cycle (4 T states)

ii) memory Read Cycle. (3 T states)

total 7 T states

⇒ During M_1


← M1 (Opecode Fetch) →
← M2 (Mem Read) →
T1
T2
T3
T4
T1
T2
T3
CLK
ALE
A15
A8
20H
20H
AD7
AD0
00H
0EH (opecode)
01H
15H (Data)
IO/M̄, S0, S1
IO/M̄ = 0, S0 = 1, S1 = 1
IO/M̄ = 0, S0 = 0, S1 = 1
RD̄
MEMR̄

⇒ During M1 Control & status Signals are $IO/\overline{M} = 0$, $S_1 = 1$, $S_0 = 1$.

→ During T_1, 8085 places Address 2000H on address Bus & makes ALE High

→ During T_2, 8085 makes $\overline{RD}$ Low alongh $IO/\overline{M}$ Low, forming $\overline{MEMR}$ to place opecode on Data Bus.

→ During T_3, 8085 reades Opecode & transfers it to instruction Register

→ During T_4, Instruction is Decoded.

⇒ During M2, (memory Read Cycle) Signals are $IO/\overline{M} = 0$, $S_1 = 1$, $S_0 = 0$.

→ During T_1, 8085 places Address 2001H on address Bus & makes ALE High

→ During T_2, 8085 makes $\overline{RD}$ & $IO/\overline{M}$ Low forming $\overline{MEMR}$ to enable memory to put Data 15H on the Data Bus.

→ During T_3, instruction is Decoded & executes.

⇒ It is two byte instruction Hence PC is incremented by Two (2000H to 2002H)

Page:
Date:

Q.5 What is stack & stack pointer? Explain stack related instruction with ~~the~~ help of example.

⇒ Stack :-

→ Stack is a container of objects that are inserted & removed according to the Last In First Out (LIFO) principle

→ Stack is part of memory which is used as tempory storage for local variables.

⇒ Stack Pointer :-

Stack pointer (SP) is a special purpose 16-bit register in a MP, which holds the address of the stack.

⇒ Instruction Related to Stack :-

There are mainly four instructions

i) PUSH ii) POP

iii) CALL iv) RET

⇒ i) PUSH & POP instructions :-

→ These instructions are used to push tempory variables onto stack & when in need we can get the

data back using the POP instruction.

→ Example of PUSH & POP statement usage are

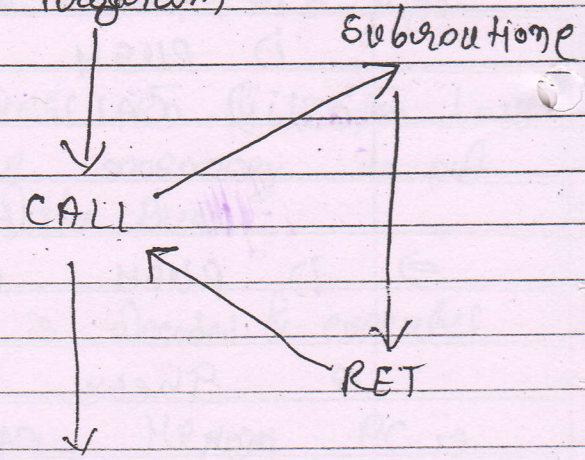
→ PUSH B ; push the contents from location specified by Rp-BC onto top of stack

POP B ; get the data from top of the stack to location specified by Rp-BC.

→ no of PUSH & POP instructions used must be equal.

ii) CALL & RET :

```
CALL 2050H
- - -
- - -
- - -
2050H :  - - -
         - - -
         - - -
         RET
- - -
- - -
- - -
- - -
```



Page:
Date:

→ CALL & RET instructions are used to form a subroutine call & to properly return from it.

→ During CALL
↳ Address specified instruction is loaded in to PC
↳ Address of next instruction is stored on the top of the stack
↳ The control is transfered to PC

→ During RET
↳ Address of next instruction is popped from the stack into PC & mainline program resume

Q.7 Explain PUSH & POP instructions with suitable example.

⇒ Format : PUSH (Rp)

e.g. : PUSH B

Addressing mode : Register - Indirect, combined

Desc :- procedure followed During PUSH Instruction

→ Decrement SP by one SP → (SP-1)

→ Copy High byte to Loc. pointed by (SP-1)

→ Decrement (SP-1) by One SP → (SP-2)

→ Copy Low byte to Loc. pointed by (SP-2)

e.g. PUSH B

before :

SP = 2099H

BC = 3257H

After

SP = 2099H

BC = 3257H

memory

2099H	XX
2098H	32
2097H	57

⇒ POP :-

Format : POP (Rp)

e.g. : POP B

Addressing mode : Register indirect, combined

Page :
Date :

Desc :-

→ Copy Data from loc (SP) to low byte Reg
→ Increment SP by 1. SP → SP+1
→ Copy Data from loc pointed by (SP+1) to High byte
→ Increment (SP+1) by 1 SP → SP+2

e.g. POP B

Before	After
SP = 2090 H	SP = 2092 H
BC = XXXX H	BC = 01F5 H

mem

2090 H	F5
2091 H	01
2092 H	XX

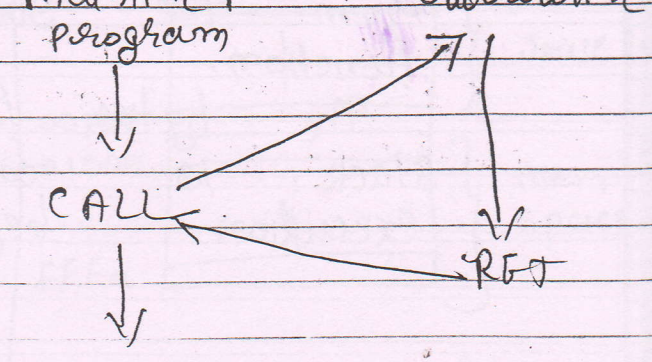
⇒ PUSH & POP instructions are used to Push temp variables (or Data) onto stack & this Data can be popped back when required.

2.8) part

```
CALL   2050H
 - - - -
 - - -

2050H: - - -
 - - -

RET
 - - -
 - - -
```



2.8 Explain CALL & RET instructions with suitable example.

⇒ CALL :- Unconditional Subroutine Call.

Format : CALL (16 bit Address)

e.g. CALL 2050H

Addressing :- Combined Addressing mode

Desc :-

The program sequence is transferred to the address ~~of NEXT~~ specified by the operand.

Before the transfer the address of next instruction to the CALL is pushed to the Stack.

⇒ RET : Return from Subroutine Unconditionally

Format : RET (no operand)

e.g. : RET

Addressing mode : Implied addressing mode

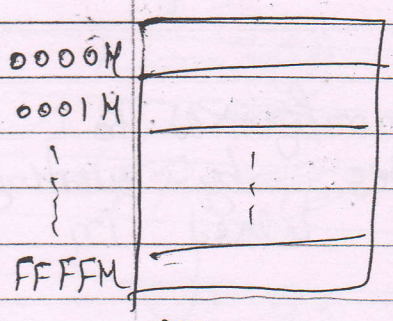
Desc :

The program sequency is transferred from the Subroutine to the calling function.

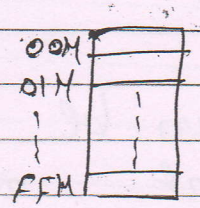
The bytes from the top of the Stack is copied to PC & program execution begins.

Q.9 Explain memory mapped I/O & I/O mapped I/O.

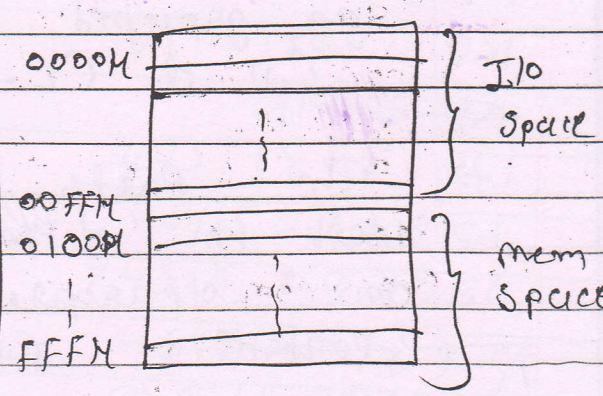
I/O mapped I/O	Memory mapped I/O
→ 16 bit address → memory 8 bit address → I/O	16 bit address → Both (I/O & mem)
→ Address space Not Share mem → 0000H - FFFFH I/O → 00 H - FFH	Address space is shared Both → 0000H - FFFFH
→ max I/O device → 512	max I/O Device → Upto 64KB
→ IO/$\overline{M}$ Signal Required	IO/$\overline{M}$ Not Required
→ Uses Special Instruction e.g IN, OUT	Uses Same Instructions e.g. LDA, STA etc.
→ Four Control Signal Req^d e.g. $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$	Two Control Signal Req^d e.g. $\overline{MEMR}$, $\overline{MEMW}$
→ More Hardware Requirements	Less H/W Requirements



Mem Addr Space



I/O Addr Space



→ In Order to communicate with the peripheral (I/O) ~~for~~ devices, µP employs either peripheral mapped (I/O mapped) or memory mapped I/O technique.

→ Based on it, it is either known ~~to~~ as peripheral (I/O) mapped processor or memory mapped ~~size~~ processor.

2.10) Explain the function of IC 8255 with a block Diagram.

⇒ The 8255A is General purpose I/O Device used ~~for~~ as programmable Peripheral Interface.

⇒ It has 3, 8-bit I/O ports - A, B & C.

→ Port C can be used as individual bits or grouped into two 4-bit parts Cu & CL.

→ The Ports can be configured to work in different modes by writing an appropriate control Word in control Register.

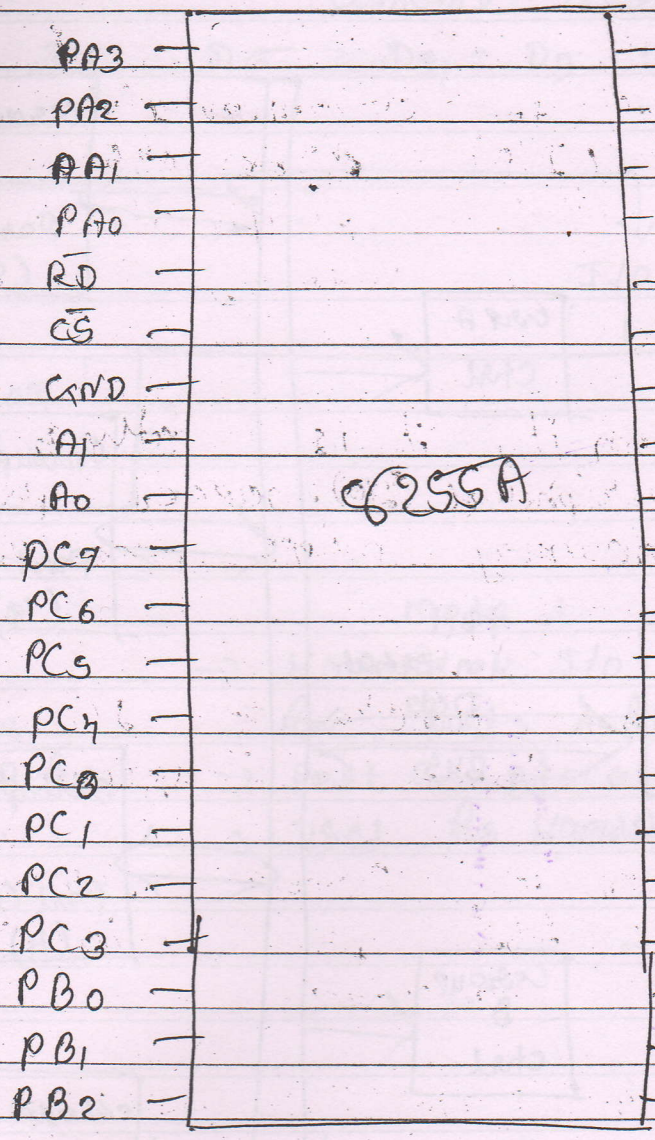


fig. 10.1 Pin Diagram of 8255A

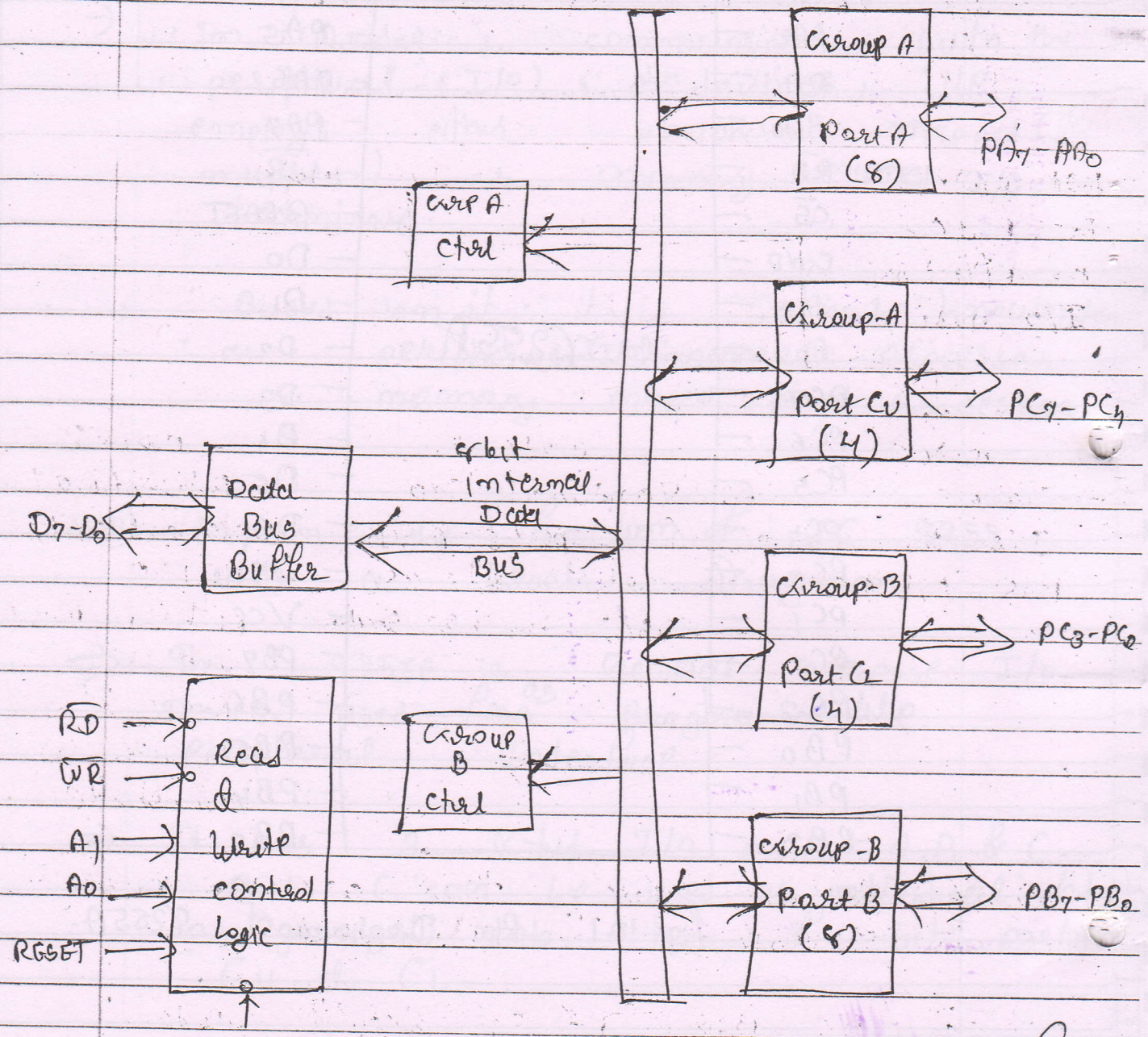


Fig 10.2 Logical Block Diagram of 8255A.

Page :
Date :

Control Word

D7 D6 D5 .. D4 D3 D2 D1 D0

0/1

BSR Mode
For Port C
No effect on
I/O mode

I/O mode

mode 0
→ Simple I/O for Port A, B & C

mode 1
→ Handshak I/O for ports A & B
→ Port C Bits are used for Handshake

mode 2
→ Bidirectional Data bus for PA
→ Port B either in mode 0 or 1
→ Port C bits are used for handshake.

⇒ As shown above there are two modes
i) BSR Mode ii) I/O Mode.

→ i) The BSR (Bit-Set Reset) mode is used to set or Reset the individual Bits of port C.

→ The I/O mode is further divided

into three modes.

→ The mode 0 is Simple I/O mode where all three ports A, B & C can be used as Simple I/O ports.

→ In Mode 1, ports A & B can be configured as handshake mode using bits of port C as handshake Signals.

→ The mode 2 is bidirectional mode for port A, but port B can't be used as bidirectional.