2020 Fall

Qno. 3(a)

$AC \leftarrow DR - 1$

$AC \leftarrow DR' + 1 \times = -x$

=) RTL Code:

FETCH1 : $AR \leftarrow PC$

FETCH2 : $DR \leftarrow M$, $PC \leftarrow PC + 1$

FETCH3 : $IR \leftarrow DR[7,6]$, $AR \leftarrow DR[5 \ldots 0]$

COM : $AC \leftarrow AC'$

JREL1 : $DR \leftarrow M$

JREL2 : $PC \leftarrow PC + DR$

OR1 : $DR \leftarrow M$

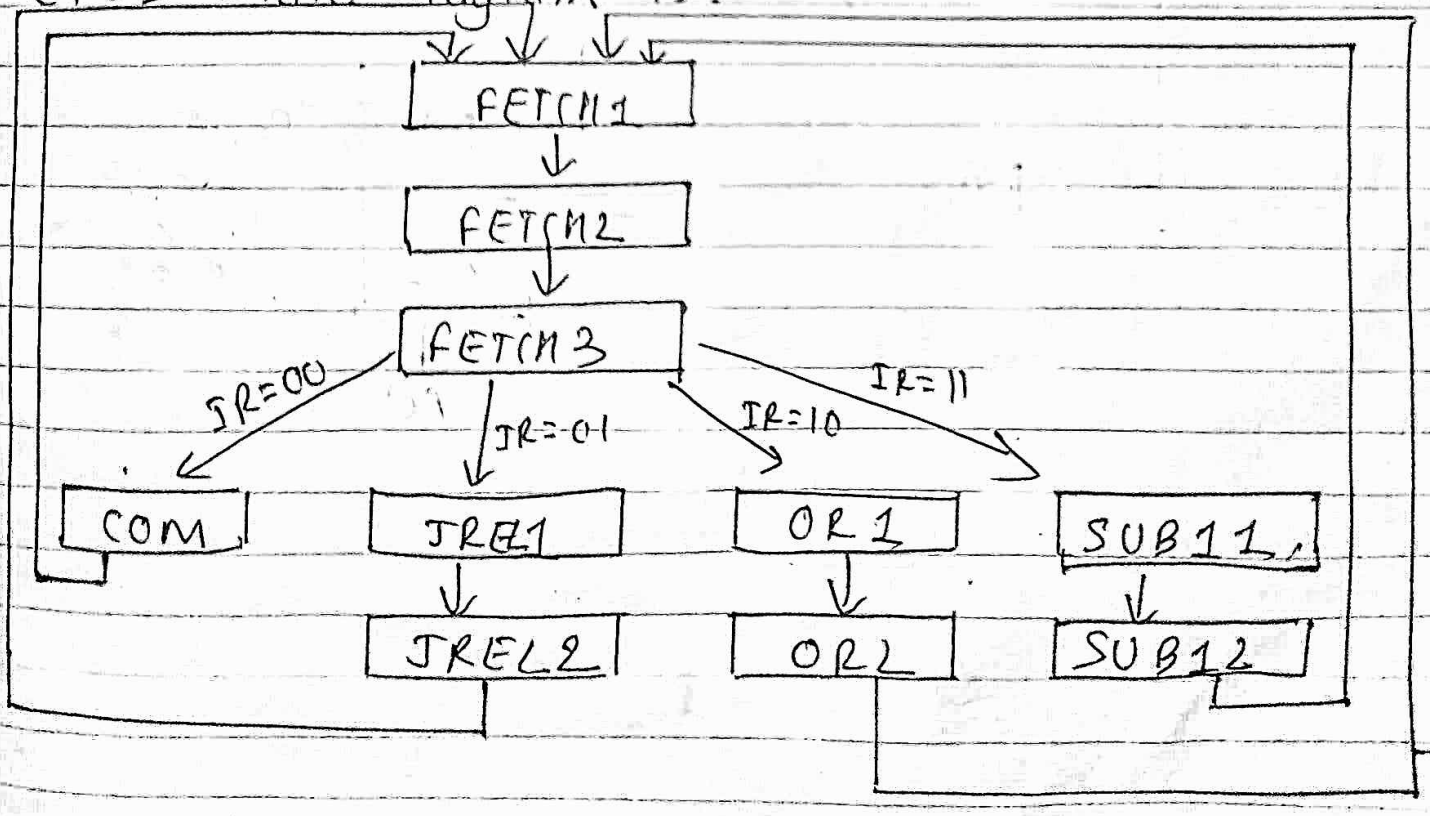
OR2 : $AC \leftarrow AC \vee DR$

SUB11 : $DR \leftarrow M$

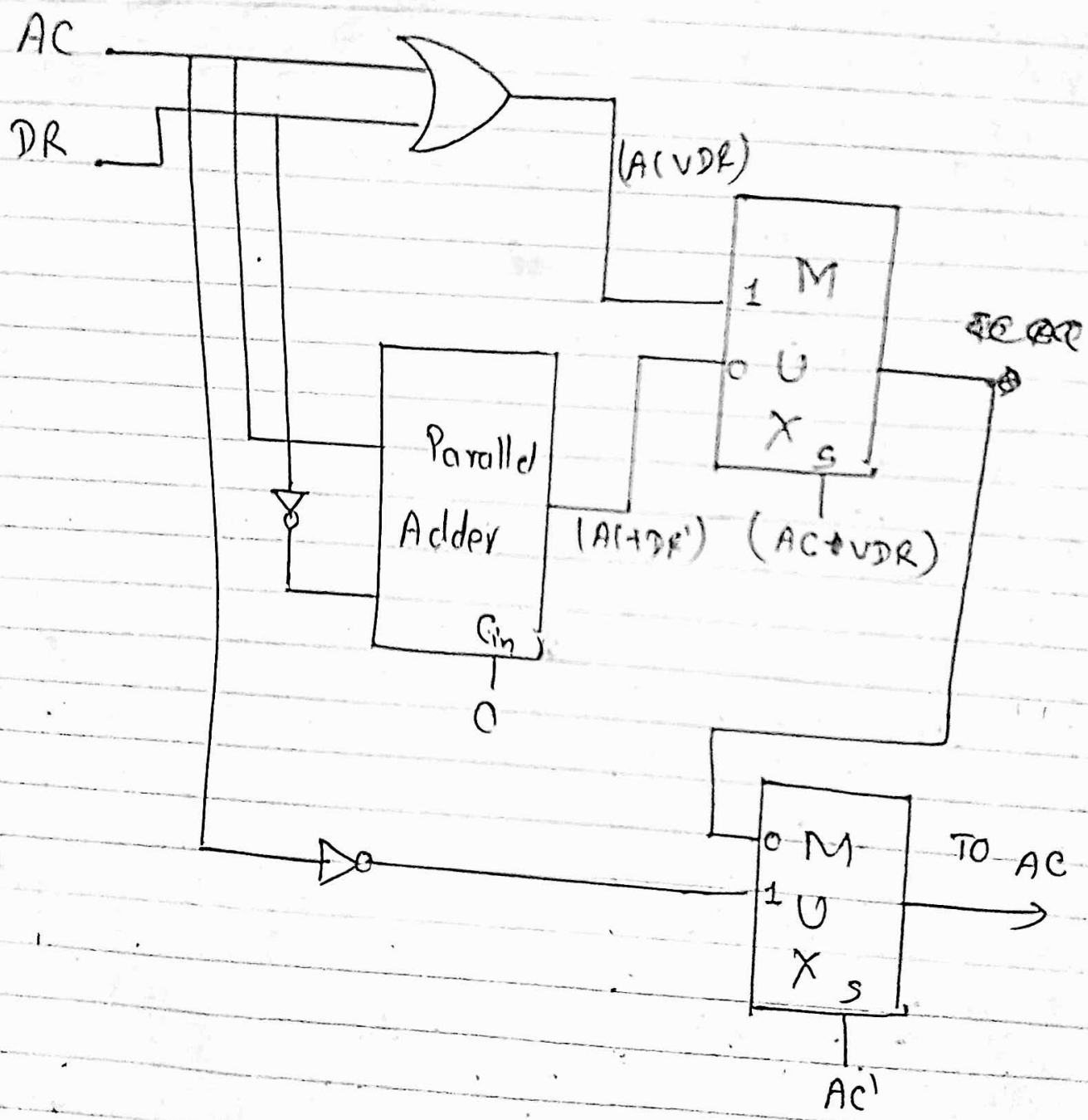
SUB12 : $AC \leftarrow AC + DR'$

Now,

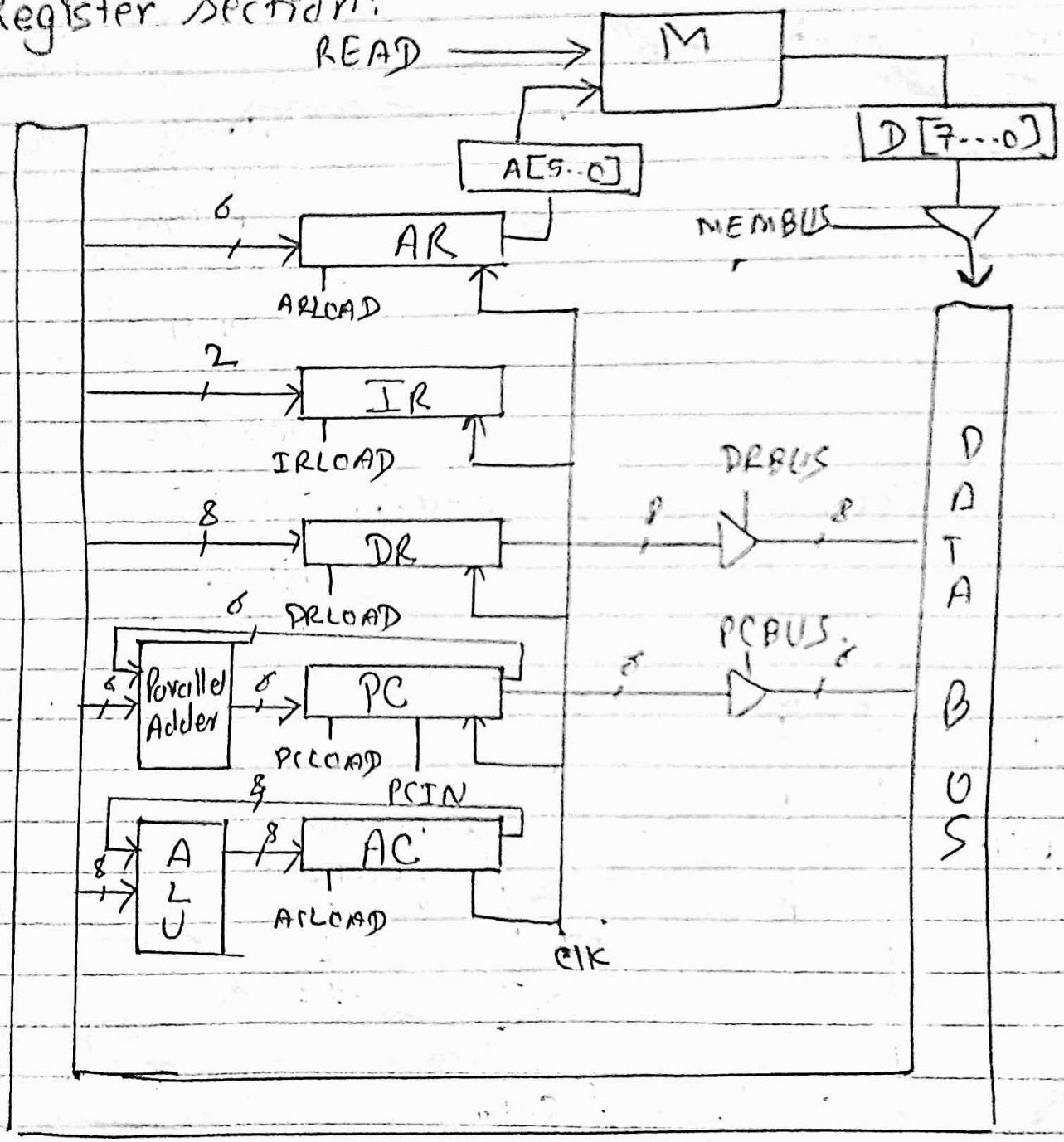
CPU's state diagram is:



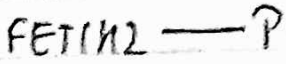
Now, ALU:

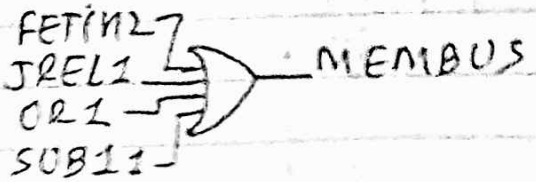


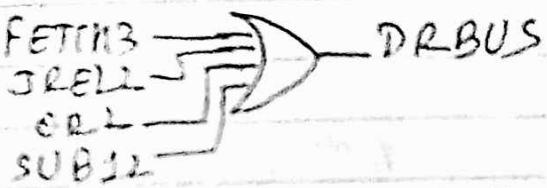
Register Section:

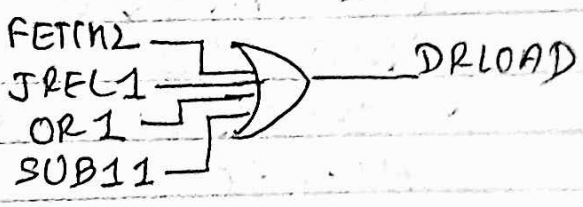


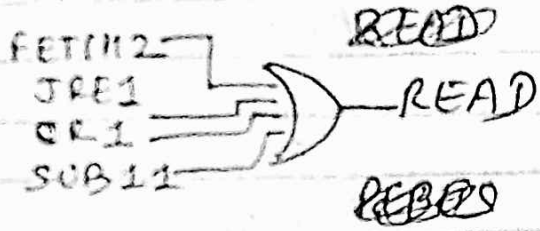
Now, control signals:

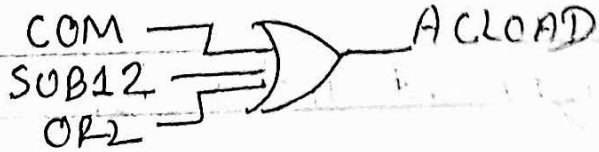


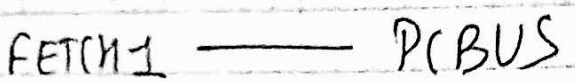












Ans. 4@

⇒

Now, microsequencer control unit with vertical microcode.

Firstly.

Mnemonic.	Micro-operation.
ARPC	AR ← PC
DRM	DR ← M
PCIN	PC ← PC + 1
[illegible]	[illegible]
[illegible]	[illegible]
AIDR	IR ← DR[7,0], AR ← DR[5:0]
COM	AC ← AC'
JREL	PC ← PC + DR
OR	AC ← AC ∨ DR
SUB.	AC ← AC + DR'

Now, micro-operation field assignment and value:

M1

Value	Micro-operation
000	Nop
001	ARPC
010	DRM
011	AIDR
100	COM
101	JREL
110	OR
111	SUB

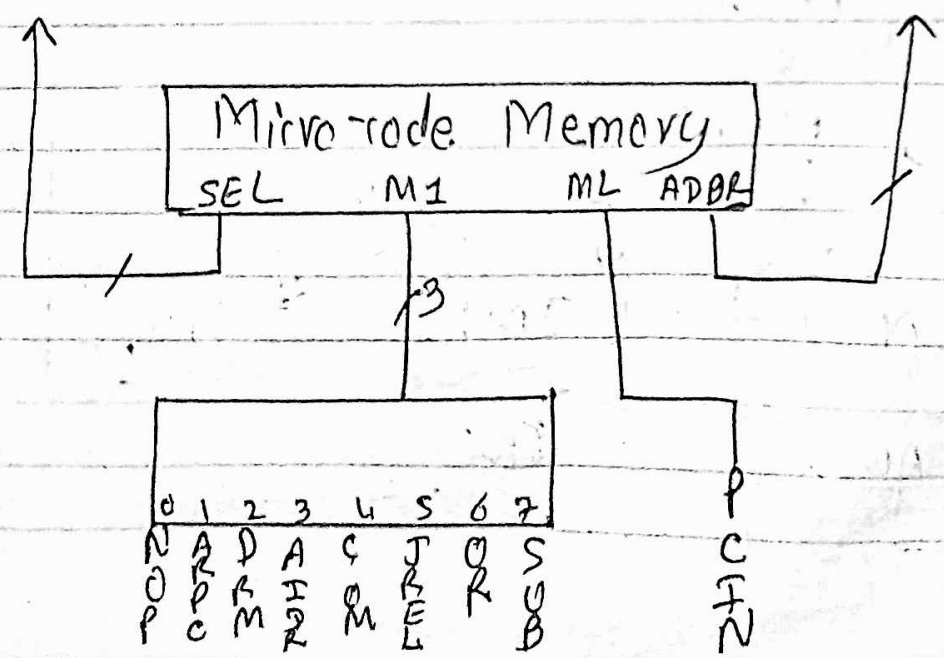
M2

Value	Micro-operation
0	NOP
1	PCIN

Now,
The microsequencer with vertical microcode.

State	Address	SEL	M1	M2	ADDR
FETCH1	0000 (0)	0	0001	0	0001
FETCH2	0001 (1)	0	0010	1	0010
FETCH3	0010 (2)	1	0011	0	XXXX
COM	1000 (8)	0	1000	0	0000
JREL1	1010 (10)	0	0010	0	1011
JREL2	1011 (11)	0	1010	0	0000
OR1	1100 (12)	0	0010	0	1101
OR2	1101 (13)	0	1110	0	0000
SUB11	1110 (14)	0	0010	0	1111
SUB12	1111 (15)	0	1111	0	0000

Now, Showing the micro-operations generation.



Q no. 19 Fall

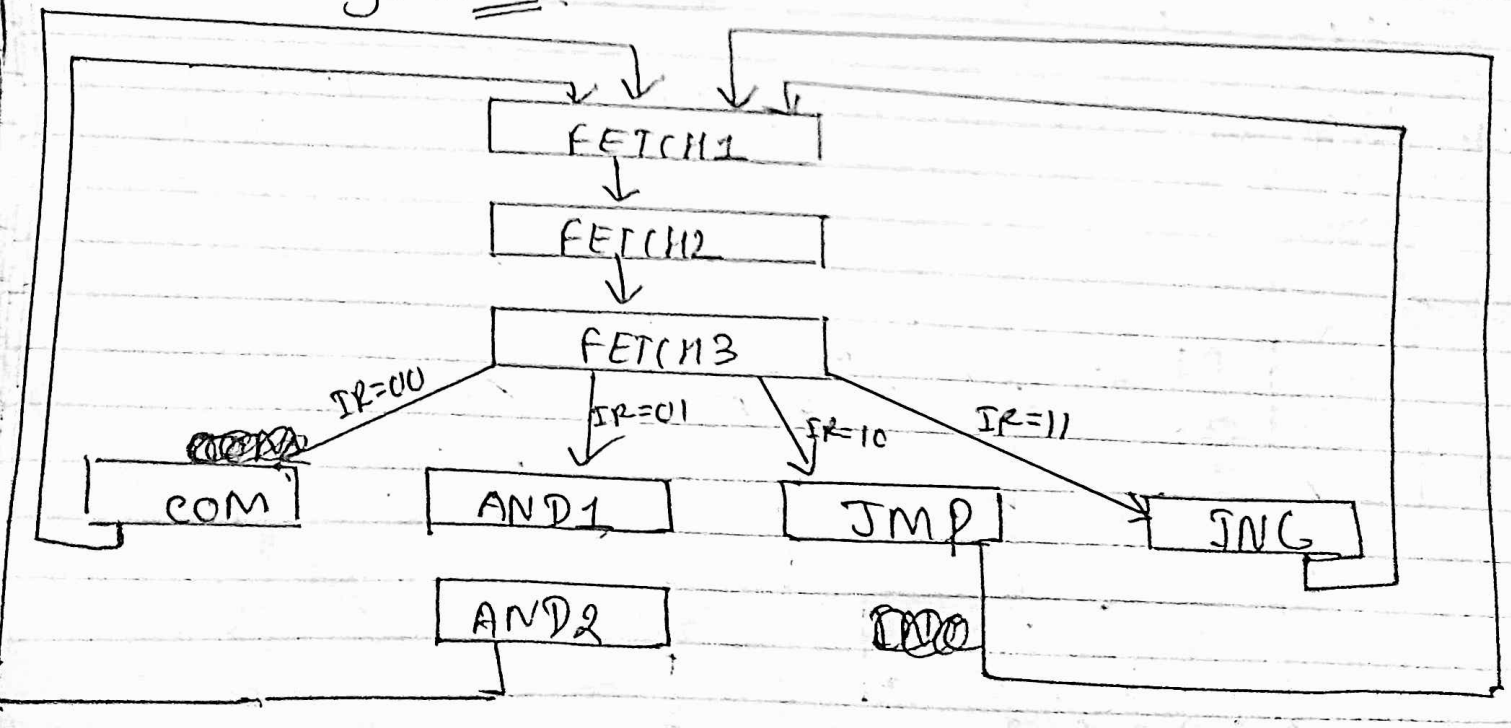
(D)

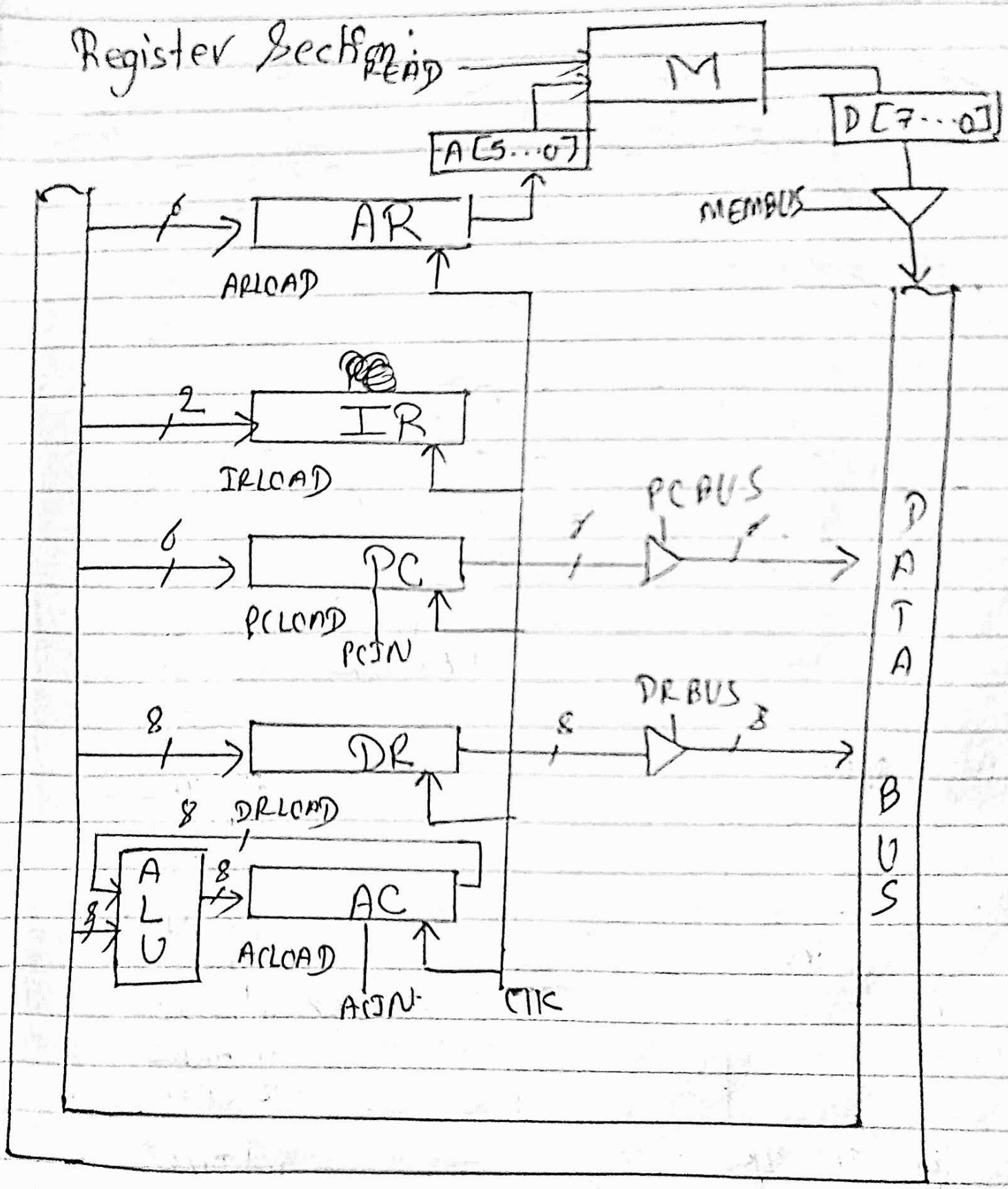
⇒ RTL Code:

```
FETCH1 :  AR ← PC
FETCH2 :  DR ← M ,  PC ← PC+1
FETCH3 :  IR ← DR[7,0] ,  AR ← DR[5...0]

COM    :  AC ← AC'
AND1   :  DR ← M
AND2   :  AC ← AC ∧ DR
JMP    :  PC ← DR[5...0]
INC    :  AC ← AC+1
```

State diagram:

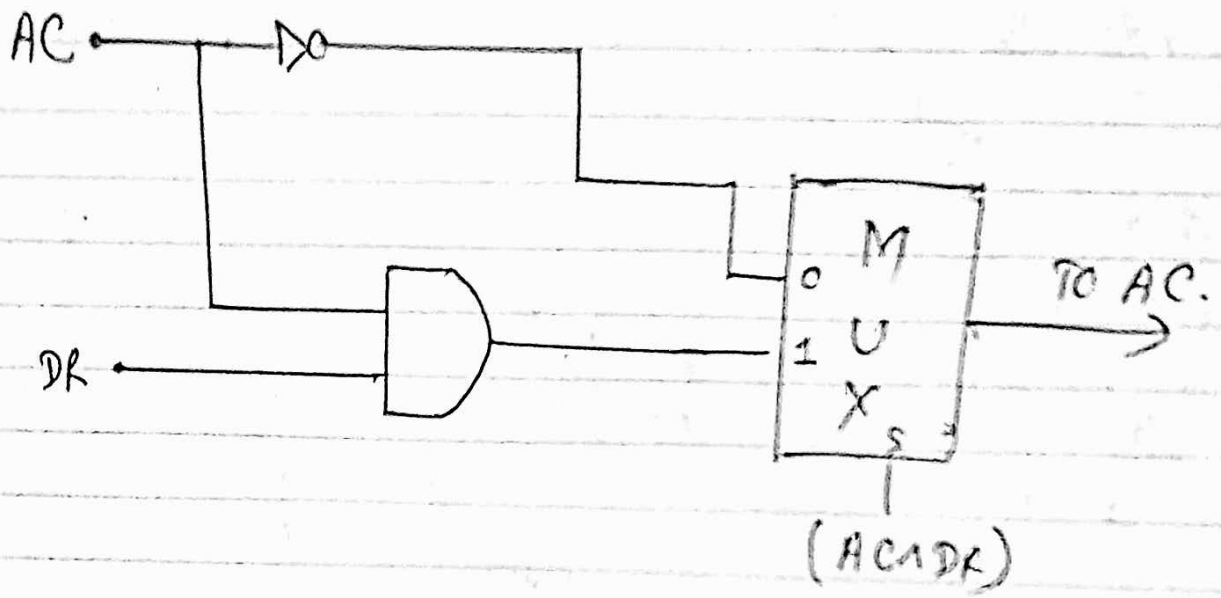



Register Section
READ
M
A[5...0]
D[7...0]
MEMBUS
AR
ARLOAD
2
IR
IRLOAD
PCBUS
6
PC
PCLOAD
PCIN
DATA
DRBUS
8
DR
8
DRLOAD
ALU
AC
ACLOAD
ACIN
CLK
BUS

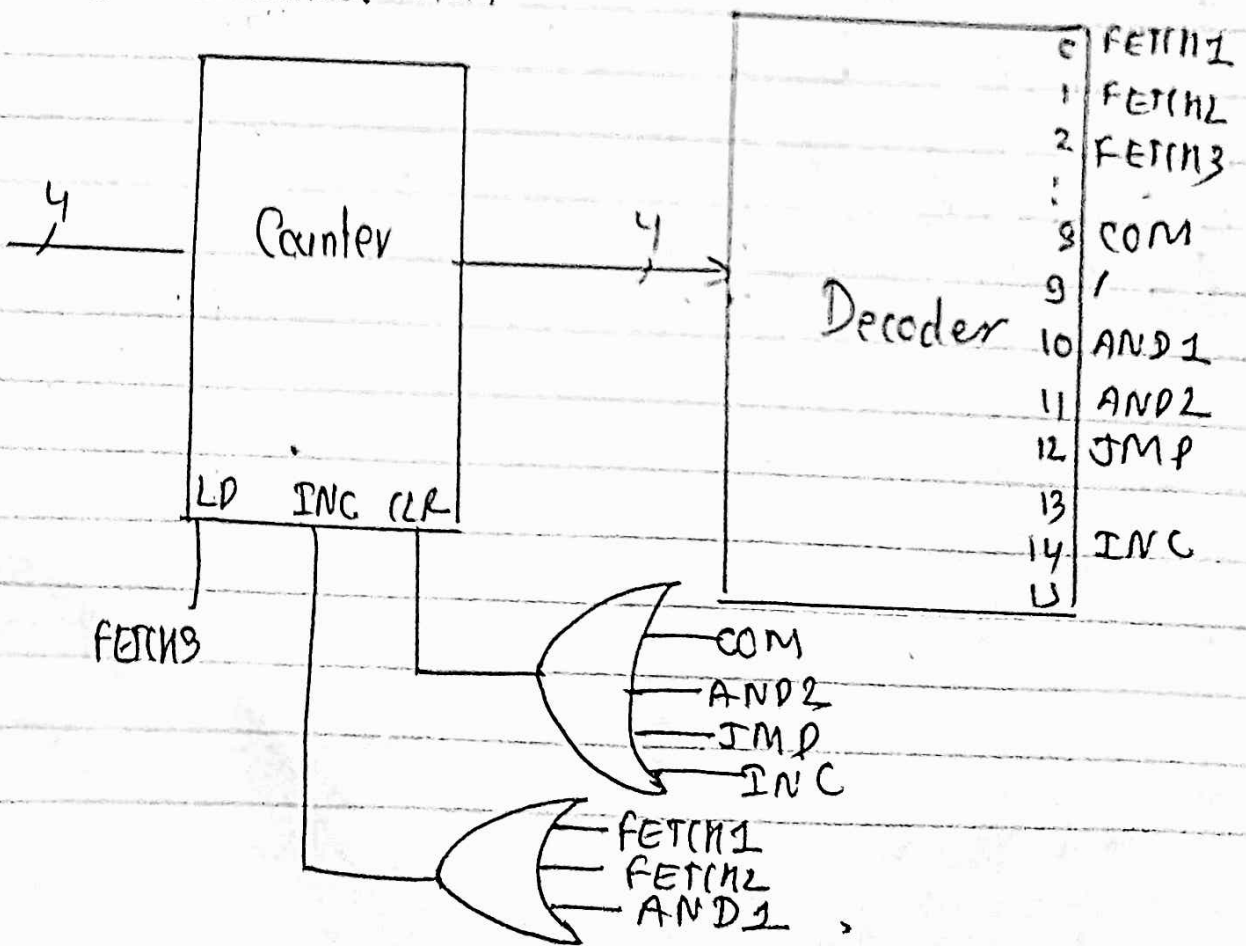
Qno. 3

(a)

⇒ ALU of very simple CPU:



Hardwired Control Unit.



Qno. 3

(b)

→ Micro-sequencer control unit with horizontal microcode:

Mnemonics of micro-operation

Mnemonics	Micro-operation
ARPC	AR ← PC
PCIN	PC ← PC+1
DRM	DR ← M
IRDR	IR ← DR[7,6]
AIRDR	AR ← DR[5...0]
COM	AC ← AC'
AND	AC ← AC ∧ DR
JMP	PC ← DR[5...0]
INC	AC ← AC+1

Now,

Partial micro-code for horizontal

State	Address	SEL	ADDR
FETCH1	0000	0	0001
FETCH2	0001	0	0010
FETCH3	0010	1	XXXX
COM	1000	0	0000
AND1	1010	0	1011
AND2	1011	0	0000
JMP	1100	0	0000
INC	1110	0	0000

8 4 21

Now
The horizontal micro-code for CPU.

State	Address	SEL	ARPC	DRM	PCIN	IRDR	ARDR	COM	AND	JMP	INC	ADDR
FETCH1	0000	0	1	0	0	0	0	0	0	0	0	0001
FETCH2	0001	0	0	1	1	0	0	0	0	0	0	0010
FETCH3	0010	1	0	0	0	1	1	0	0	0	0	XXXX
COM	1000	0	0	0	0	0	0	1	0	0	0	0000
AND1	1010	0	0	1	0	0	0	0	0	0	0	1011
AND2	1011	0	0	0	0	0	0	0	1	0	0	0000
JMP	1100	0	0	0	0	0	0	0	0	1	0	0000
INC	1110	0	0	0	0	0	0	0	0	0	1	0000

Now,
the signal of ARDR and IRDR are same.
So, optimized horizontal micro-m

FETCH1	0000	0	1	0	0	0	0	0	0	0	0001
FETCH2	0001	0	0	1	1	0	0	0	0	0	0010
FETCH3	0010	1	0	0	0	1	0	0	0	0	XXXX
COM	1000	0	0	0	0	0	1	0	0	0	0000
AND1	1010	0	0	1	0	0	0	0	0	0	1011
AND2	1011	0	0	0	0	0	0	1	0	0	0000
JMP	1100	0	0	0	0	0	0	0	1	0	0000
INC	1110	0	0	0	0	0	0	0	0	1	0000

Q no. 2017 Spring:

Q no. 3

(a)

⇒ RTL Code:

FETCH1 : AR ← PC

FETCH2 : DR ← M , PC ← PC+1

FETCH3 : IR ← DR[7,6] , AR ← DR[5...0]

STK1 : DR ← AC (ACBUS)

STR2 : M ← DR (BUSMEM) (WRITE)

AND1 : DR ← M

AND2 : AC ← AC ∧ DR

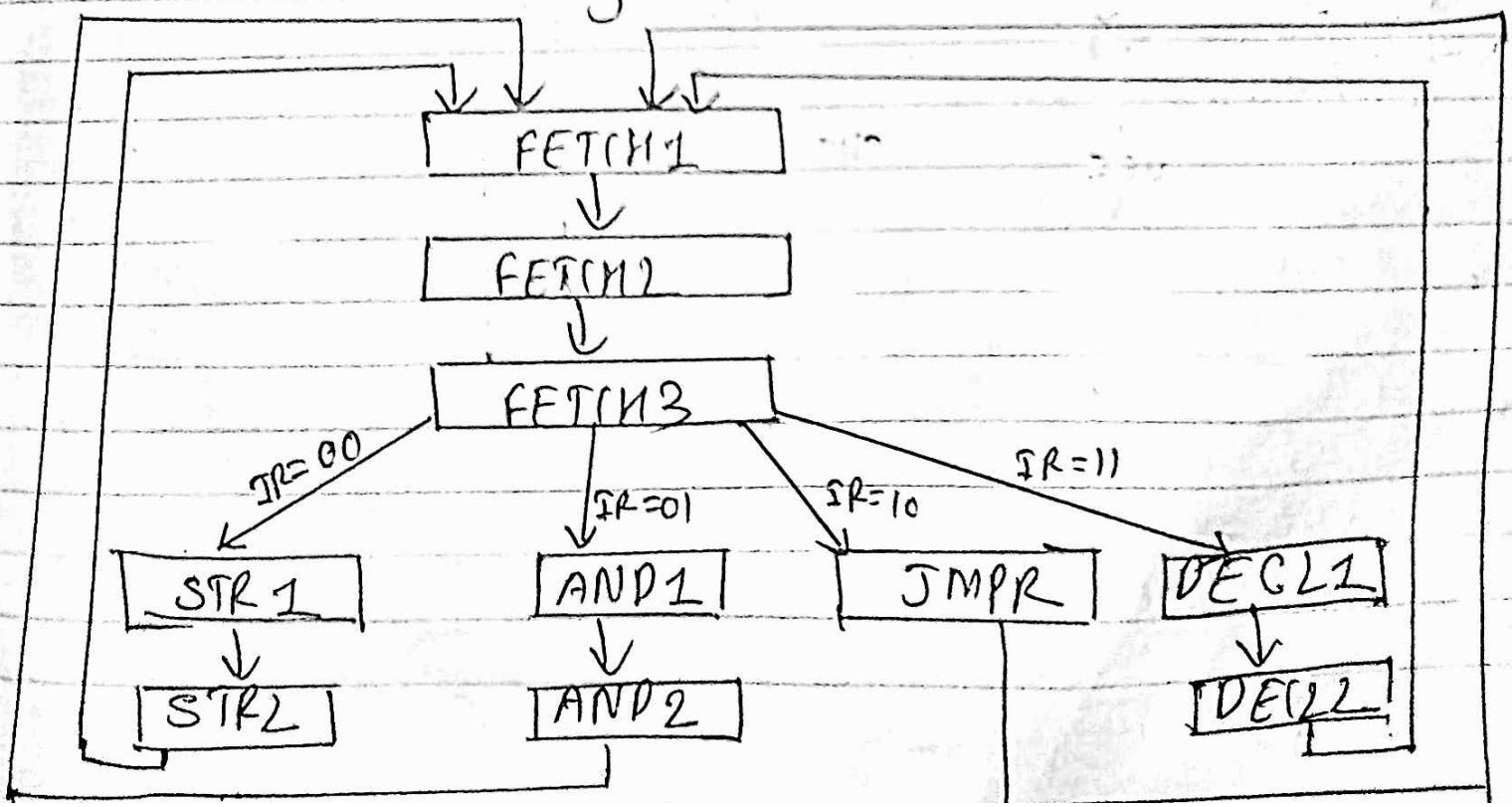
JMPR : PC ← PC + DR[5...0]

DEC21 : AC ← AC − 1

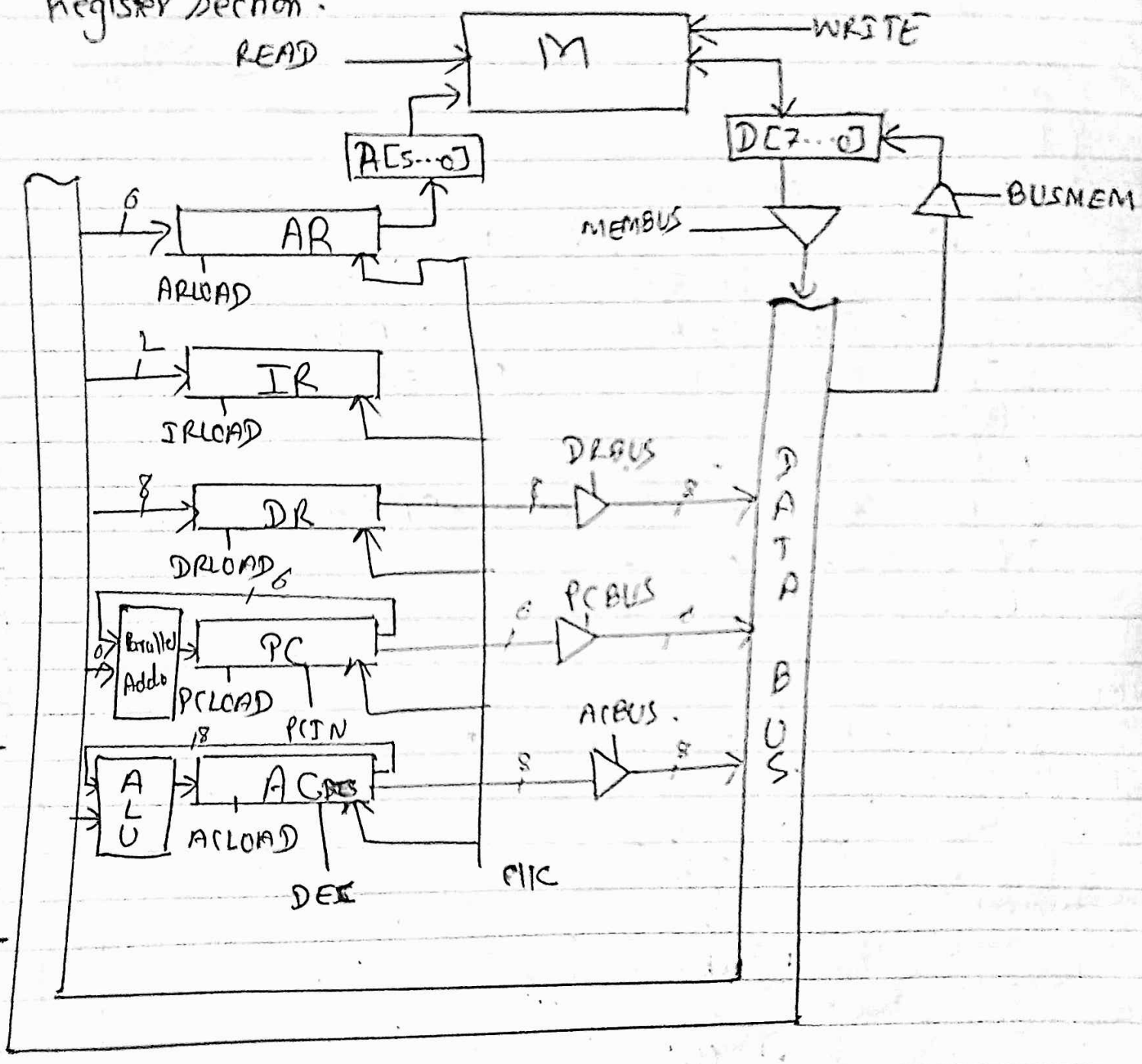
DEC22 : AC ← AC − 1

Now:

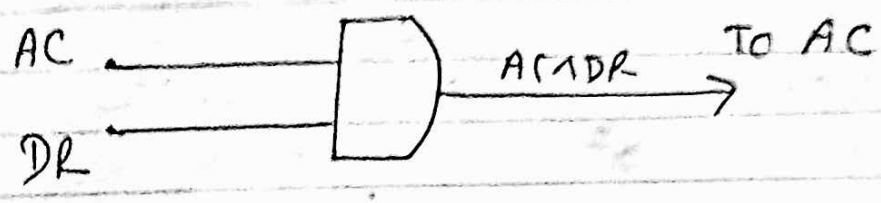
CPU State Diagram.



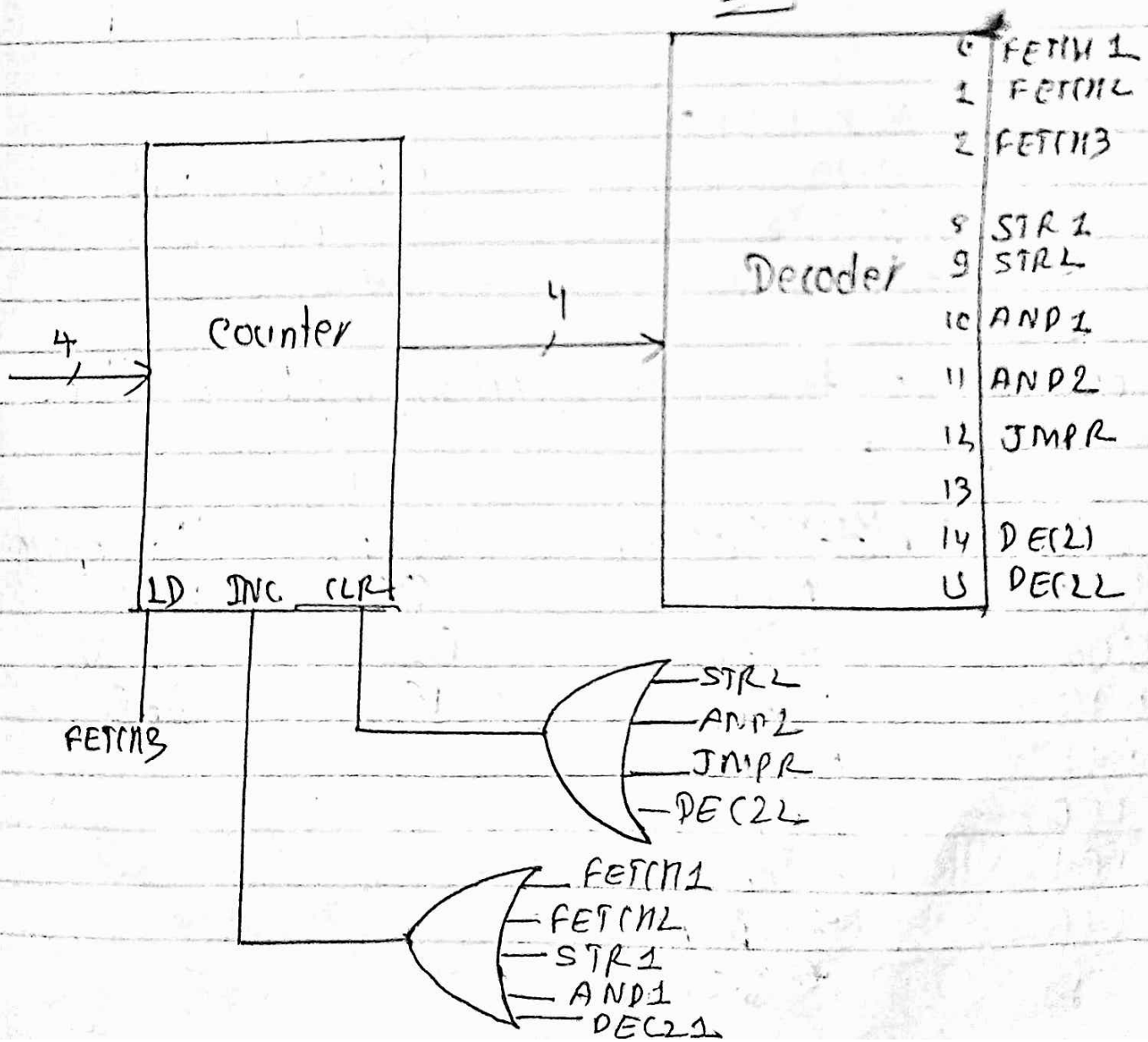
Register Section:



ALU of very simple CPU.



Qno. 3 (b) Hardwired Control Unit



Q no. 4 (a)

Microsequencer control unit with Vertical micro-code:

Mnemonics:

Mnemonics	Micro-operation
ARPC	AR ← PC
DRM	DR ← M
PCIN	PC ← PC + 1
AIDR	IR ← DR[7,6], AR ← DR[5...0]
DRAC	DR ← AC
MDR	M ← DR
AND	AC ← AC ∧ DR
JMPR	PC ← PC + DR[5...0]
DEC	AC ← AC − 1

Now, Assign the micro-operation field and value.

M1

Value	Micro-operation
000	NOP
001	ARPC
010	DRM
011	AIDR
100	DRAC
101	MDR
110	AND
111	JMPR

M2

Value	Micro-operation
00	NOP
01	PCIN
10	DEC

Now, micro-sequencer control unit with vertical micro-code:

State	Address	SEL	M1	ML	ADDR
FETCH1	0000	0	001	00	0001
FETCH2	0001	0	010	01	0010
FETCH3	0010	1	011	00	XXXX
STR1	1000	0	100	00	1001
STR2	1001	0	101	00	0000
AND1	1010	0	010	00	1011
AND2	1011	0	110	00	0000
JMPR	1100	0	111	00	0000
DEC1	1110	0	000	10	1111
DEC2	1111	0	000	10	0000

2018 fall

⇒ RTL Code:

FETCH1 : AR ← PC

FETCH2 : DR ← M , PC ← PC+1

FETCH3 : IR ← DR[7,6], AR ← DR[5...0]

STA1 : DR ← AC

STA2 : M ← DR.

XNOR1 : DR ← M

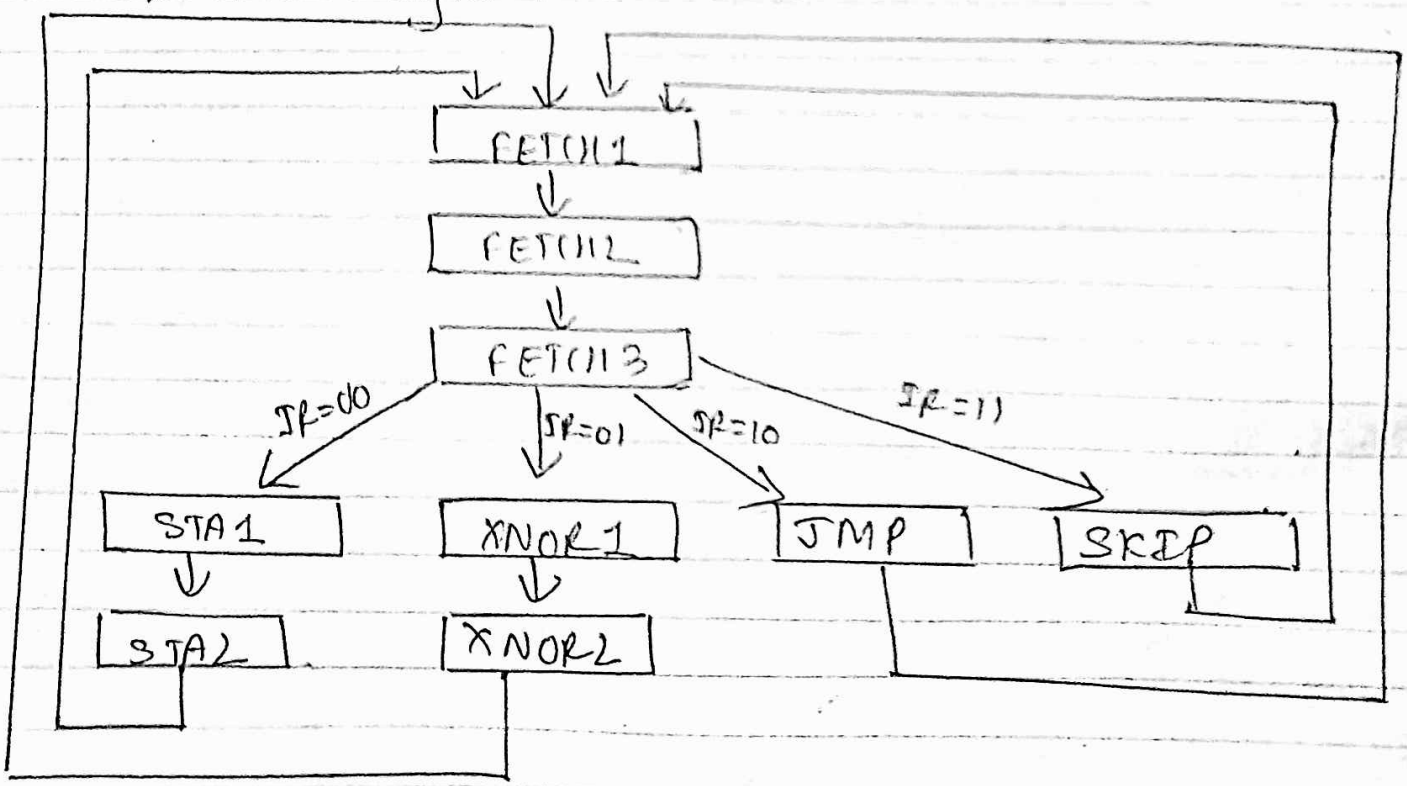
XNOR2 : AC ← AC ⊙ DR

JMP : PC ← DR[5...0]

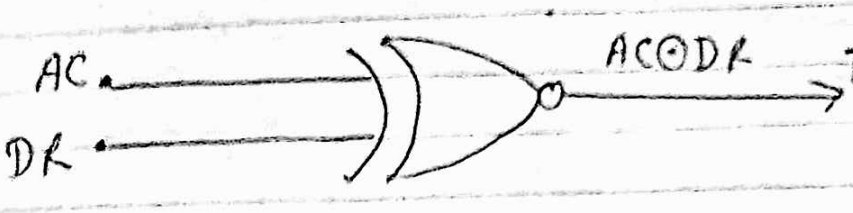
SKIP : PC ← PC+1.

Now,

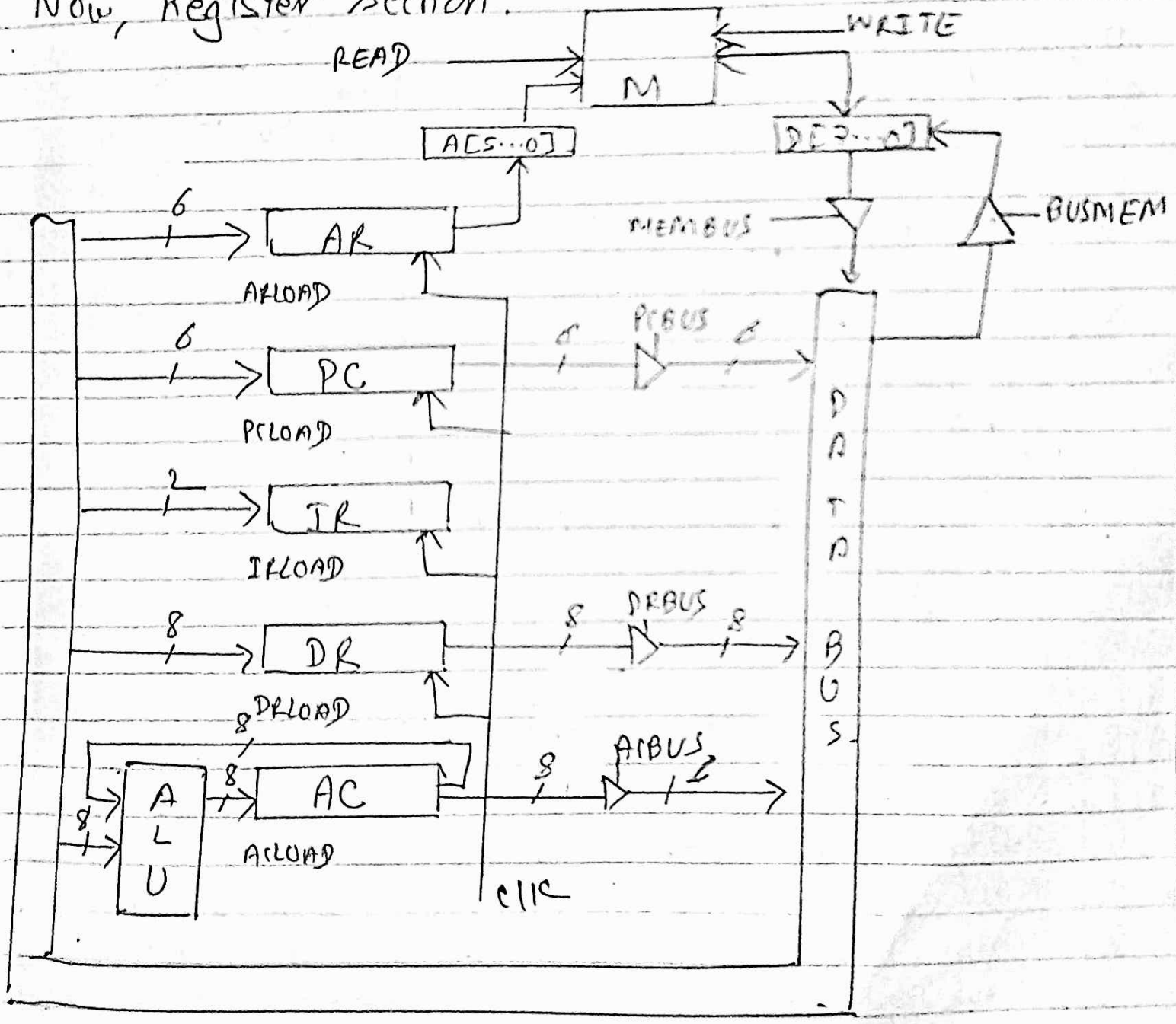
State Diagram of CPU.



Now, ALU:

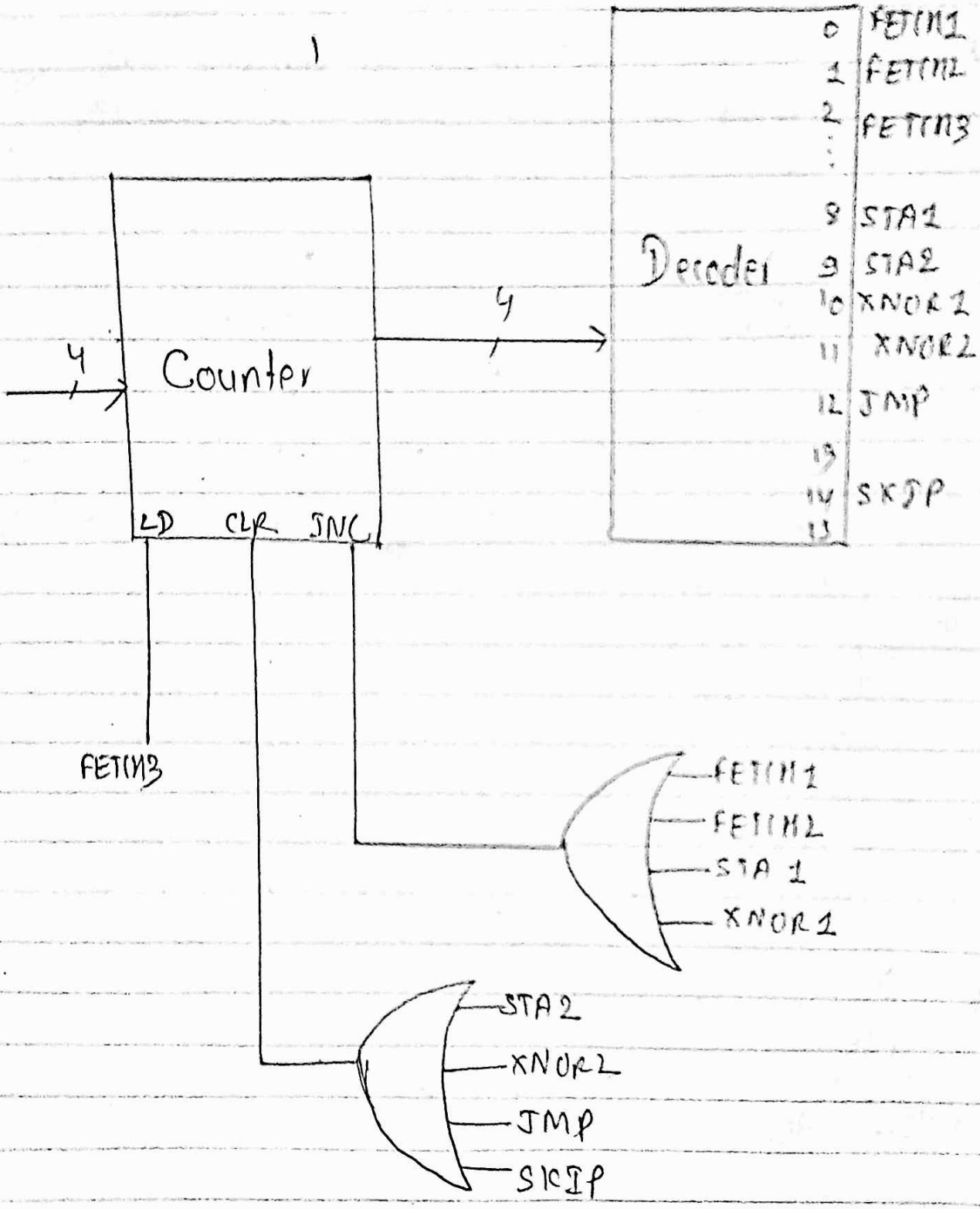


Now, Register Section:



Q no. 3(b)

Now, Hardwired control Unit:



Qno. 4(a)

⇒ for microsequencer control unit with horizontal micro-code:

Firstly,

Mnemonic of microoperation:

Mnemonics	Micro-operation
ARPC	AR ← PC
DRM	DR ← M
PCIN	PC ← PC + 1.
IRDR	IR ← DR[7,6]
ARDR	AR ← DR[5...0]
DRAC	DR ← AC
MDR	M ← DR
XNOR	AC ← AC ⊙ DR
JMP	PC ← DR[5...0]

Now, Partial horizontal microcode is :-

State	Address	SEL	ADDR
FETCH1	0000	0	0001
FETCH2	0001	0	0010
FETCH3	0010	1	xxxx
STA1	1000	0	1001
STA2	1001	0	0000
XNOR1	1010	0	1011
XNOR2	1011	0	0000
JMP	1100	0	0000
SKIP	1110	0	0000

Horizontal Microcode for micro-sequencer:

State	Address	SEL	ARPC	DRM	PCIN	IRDR	ABDR	DRAC	MDR	XNOR	JMP	ADDR
FETCH1	0000	0	1			0	0					0001
FETCH2	0001	0		1	1	0	0					0010
FETCH3	0010	1				1	1					XXXX
STA1	1000	0				0	0	1	1			1001
STA2	1001	0				0	0		1			0000
XNOR1	1010	0		1		0	0					1011
XNOR2	1011	0				0	0			1		0000
JMP	1100	0				0	0				1	0000
SKIP	1110	0			1	0	0					0000

AIDR will be combined signal for IRDR & ABDR as ARDR & IRDR signal are same. We get

State	Address	SEL	ARPC	DRM	PCIN	AIDR	DRAC	MDR	XNOR	JMP	ADDR
FETCH1	0000	0	1								0001
FETCH2	0001	0		1	1						0010
FETCH3	0010	1				1					XXXX
STA1	1000	0					1				1001
STA2	1001	0						1			0000
XNOR1	1010	0		1							1011
XNOR2	1011	0							1		0000
JMP	1100	0								1	0000
SKIP	1110	0			1						0000

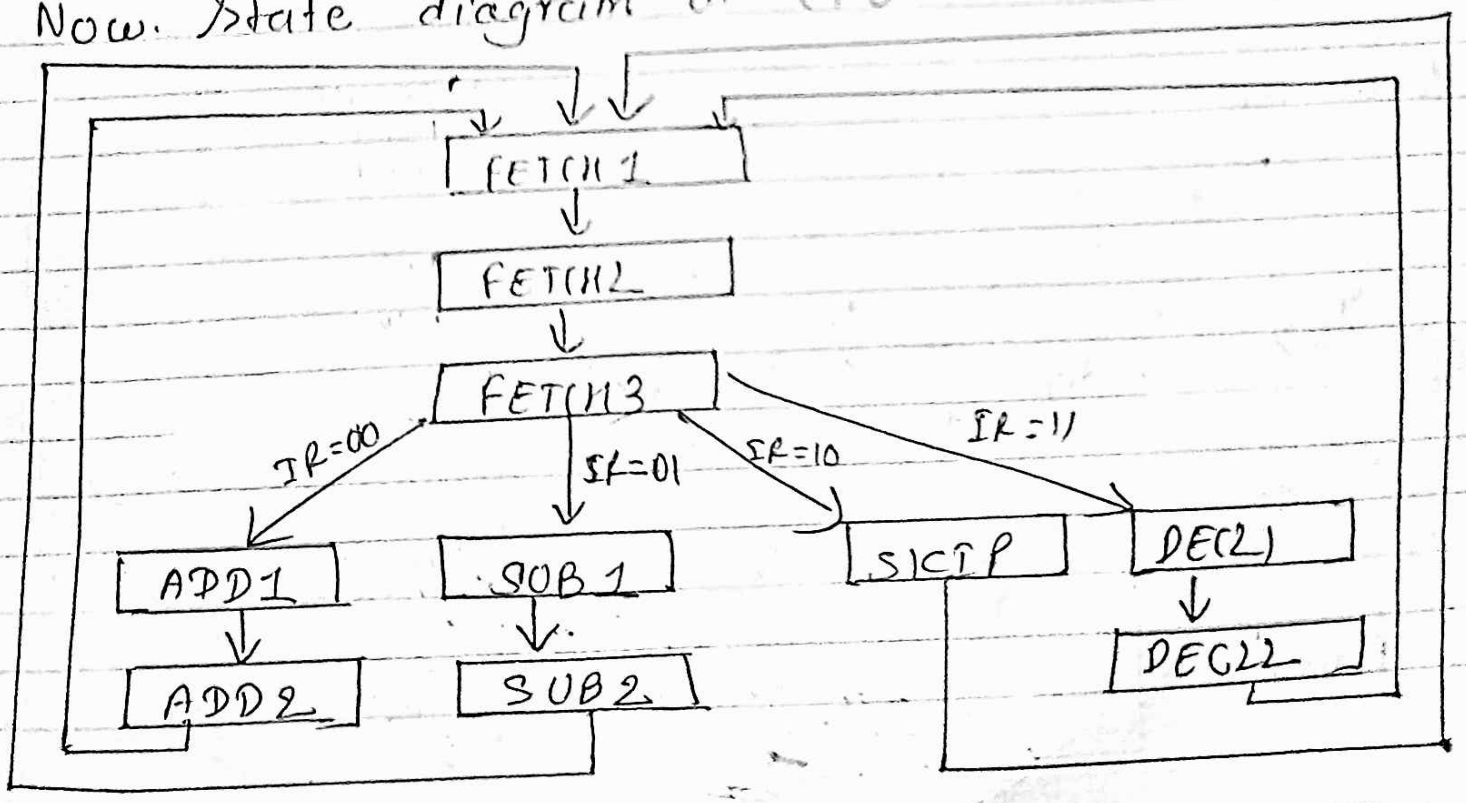
17 Spring

Qno. 3 @

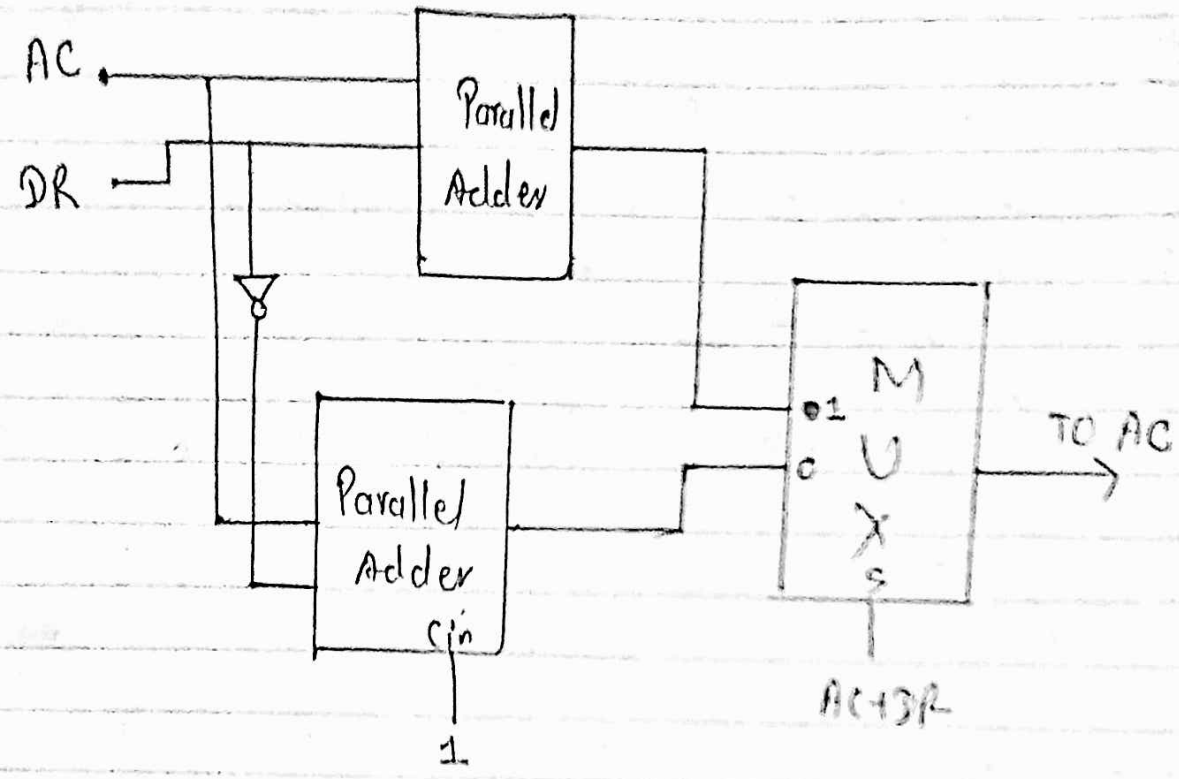
⇒ Soln: RTL code:-

FETCH1 : AR ← PC
FETCH2 : DR ← M , PC ← PC+1
FETCH3 : IR ← DR[7,6], AR ← DR[5...0]
ADD1 : DR ← M
ADD2 : AC ← AC + DR
SUB1 : DR ← M
SUB2 : AC ← AC + DR' + 1 [i.e. AC ← AC - DR]
SKIP : PC ← PC + 1
DEC21 : AC ← AC - 1.
DEC22 : AC ← AC - 1.

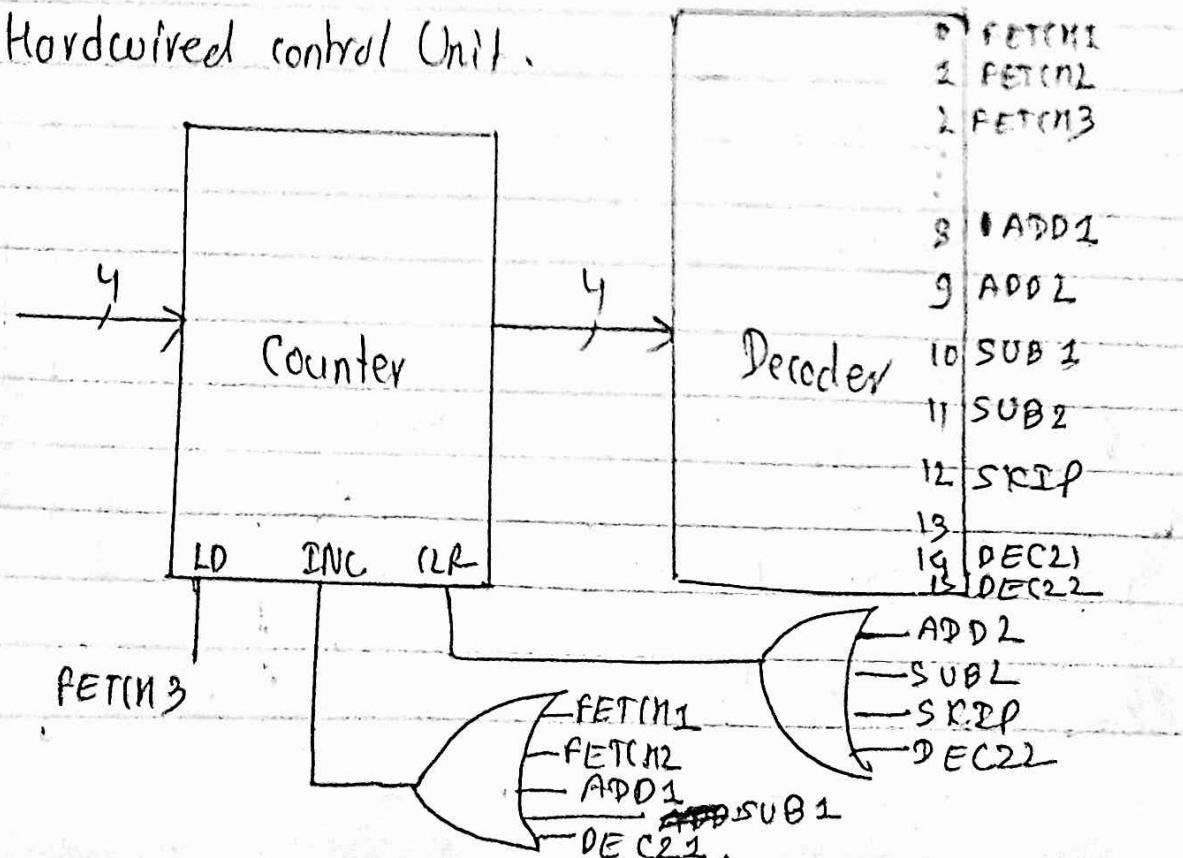
Now. State diagram of CPU.



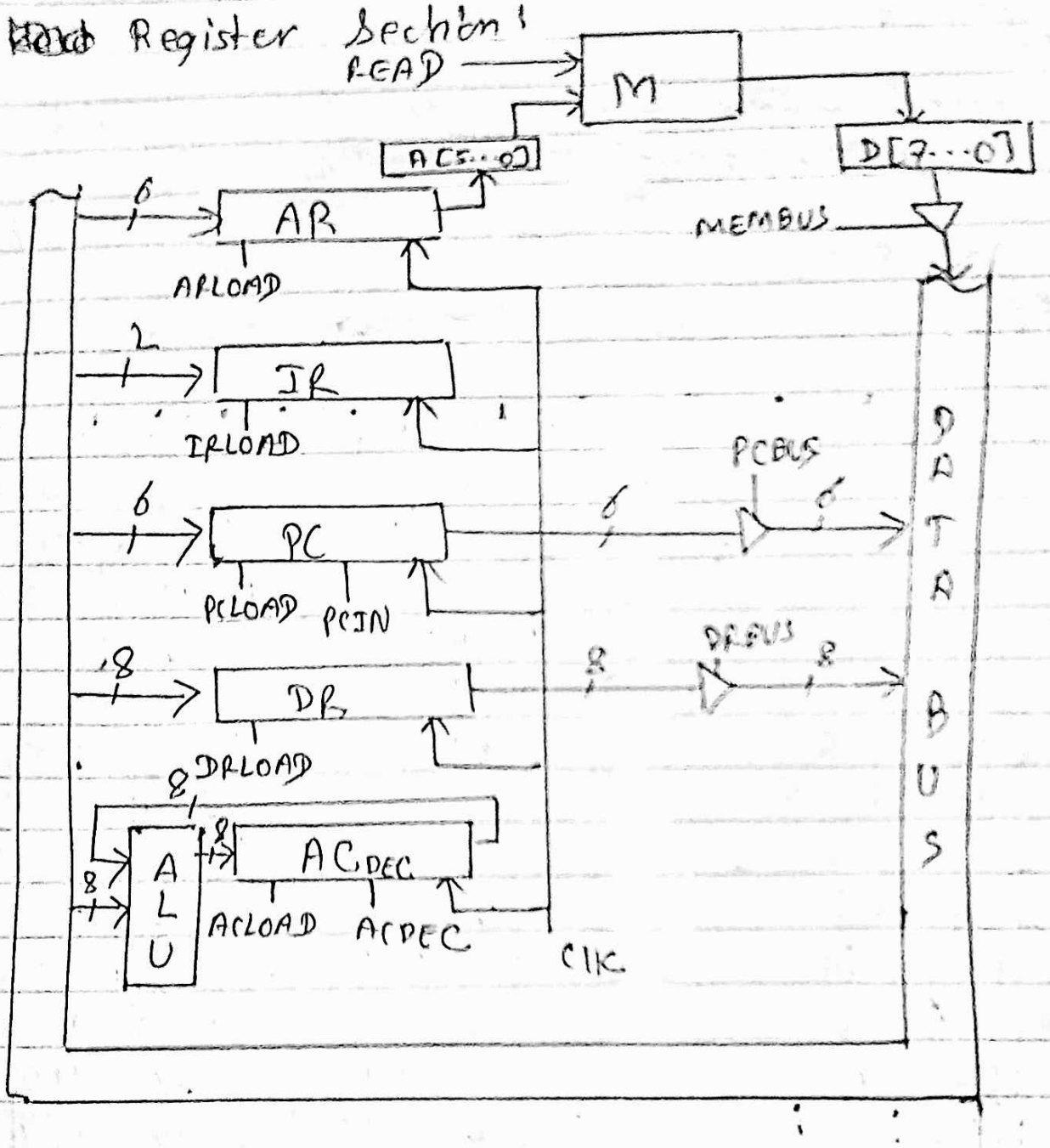
ALU of very simple CPU:



Hardwired control Unit.



Register Section



Ans. 4(a)

⇒ Micro-sequencer control unit which directly generate control signals

~~Mnemonics~~

State	Address	SEL	ARLOAD	IRLOAD	PCLOAD	DRLOAD	ACLOAD	PCIN	PCBUS	MEMBUS	READ	ADDR	DRBUS	ACDEC
FETCH1	0000	0	1			0			1	0	0	0001		
FETCH2	0001	0				1		1		1	1	0010		
FETCH3	0010	1	1	1		0				0	0	xxxx	1	
ADD1	1000	0				1				1	1	1001		
ADD2	1001	0				0	1			0	0	0000	1	
SUB1	1010	0				1				1	1	1011		
SUB2	1011	0				0	1			0	0	0000	1	
SKIP	1100	0				0		1		0	0	0000		
DEC2	1110	0				0				0	0	1111		1
DEC22	1111	0				0				0	0	0000		1

Here, DRLOAD, MEMBUS and READ signal same so common signal be DRMERD

Optimized table

State	Address	SEL	ARLOAD	IRLOAD	PCLOAD	ACLOAD	PCINC	PCBUS	DRBUS	ACDEC	DRMEMD	ADDR
FETCH1	0000	0	1					1			0	0001
FETCH2	0001	0					1				1	0010
FETCH3	0010	1	1	1							0	XXXX
ADD1	1000	0									1	1001
ADD2	1001	0				1					0	0000
SUB1	1010	0									1	1011
SUB2	1011	0				1					0	0000
SKIP	1100	0					1				0	0000
DECI1	1110	0								1	0	1111
DECI2	1111	0								1	0	0000

A
C
D
E
C

14 fall

2@

⇒

1st To construct 16X8 ROM at 00H using 8X8 chips

So, we have,

ROM = 8X8

$= 2^3 \times 8$

∴ No. of data line = 8

No. of address line = 3

∴ No. of chip required = 2

2ndly, to construct 64X8 RAM at 80H using 64X4 chip

∴ RAM = 64X4

$= 2^6 \times 4$

∴ No. of address line = 6

No. of data line = 4

∴ No. of RAM chip required = 2

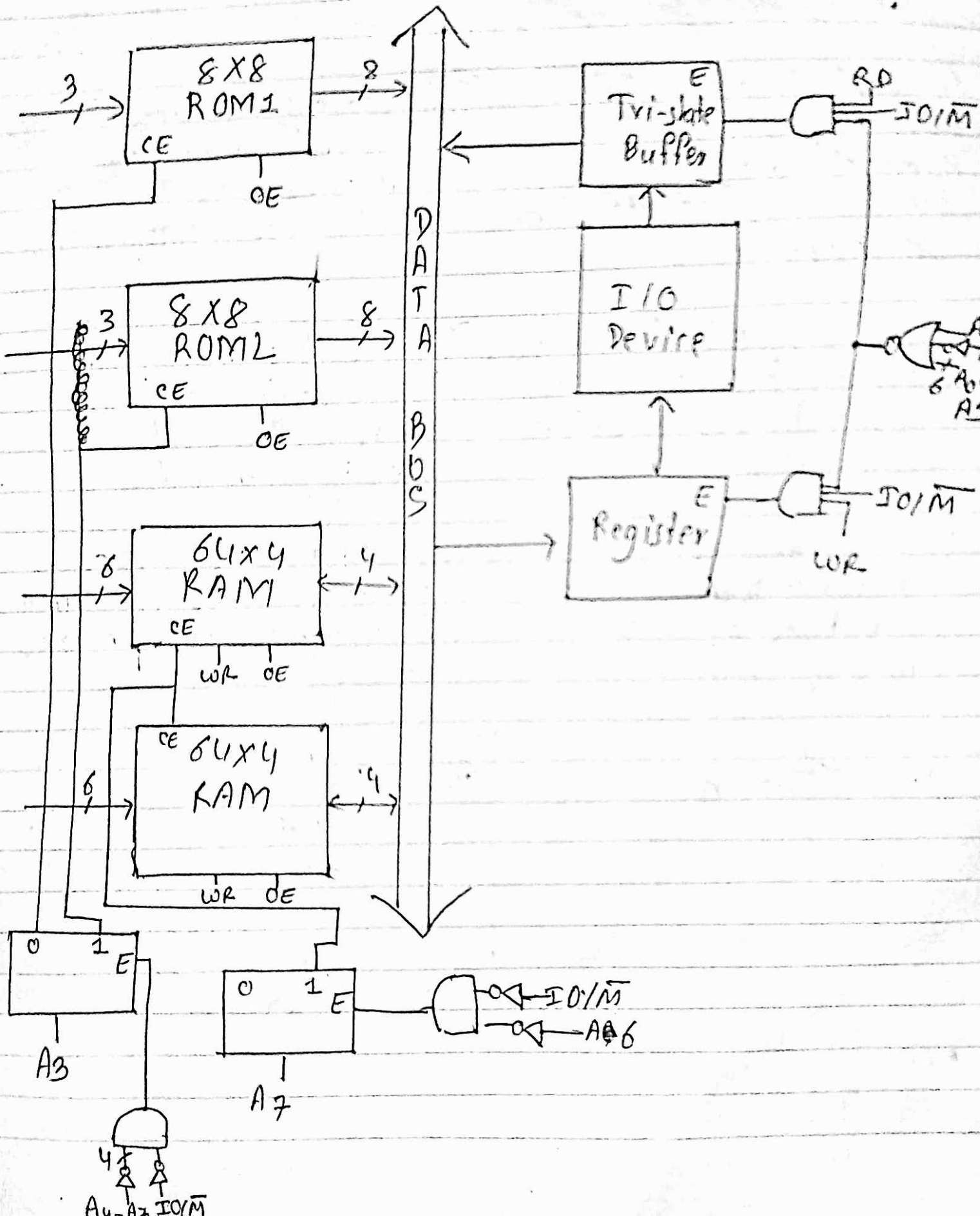
And, I/O device at 90H.

Now,

Address decoding

Memory	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Hex add.
ROM 1	0	0	0	0	0	0	0	0	00H
	0	0	0	0	0	1	1	1	07H
ROM 2	0	0	0	0	1	0	0	0	08H
	0	0	0	0	1	1	1	1	0FH
RAM	1	0	0	0	0	0	0	0	80H
	1	0	1	1	1	1	1	1	
I/O device	0	1	0	0	0	0	0	0	40H

Now,

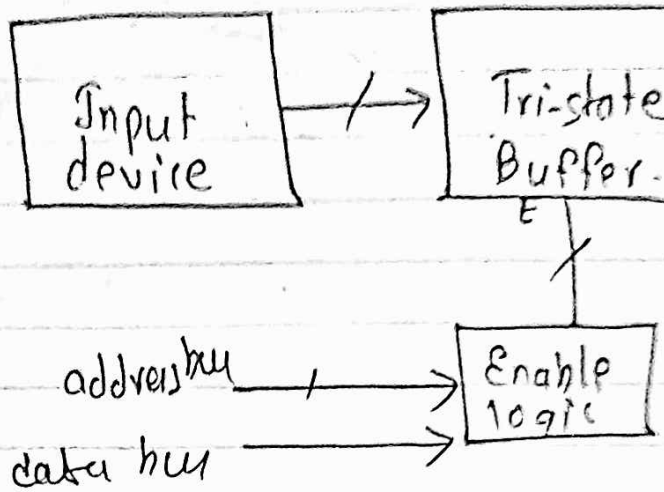

8X8 ROM1
8X8 ROM2
64X4 RAM
64X4 RAM
CE
OE
WR
3
8
6
4
DATA BUS
Tri-state Buffer
I/O Device
Register
E
RD
IO/M
WR
A3
A7
A4-A7 IO/M

High Order Interleaving	Low Order Interleaving
(i) In high-order interleaving, the most significant address bit is used to select the memory chip.	(i) In low order interleaving, the least significant address bit is used to select the memory chip.
(ii) In this interleaving, consecutive addresses tends to be in ~~different~~ same chip.	(ii) In this interleaving, consecutive addresses tends to be different memory modules.
(iii) The maximum rate of data transfer is limited by the memory cycle time.	(iii) This allows ~~the~~ memory access at much faster than the memory cycle time.
(iv) Design	(iv) Design

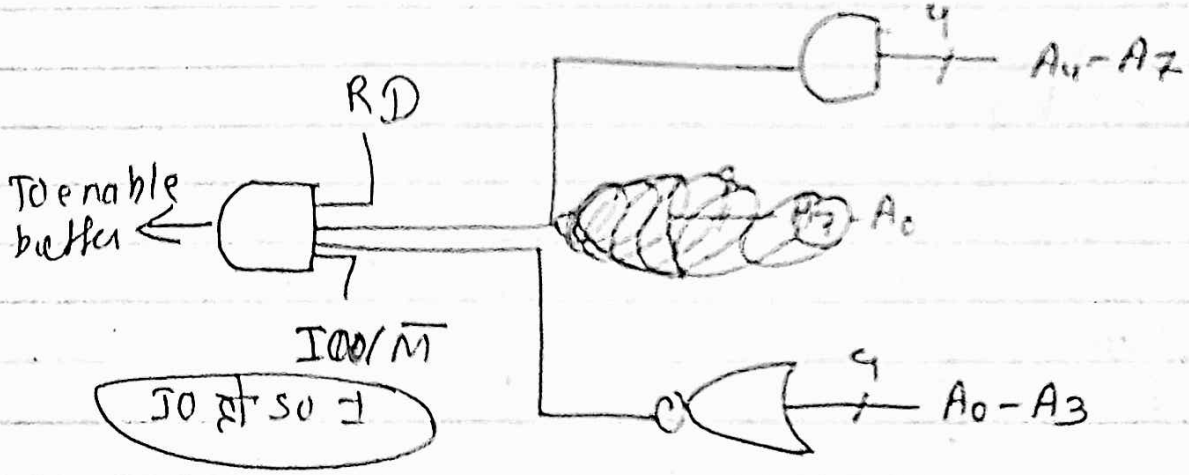
for the design of logic enable
* for input device, the tri-state buffer is required to control the data flow and read signal should be enable (RD) * so similarly IO/M signal enable

I/O subsystem organization and Interface.

* (ii) shows input device at address 11110000



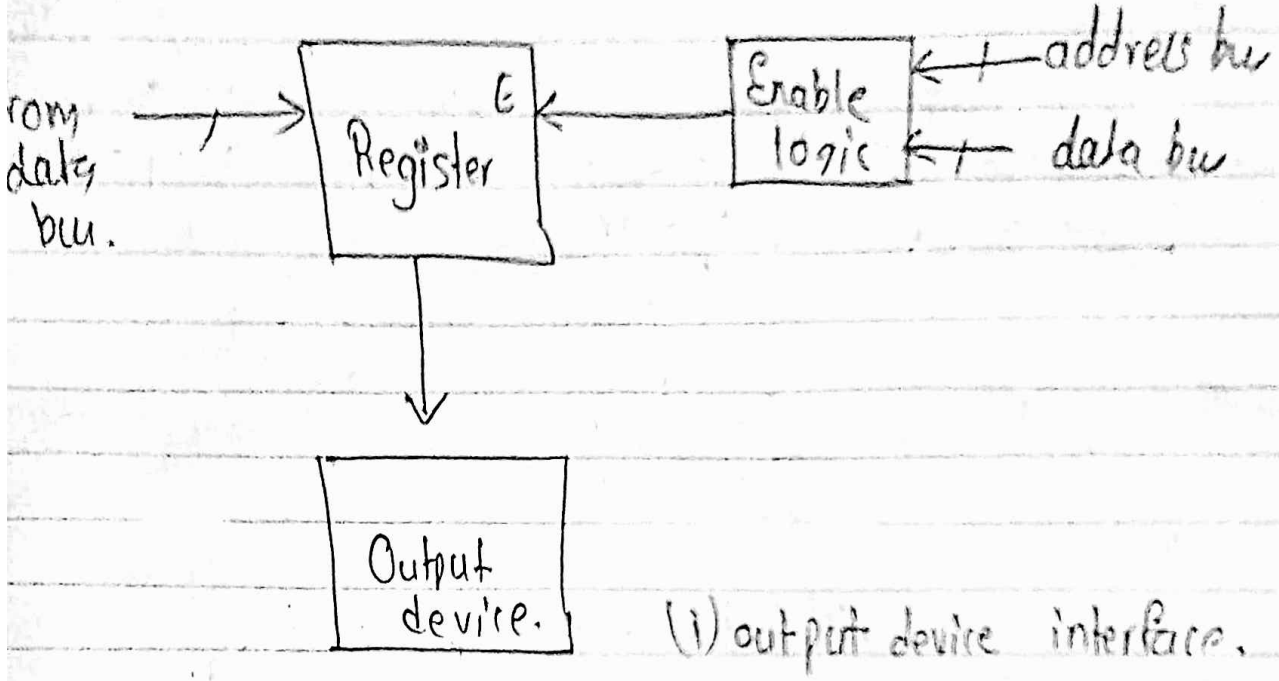
(i) Interface of Input device.



(ii) load logic for tri-state buffer.

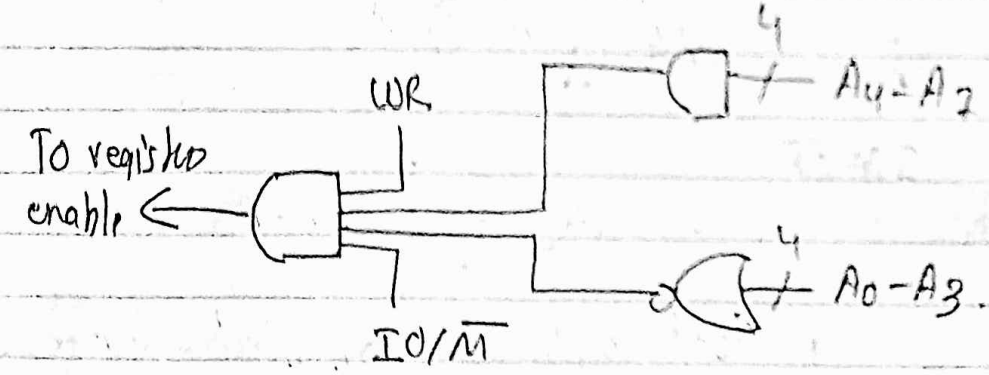
* The data from the input device goes to tri-state buffer
* When the values of address bus and data bus are correct, the tri-state buffer gets enable and data passes to data bus
* The CPU can then read the data.

Output device interface.



(i) output device interface.

Load logic.



(ii) Load logic for register

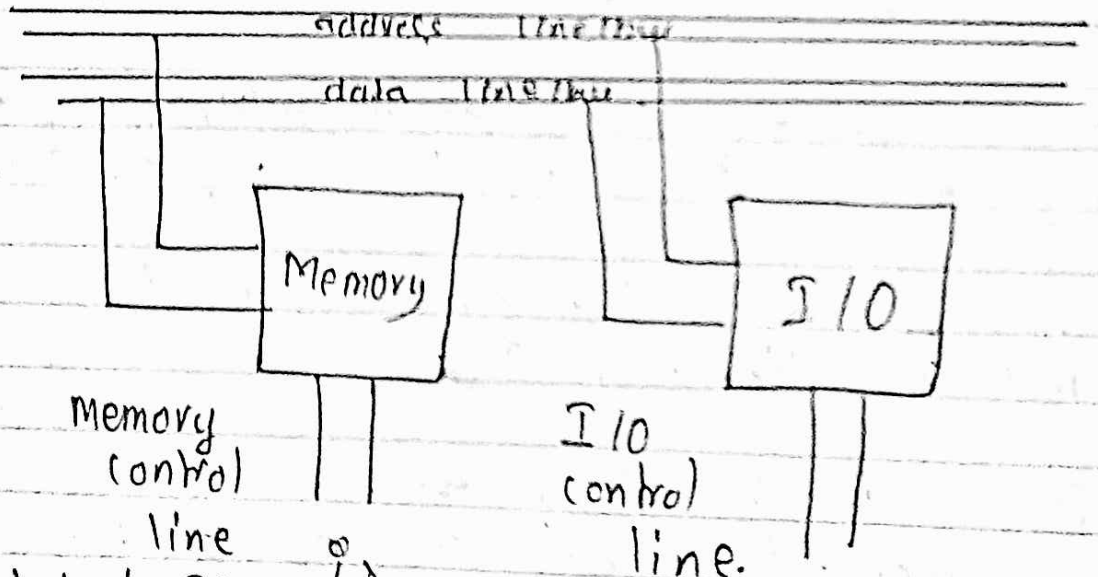
Memory mapped vs Isolated IO

① Isolated IO:

* In isolated IO, we have common address bus and data bus for IO and memory but separate read and write signal (control).

* So, when CPU decode instruction & then if data is for IO then ~~address~~ it places address on address line and set IO read or write line due to which the data transfer occur between IO and CPU.

Here we have different read-write instruction for I/O and memory.



* Isolated IO ~~are~~ more efficient due to separate buses.

* Isolated IO is larger in size due to more buses

3rd n

* It is complex due to separate logic is used to control both.

* Separate instructions control read and write operation are required for I/O and memory.

* I/O and memory have separate address space.

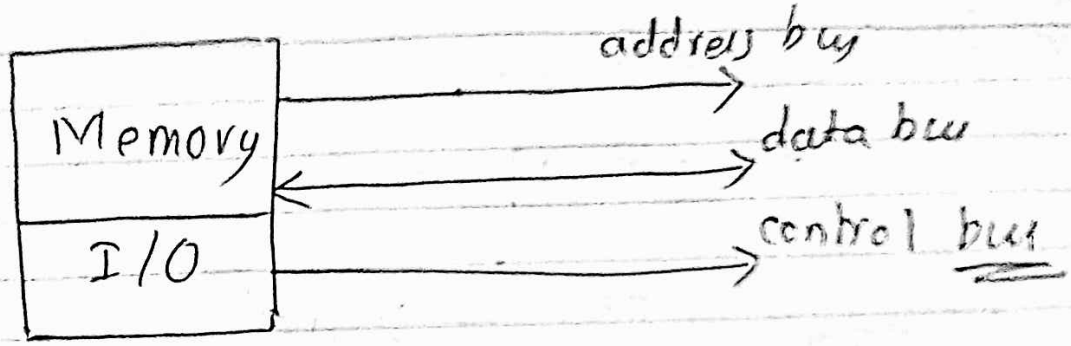
(II) Memory mapped I/O:
In memory mapped I/O, every bus is common due to which same instructions control work for both I/O and memory.

* I/O and memory have same address space so we manipulate I/O and memory as same.

* Memory mapped I/O are less efficient compared to Isolated I/O

* It is smaller in size
* Memory mapped I/O have simpler logic as I/O is also treated as memory

* Due to addition of I/O addressable actual memory becomes less for memory


Memory
I/O
address bus
data bus
control bus

17 Fall ??
1 b

2017 Spring

1 (b)

=) Construct 32 Byte of ROM at 10H using two 16 Byte ROM

ROM = 16 X 8
= 2^4 X 8
∴ No. of data line = 8
No. of address line = 4.

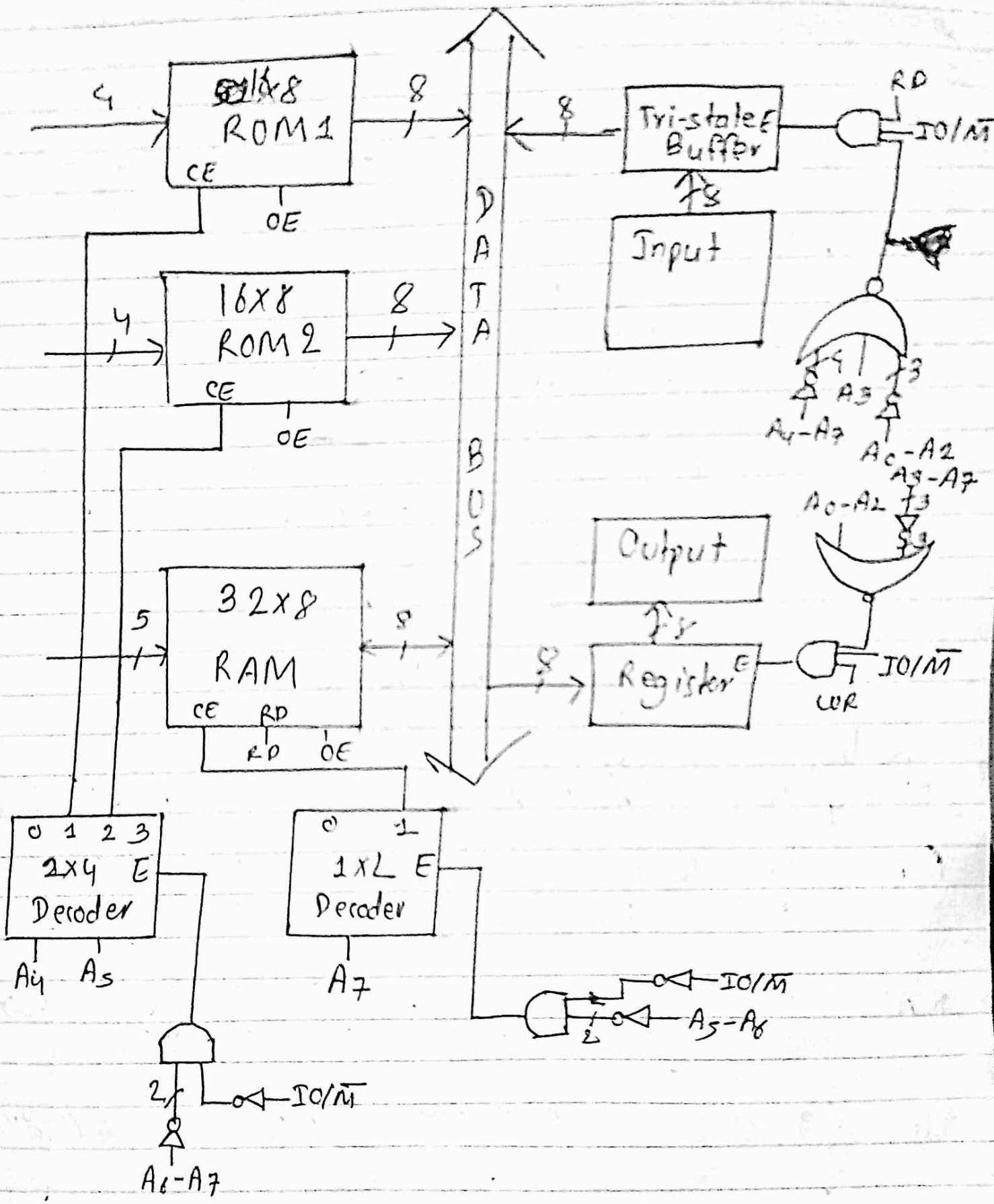
Also, need to construct 32 Byte of RAM at 80H

RAM = 32 X 8
= 2^5 X 8
∴ No. of address line = 5
No. of data line = 8

∴ Input device is at F7 H
Output device at F8 H

Address decoding

Memory	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Hex Address
ROM 1	0	0	0	1	0	0	0	0	10H
	0	0	0	1	1	1	1	1	1FH
ROM 2	0	0	1	0	0	0	0	0	20H
	0	0	1	0	1	1	1	1	
RAM	1	0	0	0	0	0	0	0	80H
	1	0	0	1	1	1	1	1	
Input	1	1	1	1	0	1	1	1	F7H
Output	1	1	1	1	1	0	0	0	F8H


4
16x8
ROM 1
CE
OE
8
8
Tri-state Buffer
RD
IO/M
8
Input
DATA
BUS
4
16x8
ROM 2
CE
OE
8
A4-A7
A3
A0-A2
5
32x8
RAM
CE
RD
RD
OE
8
8
Output
Register
E
IO/M
WR
0 1 2 3
2x4 E
Decoder
A4 A5
0 1
1x2 E
Decoder
A7
IO/M
A5-A6
2
2
IO/M
A6-A7

2020 Fall

1. (D)

⇒

To be constructed 8K of ROM at 0000H

$$\text{ROM} = 8K \times 8$$
$$= 2^3 \cdot 2^{10} \times 8$$
$$= 2^{13} \times 8$$

∴ No. of data line = 8

No. of address line = 13.

To be constructed 8K of RAM

$$\text{RAM} = 8K \times 8$$
$$= 2^3 \cdot 2^{10} \times 8$$
$$= 2^{13} \times 8$$

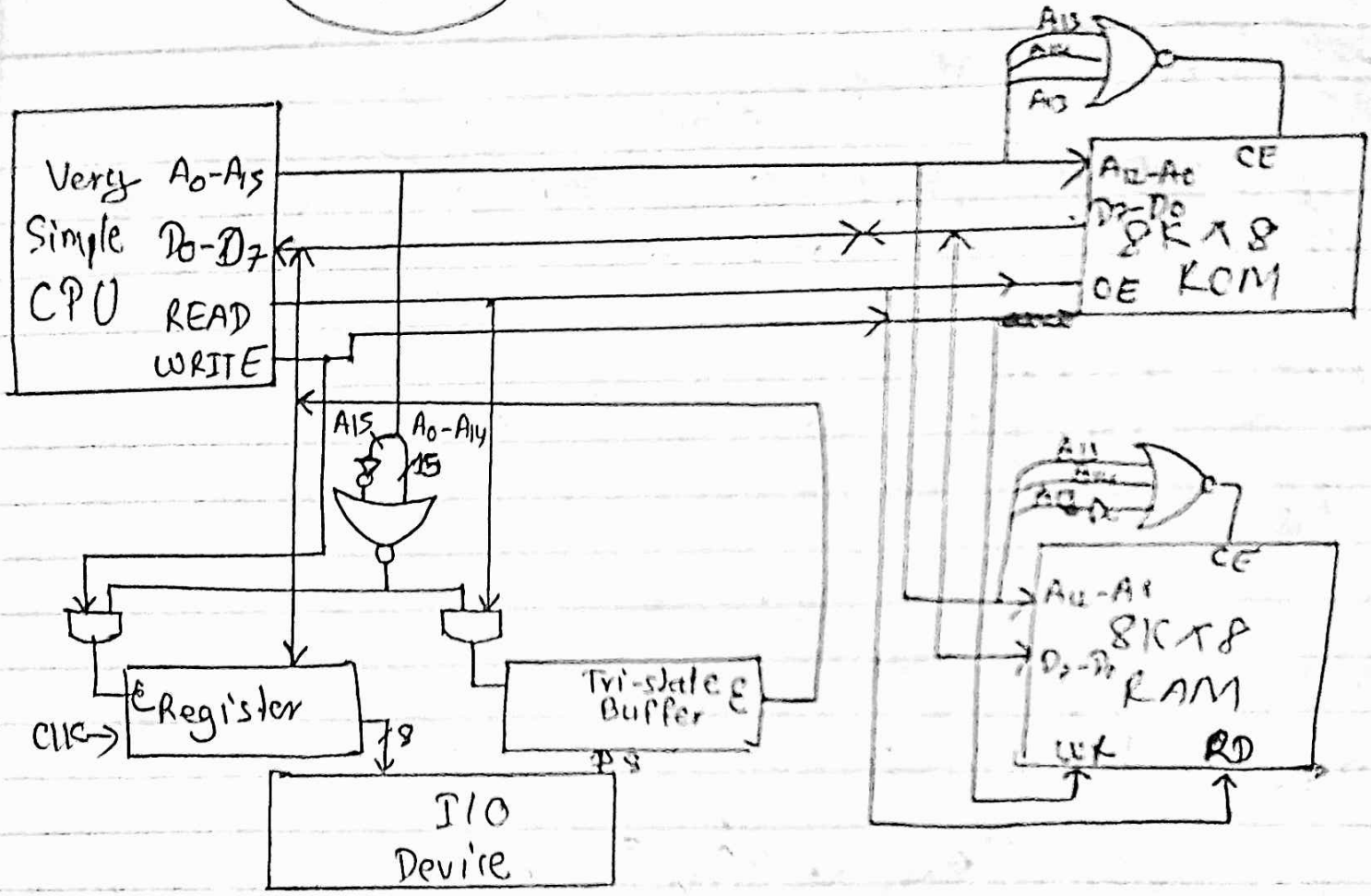
∴ No. of data line = 8

No. of address line = 13

Address Decoding:

Memory	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Hex address
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH
RAM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
I/O.	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H

18F, S
19F, 17D


Very Simple CPU
A0-A15
D0-D7
READ
WRITE
A15
A0-A14
15
8
Register
CLK
8
Tri-state Buffer
8
I/O Device
A15
A14
A13
CE
A12-A0
D7-D0
8K x 8 ROM
OE
A11
A10
A0
CE
A12-A0
D7-D0
8K x 8 RAM
WR
RD