8085 Assembly Language

8-bit Accumulate	8-bit Flag
B	C
D	E
H	L
PC	
SP	

16-bit

PC = Program Counter

SP = Stack Pointer

Flag

S	Z	X	AC	X	P	X	CY

S → Sign Flag

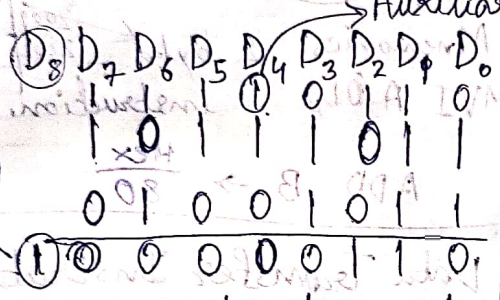
Z → Zero Flag

AC → Auxiliary Carry Flag.

P → Parity Flag.

CY → Carry Flag.

8085 supports both signed & unsigned numbers.



Classification of Instruction Set :

→ Data Transfer Instructions

→ Arithmetic Instructions.

→ Logical Instructions.

→ Branching Instructions.

→ Control Instructions.

Mnemonics	Operand
MOV (Move)	R_d, R_s
	M, R_s
	R_d, M.

M → Memory location

R_d = Register as destination

R_s = Register as Source.

MOV B, C

MOV C, H

MOV B, $(2050)_4$

MVI R_d, data

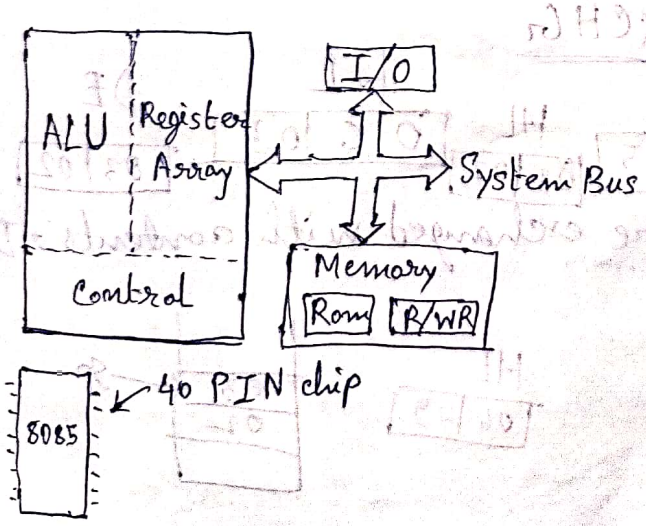
M, data

↳ Move Immediate.

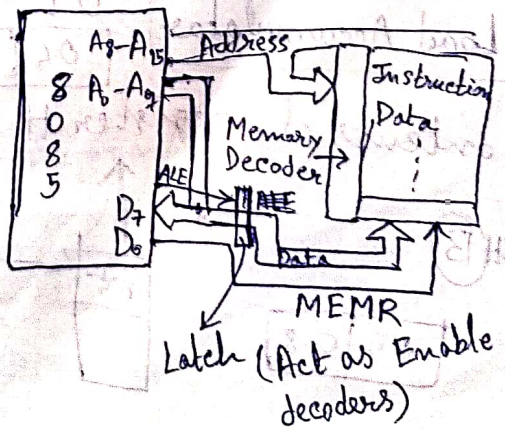
LDA 16-bit Address

Address	Memory
$(2000)_4$	8-bit
$(2001)_4$	8-bit
$(2002)_4$	8-bit
$(2003)_4$	~~06H~~
$(2004)_4$	06H

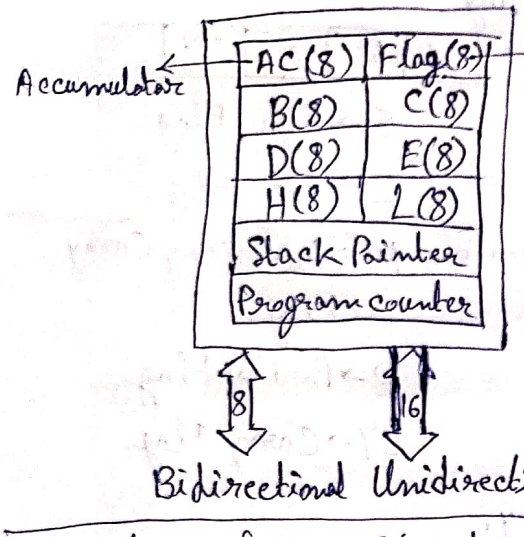
8085 Microprocessor



40 PIN chip



General Purpose Register



Most Commonly Used.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Bits

Assembly language Programming

1) Operation code (Opcode) - Instruction
2) Operand - data.

4F

1 byte instruction

MOV C, A

D

C ← A

$(4F)_H$

Mnemonics

MVI A, 01_H

2 byte instruction.

Hex Codes	
2000	3E
2001	01
2002	

ADD B → Hex 80

Addition of Two Numbers

MVI A, 32_H	3E, 32_H
MVI B, 48_H	06, 48_H
ADD B	80
OUT 01_H	D3, 01_H
HLT/	76/CF
RST1	

MOV A,B	9000	3E
MOV E,M	9001	32
	9002	06
	9003	48
		80
		D3
		01
		76/CF

Data Transfer Instructions

LDAX, B, D register pair

SHLD 2550H

H	L
06	06

06	2550
04	2551

Ex: XCHG

DE: 06 05

HL: 03 07

HL: 06 05

DE: 03 02

i) Contents of register H are exchanged with contents of D.

PUSH B

05	08

08
05

HL: 06 05

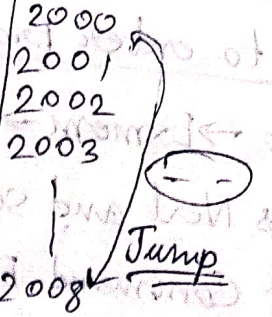
03 ← SP
02

LDA → Load Accumulator

2020	10
2021	20
2022	30
2023	40
2024	50
2025	result

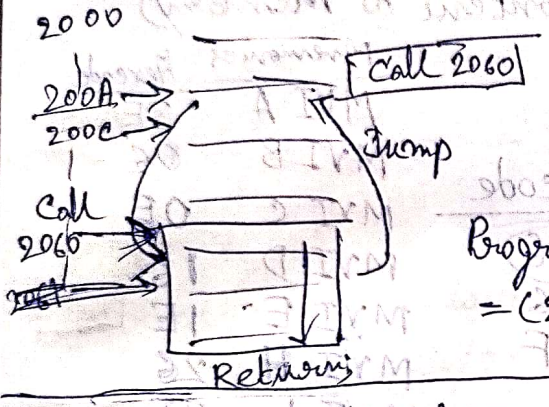
STA 2020H
~~LDA 2020H~~
~~ADD 2021H~~
~~ADD 2022H~~
~~ADD 2023H~~
ADD 2024H

```
LDA  2020H
MOV  B,2021H
ADD  B
MOV  B,2022H
ADD  B
MOV  B,2023H
ADD  B
MOV  B,2024H
ADD  B
STA  2025H.
```



Assembly level Language

Calling a Subroutine



Program Counter = $(200C)_H$

Lab

Follow Gaonkar Book

Q1 ~~Mov~~ MOV data (8 bit) directly to H, L, B, C, D, E register.

How to Handle the up kit?

1) Switch ~~the~~ on the kit using Power supply → SDA 85 display appears.

Reset → Subset Memory (exmem) → 8000/9000 → next.

8000 → next → CF → RST1

Address	Mnemonics	Hex Code
8000	MVI A,06	3E
8001	Data	06
8002	RST 1/RST5	CF/EF

```
MVI  A,06
MOV  H,06
data
RST 5
```

How to run the program?

After entering the last opcode, press next → Reset.

GO → Starting address of your program → ~~exec~~. Fill

How to check the result.

~~Reset~~ Shift → Exam Reg (For checking the content of Register)

→ Subset Memory (exmem) (For checking the content of Memory).

How to enter Program.

Reset → Exmem → Type the starting address (2000/3000)
→ Press Next and so on.
→ Last Command EF,

How to check the Result.

Reset → Shift + Exam Reg (For checking the content of Register)
~~x~~
→ exmem (For checking the content of Memory)

Move Data directly to H, L, B, C, D, E

Address	Mnemonics	Hexcode
2000	MVI H 06	26
2001	Data	06
2002	RST 5 (Variable)	EF

Mnemonics	Hexcode
MVI A	3E
MVI B	06
MVI C	0E
MVI D	16
MVI E	1E
MVI H	26
MVI L	2E
MVI M	36

Q2) Load H-L, B-C, D-E Register pairs, with the data.

[Follow LXI Command]

data :→ $(56FE)_H, (ABCD)_H, (8050)_H$

Address	Mnemonics	Hexcode
2000	LXI H, ~~56FE~~	21
2001	FE (Data)	FE
2002	56 (")	56
~~2003~~	~~RST 5~~	~~EF~~
2003	CD (")	CD
2004	AB (")	AB
2005	50 (")	50
2006	80 (")	80
2007	RST 5	EF

Address	Mnemonics	Hexcode
2000	LXI, H, 56FE	21
2001	FE	FE
2002	56	56
2003	LXI, B, ABCD	01
2004	CD	CD
2005	AB	AB
2006	LXI, D, 8050	11
2007	50	50
2008	80	80
2009	RST 5	EF

Q3) MOV data to the registers B, H, D from the memory locations given below.

3000/8000 – FF

3001/8001 – CD

3002/8002 – 01

Address	Mnemonics	Hexcode
3005	MVI A, FF_H	3E
3006		FF
3007	STA 3000_H	32
3008		00
3009		30
3010	MVI A, CD_H	3E
3011		CD
3012	STA 3001_H	32
3013		01
3014		30
3015	MVI A, 01_H	3E
3016		01
3017	STA 3002_H	32
3018		02
3019		30

Microprocessor Notes

①
```
MVI  A, 00H
MVI  B, F8H
MOV  C, A
MOV  D, B
HLT
```

A	B	C	D	S	Z	CY
00	F8	00	F8	0	1	0

②
```
MVI  A, F2H
MVI  B, 7AH
ADD  B
OUT  PORT1
HLT
```

A	B	S	Z	CY
~~00~~ F2	~~F8~~ 7A	0	0	~~1~~
6C				

F2
+ 7A
16C

③ Example programs

Memory Address	Mnemonics	Value
3020	LXI H, 3000H	21
3021		00
3022		30
3023	MOV B,M	46
3024	LXI H, 3001H	21
3025		01
3026		30
3027	MOV H,M	66
3028	LXI H, 3002H	21
3029		02
3030		30
3031	MOV D,M	56
3032	RST 5	EF

③
```
SUB  A
MOV  B, A
DCR  B
INR  B
SUI  01H
HLT
```

A	B	S	Z	CY
~~00H~~ (-1)H	~~00H~~ ~~(-1)H~~ 00H	1	0	0

Content of accumulator.

④
```
Label   Mnemonic   Operand
        MVI        A, 8FH
        ADI        72H
        JC         DISPLAY
        OUT        PORT1
DISPLAY XRA        A
        OUT        PORT1
        HLT
```

A	B	S	Z	CY
~~8FH~~				1
~~01H~~				
00H				

8FH
+ 72H
101H
CY

JC → If carry flag = 1 then jump to the specified label.

Label	Mnemonic	Operand
(5)	MVI	A, BYTE 1
	ORA	A
	JM	OUTPRT
	OUT	$0F_H$
OUTPRT	HLT	
	CMA	
	ADI	01_H
	OUT	01_H
	HLT	

JM → Jump on minus
CMA → Complement of A
ORA → OR with A.

BYTE1 → any 8 bit data

Label	Mnemonic	Operand
(6)	LXI	H, 2090_H
	SUB	A
	MVI	D, $0F_H$
LOOP:	MOV	M, A
	INX	H
	DCR	D
	JNZ	LOOP
	HLT.	

H | L : 20 | 90

A : 00

D : 0F

2090	00
2091	00
2092	00
2093	00
2094	00
	00
	15 times total

Simply Using a For loop.

A : 00 → 01 ; D : 0F → 0E → 0D.

Z=0 then

JNZ → Jump on Non-zero

(7)

LXI	H, 2070_H	H | L : 20 | 70
MVI	B, 03_H	B : 05 → 04
MVI	A, 01_H	A : 01 → 02

→ Initialization Part.

2070	01
2071	02 → 04
2072	03 → 03
2073	04 → 02
2074	05 → 01
2075	

Label	Mnemonic	Operand
ST:	MOV	M, A
	INR	A
	INX	H
	DCR	B
	JNZ	ST
	HLT	

INX → Increment the content of a register pair.

→ Condition part.

(8) 16 bytes of data are present from memory location 2050_H onwards.

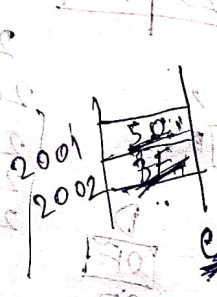
Move all these 16-byte to some other memory location starting from 2070_H

```
LXI  H, 2050H
LXI  B, 2070H
LOOP: MOV M,A
      INX H
      INX B
      DCR C
      JNZ LOOP
      HLT
```

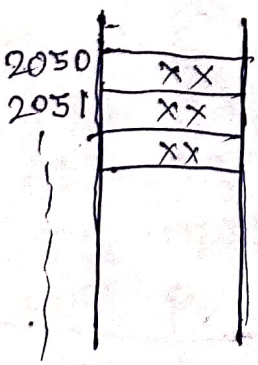
```
LXI  H, 2050H
MVI  D; 10H
LOOP: LDAX B
      STAX M
      INX B
      INX H
      DCR D
      JNZ LOOP
      HLT
```

```
MOV A,M
```

```
LXI  D, 2070H
LXI  H, 2050H
MVI  B, 10H
LOOP: MOV A, M
      STAX D
      INX H
      INX D
      DCR B
      JNZ LOOP
      HLT.
```



10 Bytes of data are present from the memory location 2050_H onwards. Do the sum of those numbers.




```
MVI B, 10H
LXI H, 2070H
LXI D, 2050H
LOOP1: ADD M, A
```

```
  F2 (1)
  78 (2)
 ----
  16A
  FB
 ----
  265
```

One need to handle the carry bit generated each time by another memory.

Repeat the same question for negative numbers.

① A block of data is stored in memory locations from $XX35_H$ to $XX5A_H$. Transfer the data to the locations $XX80_H$ to $XX85_H$ in the reverse order.

② A string of six data bytes is stored starting from memory locations 2050_H. The string includes some blank. Write a program to eliminate those blanks.

③ A set of 8 bytes is stored in the memory locations starting from $XX50_H$. Check each data bytes for bits D_7 and D_0. If D_7 or D_0 is 1, reject databytes; Otherwise store the databytes at memory locations starting at $XX60_H$.

Data:- $80_H, 52_H, E8_H, 78_H, F2_H, 67_H, 35_H, 62_H$

④ A set of 10 bytes is stored in memory. Starting with address $XX50_H$. Write a program to check each bytes and save the bytes that are higher than 60_{10} and smaller than 100_{10} in memory locations starting from $XX60_H$.

Data: $6F_H, 28_H, 5A_H, 49_H, C7_H, 3F_H, 37_H, 3B_H, 78_H, 64_H$

Due date
18th August
2022

LXI H, 2050_H | $3F_H, 37_H, 3B_H$

8000	21
8001	50
8002	20

Addressing Modes

Instruction & Data Formats

1 byte - Instruction
2 bytes - Instruction
3 bytes - Instruction

1-byte Instructions

Mnemonic	Operand	
MOV	CA	$\Rightarrow 4F_H$
ADD	B	$\Rightarrow 80_H$
CMA	--	$\Rightarrow 2F_H$

16-bytes of data are present from the memory location 2050_H onwards.
Move all these 16-byte to some other memory location from 2070_H

2-byte Instructions

MVI A, 05_H
3E — machine code
05 — address

3-byte Instructions

First byte: Mnemonic Equivalent Machine code
Second byte and Third byte → Address

LXI-H 2050H
XX 50 20

Code	Registers
000	B
001	C
010	D
011	E
100	H
101	L
111	A
110	Reserved for memory related operations

MOV C, A → $4F_H$

01 001 111
C A

RAL: $(07)_H$

0000 0111
A

ADD B

80_H

1000 0000

Register Pairs

Code	Registers
00	BC
01	DE
10	HL
11	AF or SP

Data Format

8085 → 8-bit processor.

- ASCII :- 7-bit alphanumeric characters.
- BCD (Binary coded decimal no.s)
  - → Ten digits (0, 9)
  - → 4 bits are used.

Signed Integer

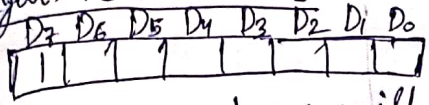
- 8-bit
- Positive No:-

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

0 0 0 0 0 0 0 0 → 00_H

0 1 1 1 1 1 1 1 → $7F_H$

• Negative Numbers



⇒ All -ve numbers will be represented in its 2's complement form.

→ $80_H - FF_H$

Unsigned Numbers

→ $00_H - FF_H$

Addressing Modes:-

→ Direct Addressing Mode [LDA 2050_H] [e.g. JNZ XX_H]

→ Register Addressing Mode [MOV C,A]

→ Register Indirect Addressing Mode [MOV A,M] A ← (M) HL pair contains the address of this data.

→ Immediate Addressing Mode [LXI H, 2050_H; MVI A, 30_H]

→ Implicit Addressing Mode [RAL CMA]

Microprocessor Architecture and its operations

→ ~~Internal Microprocessor~~

→ ~~Peripheral Microprocessor~~

→ Microprocessor Initiated Operations.

→ Internal Operations

→ Peripherals for externally initiated Operations

→ Microprocessor Initiated ~~Operations~~ Operations:-

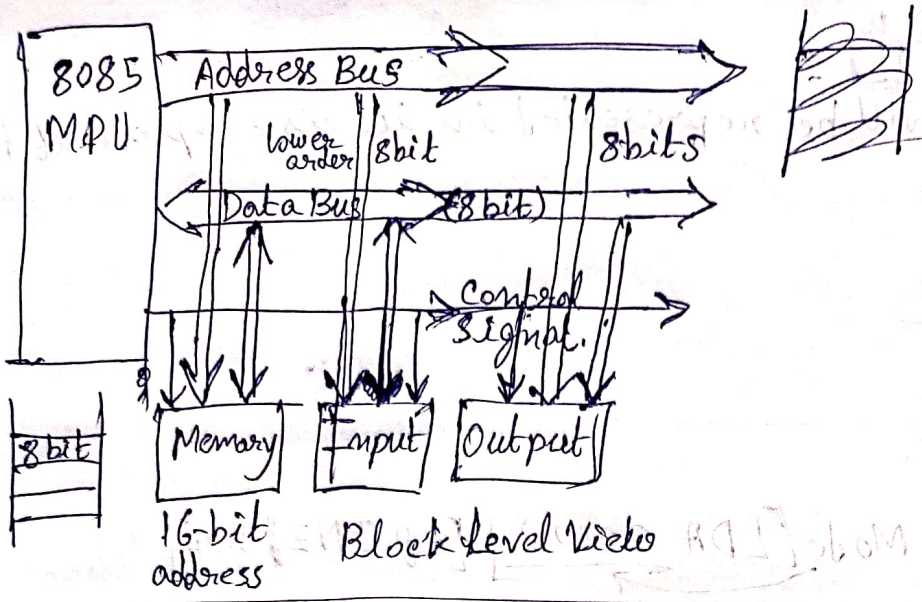
- Memory Read.
- Memory Write.
- I/O Read.
- I/O Write.

⇒ communication Oriented.

⇒ Address bus. (Collection of lines) (Unidirectional) (Store Address in Registers)

⇒ Data bus. (Bidirectional) (Read and Write data)

⇒ Control Signals. (Unidirectional).



→ Internal Data Operations

→ Store 8-bit data
→ Perform arithmatic and logical Operations.
→ Test for conditions.
→ Sequence the execution operation.
→ Temporary store of data after execution of an instruction.

→ Peripheral for Externally Initiated Operations

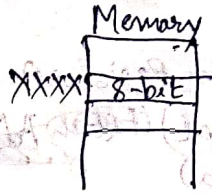
⇒ External Devices can initiate,

* Reset
* Interrupt
* Ready
* Hold

Microprocessor Initiated Operations

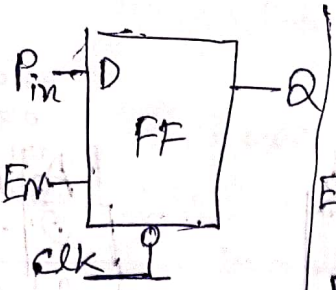
- Memory Read and write
- I/O Read or Write.

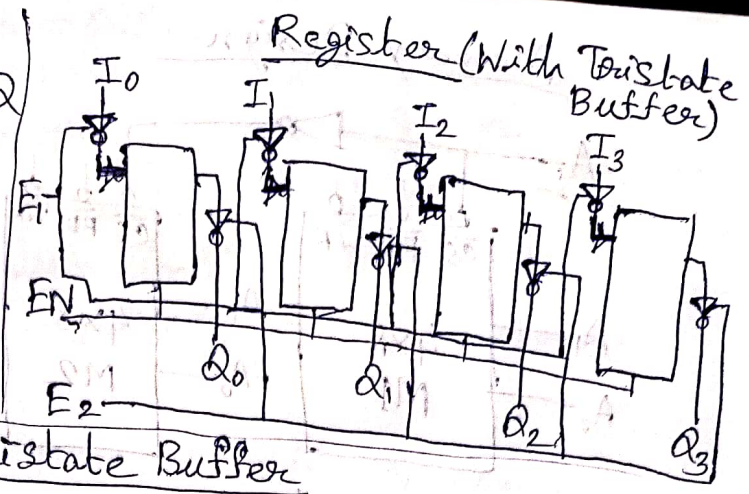
Basic Elements
→ Latch or Flip Flops.

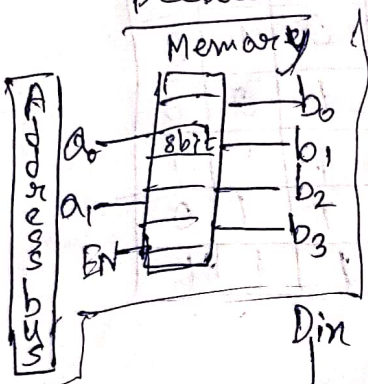


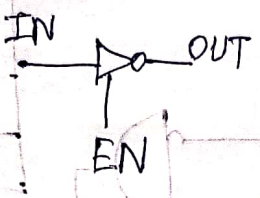
Two types
- R/W memory
- ROM

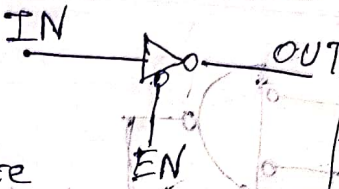
Control signals
→ Memory Read ($\overline{MEMR}$)
→ Memory Write ($\overline{MEMW}$)
→ Chip Select ($\overline{CS}$)



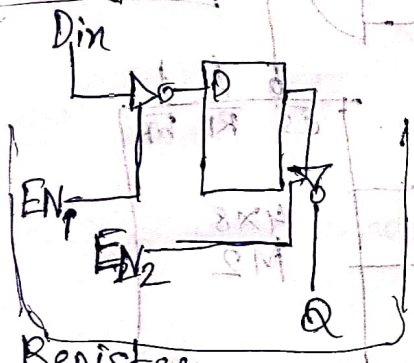




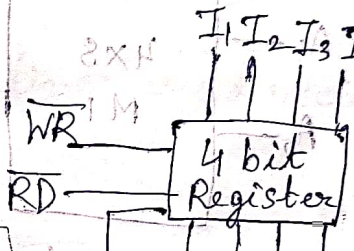




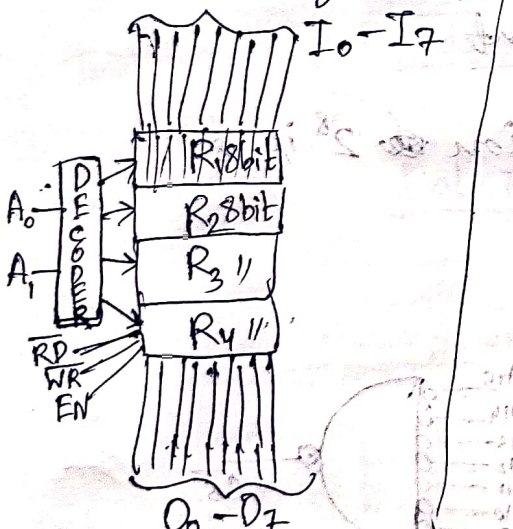
EN	IN	OUT
0	0	Z=0
0	1	Z=0
1	0	1
1	1	0

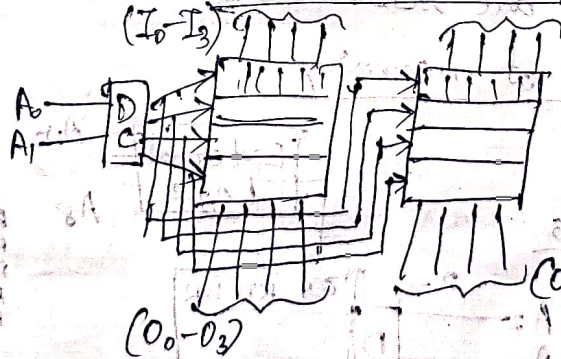


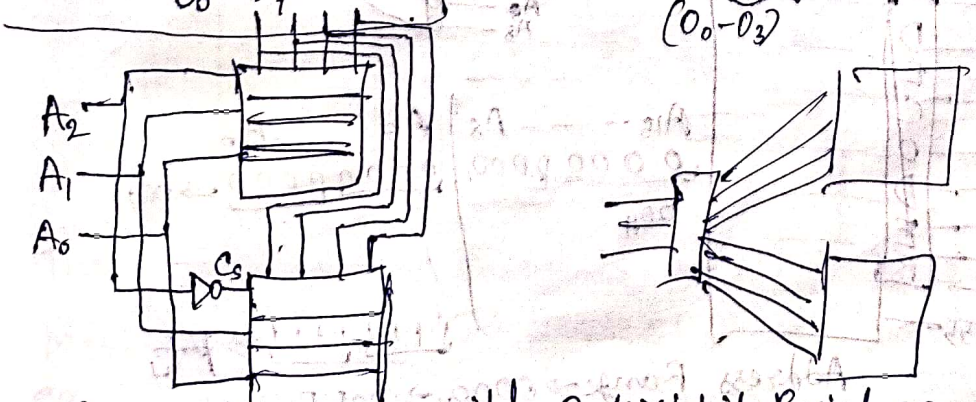
4×8-bit Register



4×8 bit Register using 2 4×4 bit Registers

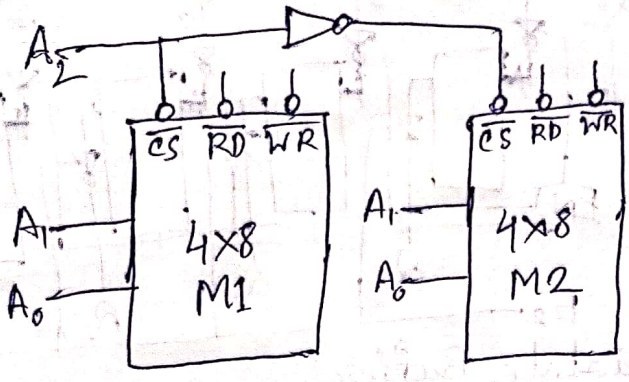






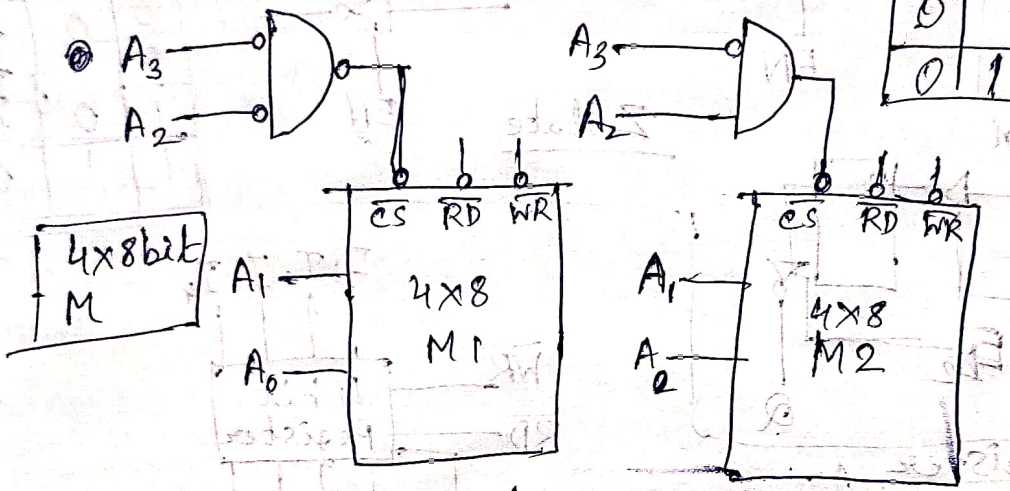
8×4-bit Register with 2, 4×4 bit Registers.

Memory Design



A3	A2	A1	A0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

Design with extra bit $A_3 = 0$,



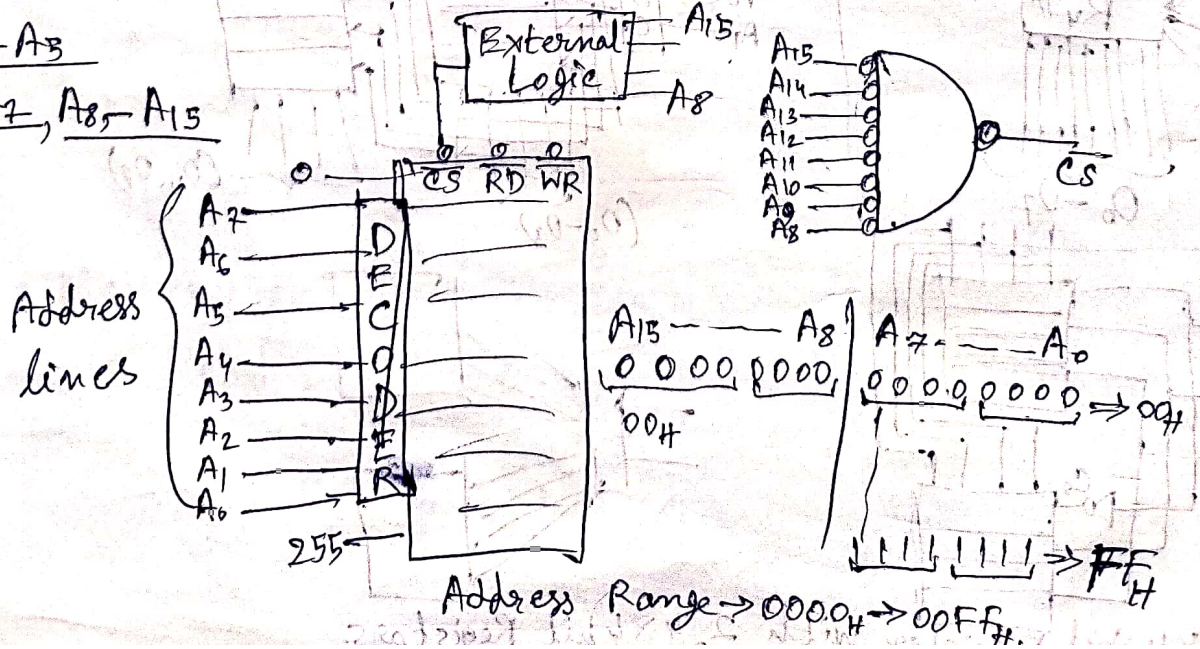
In 8085, 16 Address lines are Present

In order to access 256 byte of memory location 2^8 i.e. 8 address lines are sufficient.

256 byte of memory

$A_0 - A_3$

$A_0 - A_7$, $A_8 - A_{15}$

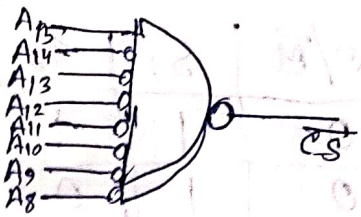


Address Range → $0000_H \rightarrow 00FF_H$.

New Address range, $8000_H \rightarrow 80FF_H$

A_{15} — — A_8
1 000 0000
80_H

For accessing 1'kB address lines (2^{10}) 10 address lines of M is required.



I/O Devices

→ Assign address

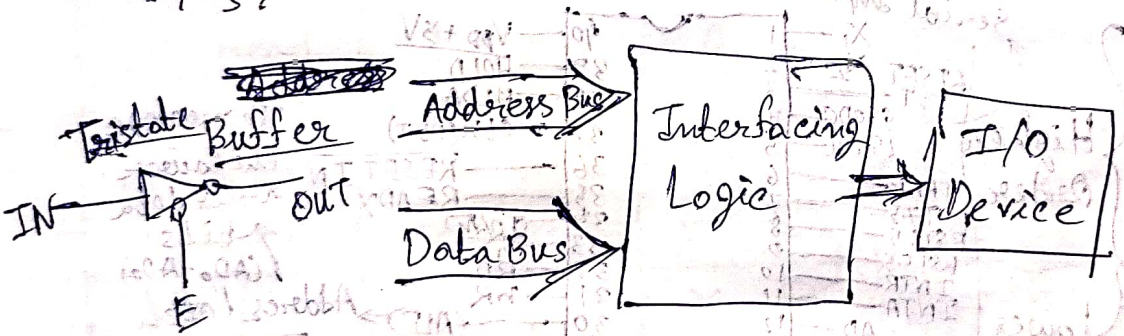
IN 05_H (Range 00_H to FF_H)
OUT 05_H

→ 8-bit address ← port number (to Identify the bit address)

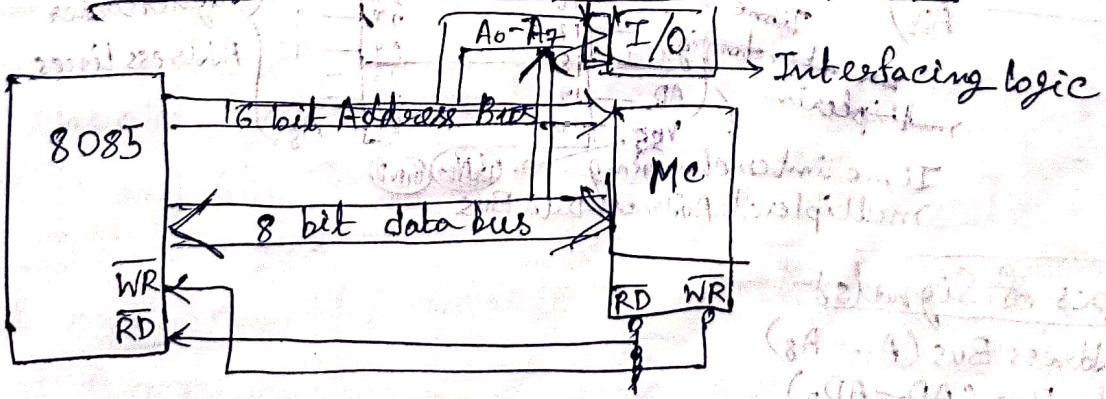
→ 16-bit address → Memory Mapped I/O.

Peripheral Mapped I/O

① Are the address lines and data lines are directly connected to I/Os?



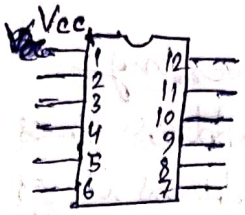
Reading and Writting in Memory Chip



Three signals are there in the Microprocessor to detect Memory chip or I/O Devices with the read write combinations → $IO/\overline{M}$, S_1, S_0

	IO/M	S_1	S_0
Opcode fetch	0	1	1
M RD	0	1	0
M WR	0	0	1
IO RD	1	1	0
IO WR	1	0	1

Tri-State Buffers : 74LS 244



Decoder: 74LS 138

Priority Encoder: 74LS 148

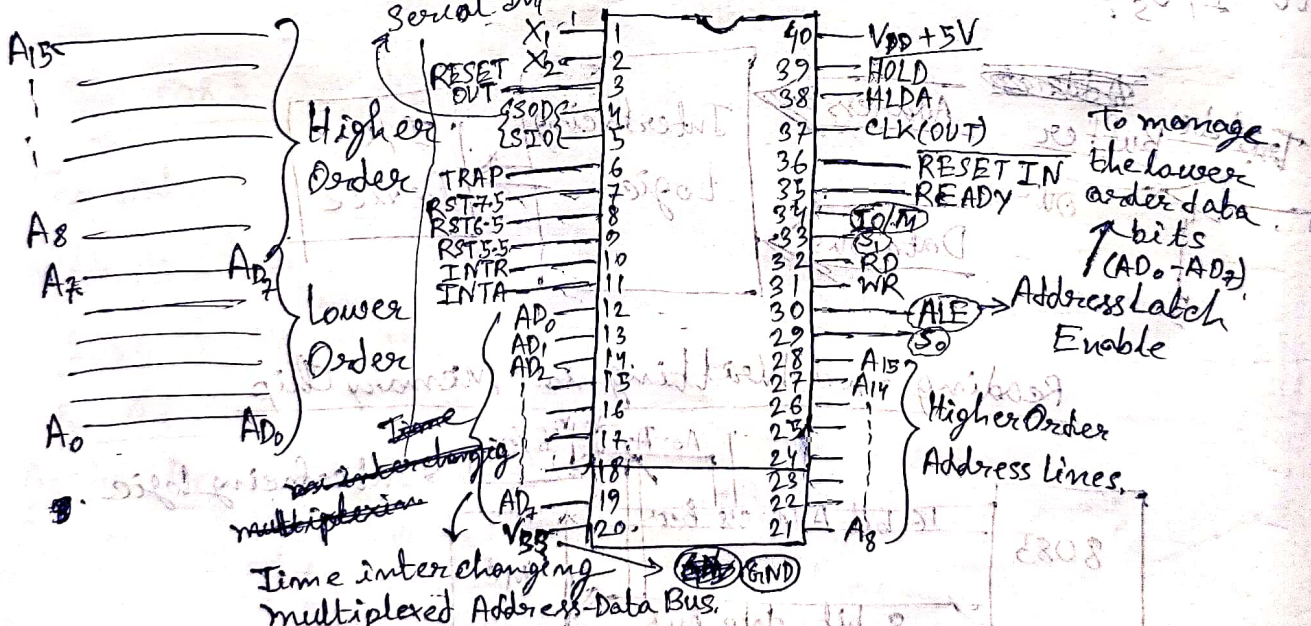
D-Latch: 74LS 373

Microprocessor Architecture

Basic Limitation

- 16-bit ~~bi~~ address
  - ↳ lower order signals are shared with the data bus.

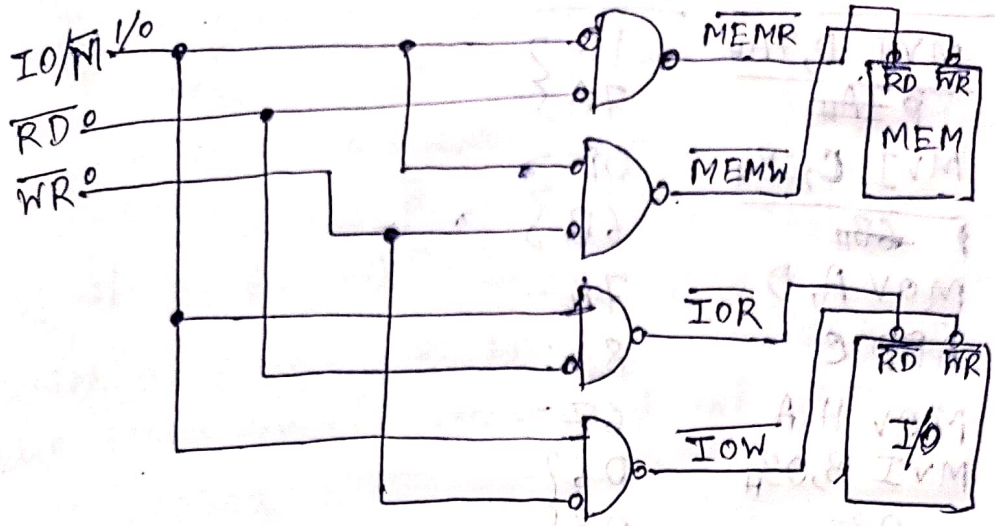
- 8085
  - Address Bus (16-bit)
  - Data Bus (8-bit)



Types of Signals:

- Address Bus (A_{15}-A_8)
- Data Bus (AD_0-AD_7)
- Control and Status Signals (IO/M-S_0)
- Power supply and Frequency Signal. (X_1, X_2, V_{DD}, V_{SS})
- Exactly initiated Signals. (TRAP-INTA)
- Serial I/O Ports. (SOD, SID)

MEMR, MEMW, IOR, IOW



Various Control and Status Signals

Machine Code	IO/M	S_1	S_0	Control Signals
Opcode Fetch	0	1	1	$\overline{RD}=0$
Memory Read	0	1	0	$\overline{RD}=0$
Memory Write	0	0	1	$\overline{WR}=0$
I/O Read	1	1	0	$\overline{RD}=0$
I/O Write	1	0	1	$\overline{WR}=0$
Interrupt Ack	1	1	1	$\overline{INTA}=0$
Reset	Z	X	X	$\overline{RD}, \overline{WR}=Z$
Hold	Z	X	X	$\overline{INTA}=1$
Halt	Z	0	0	

Microprocessor Lab

1) Write the assembly language program to place $7A_H$ in register D and $6B_H$ in register C. Add the above two numbers. Transfer the result of addition to register H. Subtract 03H from the result. Store the final result in register E.

Register
C
D
H
E
A

Output
6BH
7AH
E5H
E2H
E2H

2) 10A
11B
12C
13D
14E

Address	Mnemonics	Hex code
2000	MVI D, 7A$_H$	16
2001		7A
2002	MVI C, 6B$_H$	0E
2003		6B
2004	MOV A, D	7A
2005	ADD C	81
2006	MOV H, A	67
2007	MVI B, 03$_H$	06
2008		03
2009	SUB B	90
200A	MOV E, A	5F
200B	RST 5	EF

2) Write the assembly language program to fill 20 bytes of memory starting from F250$_H$/3050$_H$ with natural numbers in increasing order.

Result:-

Memory location	Content
F250$_H$/3050$_H$	01
F251$_H$/3051$_H$	02

$20_D \rightarrow 14_H$

Address	Mnemonics	Hex Opcode
2000	MVI B, 14$_H$	06
2001		14
2002	MVI A, 01$_H$	3E
2003		01
2004	LXI H, 3050$_H$	21
2005		50
2006		30
2007	MOV M, A	77
2008	INR A	3C
2009	INX H	23
200A	DCR B	05
200B	JNZ, 2007$_H$	

MVI B, 14$_H$
MVI A, 01$_H$
LXI H, 3050$_H$
ST: MOV M, A
INR A
INX H
DCR B
JNZ ST
HLT

Address	Mnemonics	Hex code
2009	JNZ, 2007_H	C2
200A		07
200B		20
~~200E~~	~~HLT~~	~~76~~
200E	RST 5	EF

Q3) Write an Assembly language program to generate a AP series of 8 terms with common Difference 2 and store the series in the memory location starting from $FA00_H / 3500_H$.

Address	Mnemonics	Hex code
3500_H	MVI C, 02_H	$0E_H$
3501_H		02_H
3502_H	MVI B, 08_H	06_H
3503_H		08_H
3504_H	SUB A	97_H
3505_H	LXI H, 3500_H	21_H
3506_H		00_H
3507_H		35_H
3508_H	MOV M, A	77_H
3509_H	ADD C	81_H
$350A_H$	INX H	23_H
$350B_H$	DCR B	05_H
$350C_H$	JNZ, 3508_H	C2
$350D_H$		08
$350E_H$		35
$350F_H$	RST 5	EF

```
    LXI H, 3500H
    SUB A
    MVI C, 02H
    MVI B, 08H
ST→ MOV M, A
    ADD C
    INX H
    DCR B
    JNZ ST
    HLT
```

Q4) Store the Number 24_H in 30 Successive Memory locations

Address	Mnemonics	Hex code
3800_H	MVI A, 24_H	$3E_H$
3801_H		24_H
3802_H	MVI C, $1E_H$	$0E_H$
3803_H		$1E_H$
3804_H	LXI H, 3000_H	21_H
3805_H		00_H
3806_H		30_H

```
    MVI A, 24H
    LXI H, 3000H
    MVI C, 1EH
ST→ MOV M, A
    INX H
    DCR C
    JNZ ST
    HLT
```

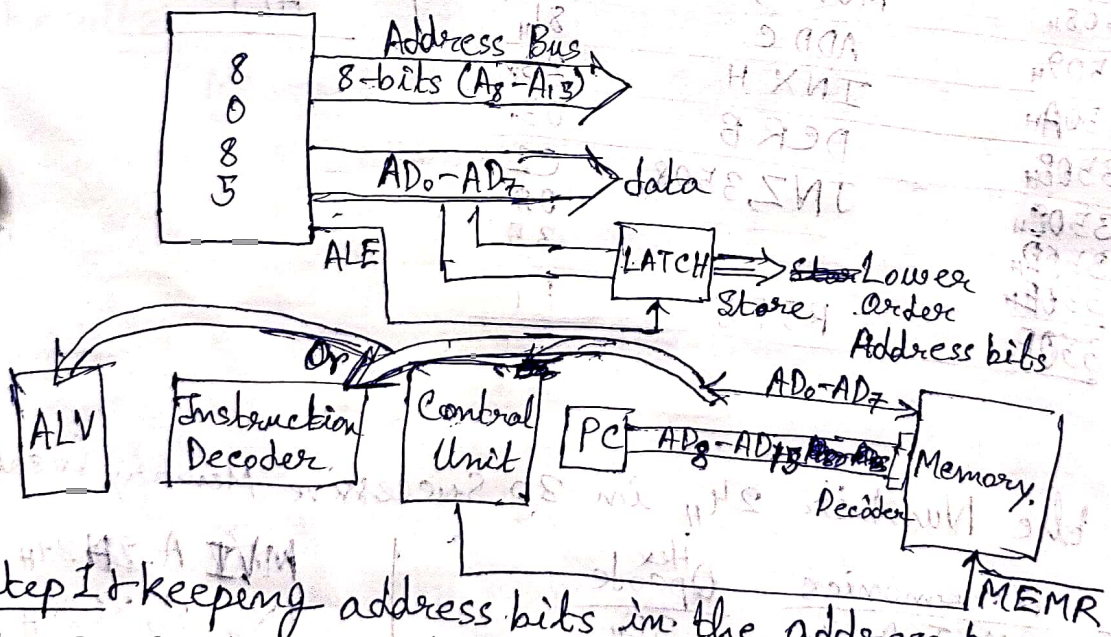
Address	Mnemonics	Hexcode
3807_H	MOV M,A	77_H
3808_H	INX H	23_H
3809_H	DCR C	$0D_H$
$380A_H$	JNZ, 3807_H	$C2_H$
$380B_H$		07_H
$380C_H$		38_H
$380D_H$	RST 5	EF_H

Mircoprocessor Communication And Bus timings

- Bus Timings => At time instants different buses will be used.

16-bit address bus

- lower order $AD_0 - AD_7$ is used in multiplexed manner.

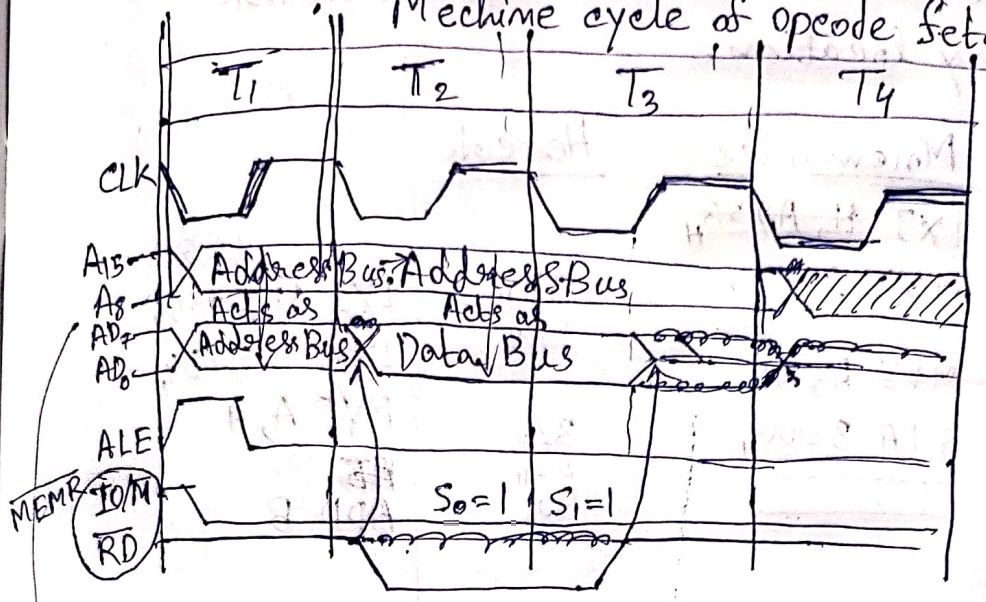


Step 1 - keeping address bits in the address bus.

Step 2 - Sending Control Signal for memory read.

Opcode Fetch Timing Diagram

Mechine cycle of opcode fetch



Clock Memory Cycle

- Opcode fetch
- Memory Read.
- Memory Write.
- I/O Read.
- I/O write.
- ~~Hold~~
- ~~Reset~~
- Interrupt ~~Acknowledge~~ Acknowledge.
- Halt
- Hold
- Reset.

If freq of clock, $f = 3\,MHz$

$\therefore T = 1/f$

$\therefore$ Total $T = 1/f \times 10 \Rightarrow$ LXI H, 2050_H

LXI H_2 2050_H

Opcode fetch ← XX → 4-T states
Memory Read ← 50 → 3-T states
20 → 3-T states.

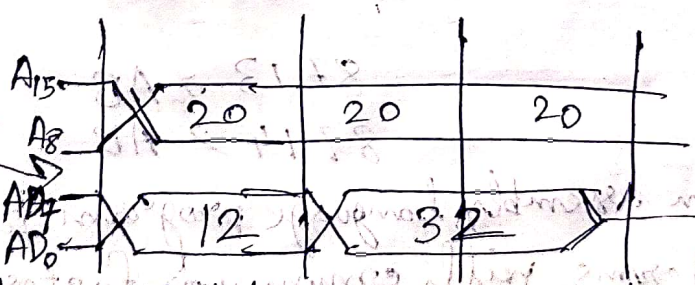
T → clock cycles passed.

→ Total 10-T States.

SUB A = XOR A = Makes Accumulator content 0,

MVI A, 32_H

2011 3E
2012 32



Total 7T states.

For Opcode fetch → 4 T states, Memory Read → 3 T states.

~~STA~~ STA, 2065_H → ~~Done by~~ Decesions are taken by Instruction Decoder.

4T ← XX → Opcode fetch
3T ← 65 → Memory Read
3T ← 20 → Memory Read
3T ← → Memory Write.

} 4 Memory Cycles

Total 13T states.

1) Store two 16 bit numbers in different memory locations. Add these two 16 bit numbers. Store the result in a memory location.

Address	Mnemonics	Hexcode
~~2000~~	~~LXI H, A95B$_H$~~	
2000	MVI A, 5B$_H$	3E$_H$
2001		5B$_H$
~~2002~~	~~MVI A, A9~~	
2002	STA 3000$_H$	32$_H$
2003		00$_H$
2004		30$_H$

~~205B~~

LXI ~~H, A95B$_H$~~

LXI ~~B, 7D4$_H$~~

MVI A, H

~~RST~~

ADD B

LXI Reset

Subset memory: 8711 → CP, 8712 → CD — CDCD

8713 → AB, 8714 → AB — ABAB

Q2) Write an assembly language program to generate a GP series of 8 terms with common factor 2 and store the series in the memory location starting from FB00$_H$

20)

Address	Mnemonics	Hexcode
2011	LDA, 2050_H	
2012		
2013		
2014	MOV B, A	
2015	LDA, 2052_H	
2016		
2017		
2018	ADD B	
2019	STA, 3050_H	
202A		
202B		
202C	LDA 2051_H	
202D		
202E		
202F	MOV B, A	
2030	LDA, 2053	
2031		
2032		
2033	ADC B	
2034	STA 3051	
2035		
2036		
2037	RST 5	

JNC 2037
MVI A, 01
STA 3052
RST 5
~~SUB A~~
~~ADC A~~
~~STA 3052~~

2)

```
LXI  H, FB00H
MVI  A, 01H
MVI  B, 08H
MOV  M, A
INX  H
ADD  A
DCR  B
JNZ  ST
HLT
```

~~SUB A~~
~~ADC A~~

2

Add	Data
FB00	01
FB01	02
02	04
03	08
07	

Subset memory (exmem) ⇒ 2050 → 01, 5B
2051 → 00, A9
2052 → 02, D4
2053 → 00, F7

A9 5B
F7D4
A12F

Address	Mnemonics	Hexcode	
2000	LHLD 2050	2A	→ MVI C, 00
2001		50	
2002		20	
2003	XCHG	EB	
2004	LHLD 2052	2A	00, F7, A1, 2F
2005		52	
2006		20	
2007	DAD D	19	→ INC 2008 INR C
2008	SHLD 3050	22	
2009		50	
200A		30	
200B	SUB A	97	→ MOV A, C
200C	ADC A	8F	STA 3052
200D	STA 3052	32	RET,
200E		52	
200F		30	
2010	RST 5	EF	

2) LXI H, FB00H
MVI A, 01H
MVI B, 08H
MVI C, 02H
ST → MOV M, A
INX H
MOV D, A
MVI E,
MOV E, C
ST2 → ADD D
JNZ ST2
DCR E
JNZ ST2
DCR B
JNZ ST
HLT

A = 01 = a
B = 08 = n
C = 02 = r
C = 1

FB00	1	D = 2, E = 2 − 0
FB01	2	A = 2 + 2 = 4
FB02		07

A = 4 + 2 = 6

Q1) Method I: Using 8 Bit Instructions:

Address	Mnemonics	Hexcode	Comment
2050	A958	58	$2050_H \leftarrow A958_H$
2051	———	A9	
2052	F7D4	D4	$2052_H \leftarrow F7D4_H$
2053	———	F7	
2000	MVI C, 00_H	0E	$C \leftarrow 00_H$
2001	———	00	
2002	LDA, 2050_H	3A	$A \leftarrow 2050_H$
2003	———	50	
2004	———	20	
2005	MOV B, A	47	$B \leftarrow A$
2006	LDA 2052_H	3A	$A \leftarrow 2052_H$
2007	———	52	
2008	———	20	
2009	ADD B	80	$A \leftarrow A+B$
200A	STA 3050_H	32	$3050_H \leftarrow A$
200B	———	50	
200C	———	30	
200D	LDA 2051_H	3A	$A \leftarrow 2051_H$
200E	———	51	
200F	———	20	
2010	MOV B, A	47	$B \leftarrow A$
2011	LDA 2053_H	3A	$A \leftarrow 2053_H$
2012	———	53	
2013	———	20	
2014	ADC B	88	$A \leftarrow A+B+CY$
2015	STA 3051_H	32	$3051_H \leftarrow A$
2016	———	51	
2017	———	30	
2018	INC $201C_H$	D2	if (!CY)
2019		1C	Jump to $201C_H$
201A		20	
201B	INR C	0C	$C \leftarrow C+01_H$
201C	MOV A, C	79	$A \leftarrow C$
201D	STA 3052_H	32	$3052_H \leftarrow A$
201E	———	52	
201F	———	30	
2020	RST 5	EF	Reset

Result ⇒

2050	58
2051	A9
2052	D4
2053	F7

3050	2F
3051	A1
3052	01

$$\begin{array}{r} A958_H \\ +F7D4_H \\ \hline 1A12F_H \end{array}$$

Q1> Method 2:- Using 16 Bit Instructions:-

Address	Mnemonics	Hexcode	Comments
2050	A958	5B	$2050_H \leftarrow A95B_H$
2051		A9	
2052	F7D4	D4	$2052_H \leftarrow F7D4_H$
2053		F7	
2000	MVI C, 00_H	0E	$C \leftarrow 00_H$
2001		00	
2002	LHLD 2050_H	2A	$H\text{-}L \leftarrow 2050_H$
2003		50	($H \leftarrow 2051_H$ & $L \leftarrow 2050_H$)
2004		20	
2005	XCHG	EB	$D \leftrightarrow H$ & $E \leftrightarrow L$
2006	LHLD 2052_H	2A	$H\text{-}L \leftarrow 2052_H$
2007		52	
2008		20	($H \leftarrow 2053_H$ & $L \leftarrow 2052_H$)
2009	DAD D	19	$H \leftarrow H+D$ & $L \leftarrow L+E$
200A	SHLD 3050_H	22	$3050 \leftarrow L$ &
200B		50	
200C		30	$3051 \leftarrow H$
200D	JNC 2011_H	D2	IF(!CY)
200E		11	
200F		20	Jump on 2011_H
2010	INR C	0C	$C \leftarrow C+01_H$
2011	MOV A, C	79	$A \leftarrow C$
2012	STA 3052_H	32	$3052_H \leftarrow A$
2013		52	
2014		30	
2015	RST 5	EF	Reset

Results:

2050	5B
2051	A9
2052	D4
2053	F7

3050	2F
3051	A1
3052	01

```
  1 1
 A95B
+F7D4
1A12F
```

Q2)

Address	Mnemonics	Hexcode	Comments
4000	LXI H, FB00$_H$	21	H-L ← FB00$_H$
4001	———	00	(H ← 00$_H$ & L ← FB$_H$)
4002	———	FB	
4003	MVI A, 01$_H$	3E	A ← 01$_H$ (a)
4004	———	01	
4005	MVI B, 08$_H$	06	B ← 08$_H$ (n)
4006	———	08	
4007	MVI C, 02$_H$	0E	C ← 02$_H$ (r)
4008	———	02	
4009	MOV M, A	77	M ← A
400A	INX H	23	H-L ← HL+01$_H$
400B	MVI D, 00$_H$	16	
400C	———	00	
400D	MOV E, C	59	
400B	MOV D, A	57	D ← A
400C	SUB A	97	A ← A-A = 0
400D	MOV E, C	59	E ← C
400E	ADD D	82	A ← A+D
400F	DCR E	1D	E ← E-01$_H$
4010	JNZ 400E$_H$	C2	If(!Z)
4011	———	0E	Jump on 400E$_H$
4012	———	40	
4013	DCR B	05	B ← B-01$_H$
4015	JNZ 4009$_H$	C2	If(!Z)
4016	———	09	Jump on 4009$_H$
4017	———	40	
4018	RST 5	EF	Reset

Result: