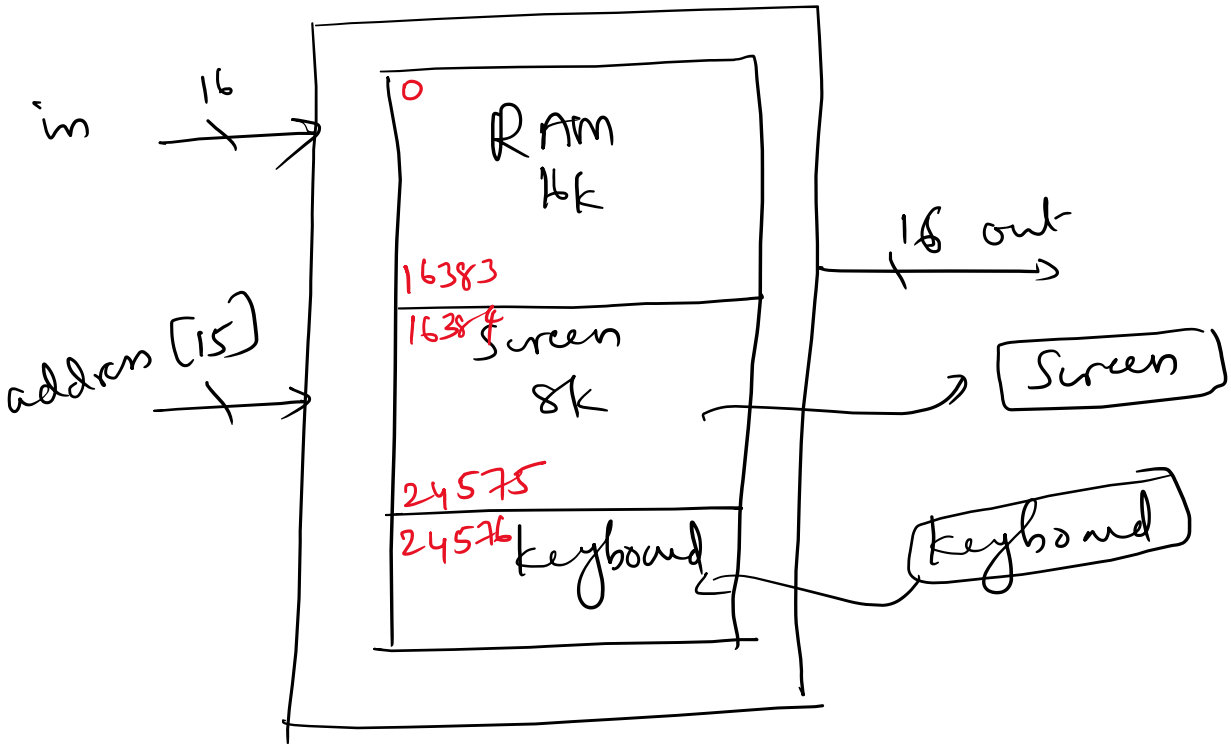
Week 5 – CPU

① Memory



→ 16 bit, 16k RAM chip → RAM
→ built in 8k screen chip → Screen
→ built in keyboard chip → keyboard

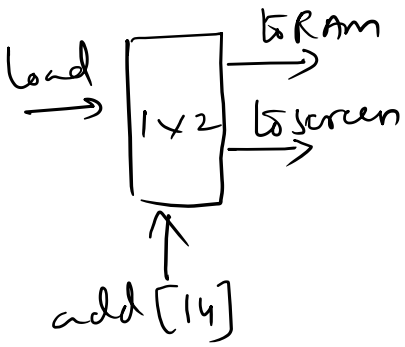
0	: 0000	⇒	000 000	RAM start
16383	: 3FFF	⇒	011 FFF	RAM end
16384	: 4000	⇒	100 000	screen start
24575	: 5FFF	⇒	101 FFF	screen end
24576	: 6000	⇒	110 000	keyboard

address[14]

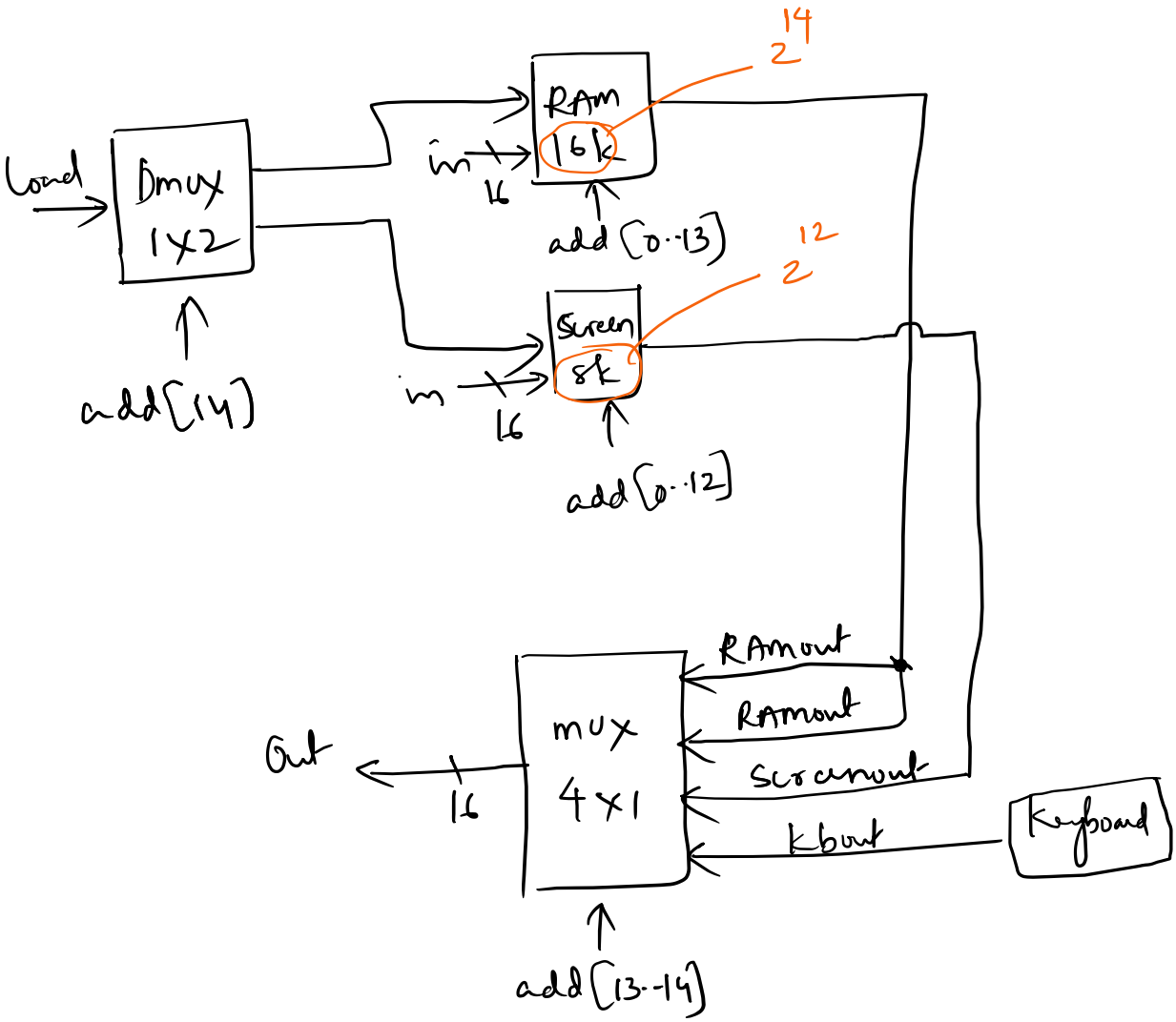
No I/P for keyboard

ig

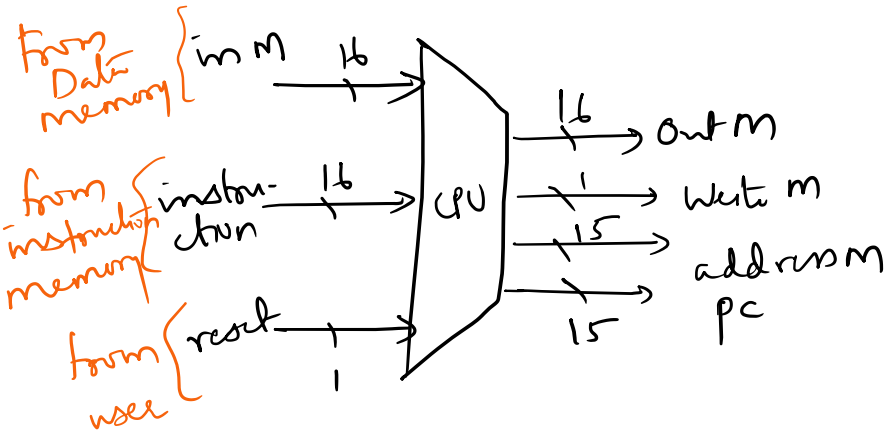
address[14]	Output
0	load
1	load

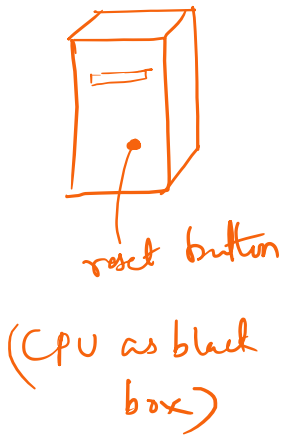


address[13..14]	Out
00	Ramout
01	Ramout
10	screenout
11	kbout



② CPU

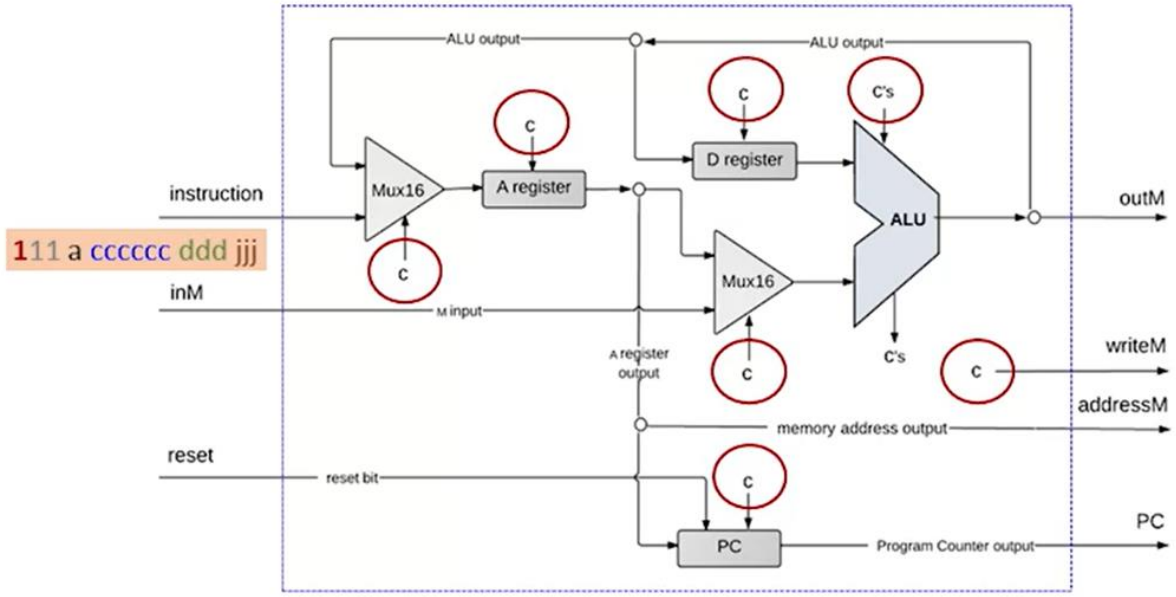




(CPU as black box)

* The ROM is loaded with a Hack program. When reset button is pushed the program starts running.
* At any point of time, there is always a selected register in the instruction memory and there is always a selected memory register in the data memory. So something always comes into the CPU.

CPU Implementation



The C-instruction: symbolic and binary syntax

Symbolic syntax: *dest* = *comp* ; *jump*

Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a=0	a=1						

dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register

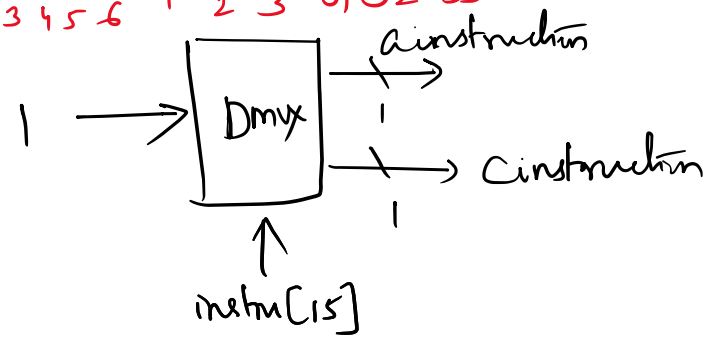
jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump

Instructions

A instruction $\Rightarrow$ Instruction[15] = 0 [@ value 15 bit]

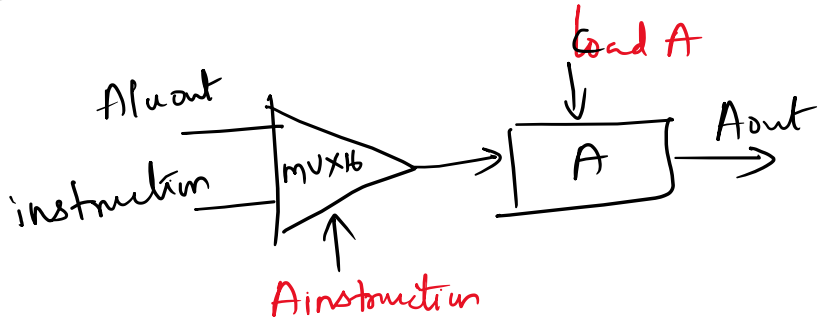
C instruction $\Rightarrow$ Instruction[15] = 1

111 a $c_1 c_2 c_3 c_4 c_5 c_6$ $d_1 d_2 d_3$ $j_1 j_2 j_3$



By this way we select the type of instruction.

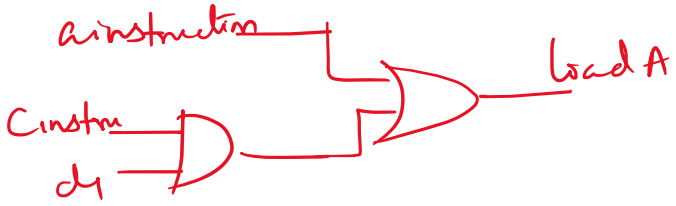
A Register



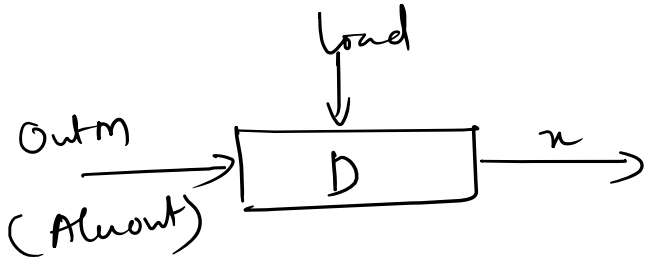
* If A instruction, store address in A
* for C instruction, if destination bit d_1 is 1, store result in A.

is load of A = 1 in these 2 cases.

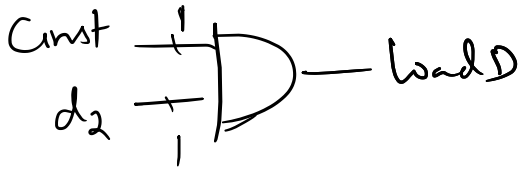
* If not an A instruction, ALU o/p is given to A.



D Register

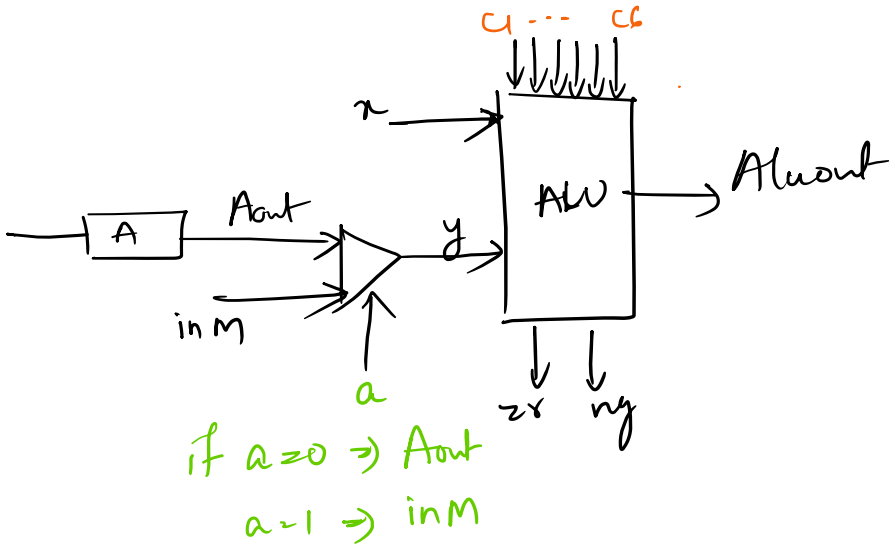


load = 1 when Cinst & d_2 = 1



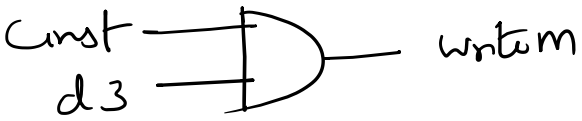
Cinst : 111 a c_1 c_2 c_3 c_4 c_5 c_6 d_1 d_2 d_3
j_1 j_2 j_3

ALU

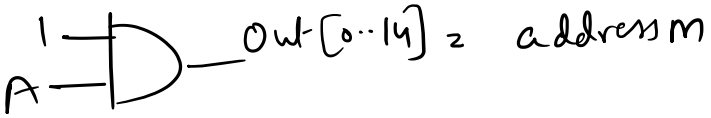


Outputs

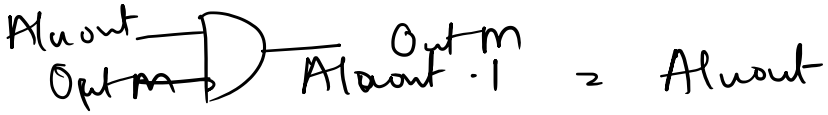
① writeM



② addressM = Aregister o/p [0 ... 14]



③ outM



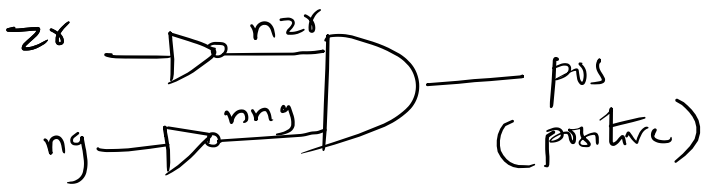
④ PC

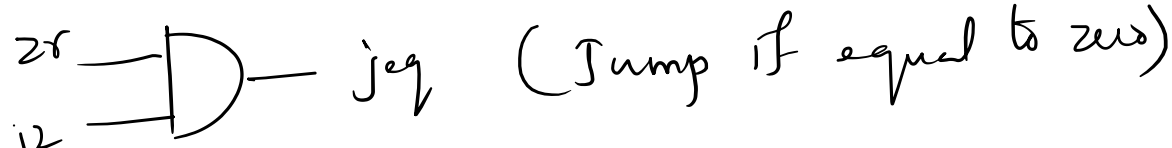
If jump, PC = A → jump = 1

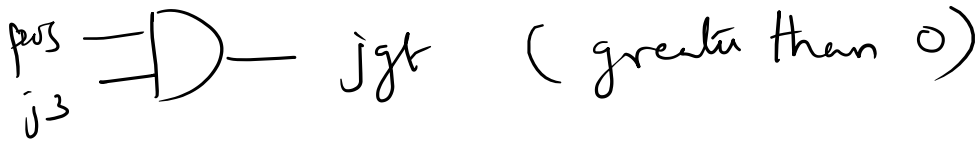
else, PC = PC + 1 → jump = 0

zr → Indicates o/p is zero

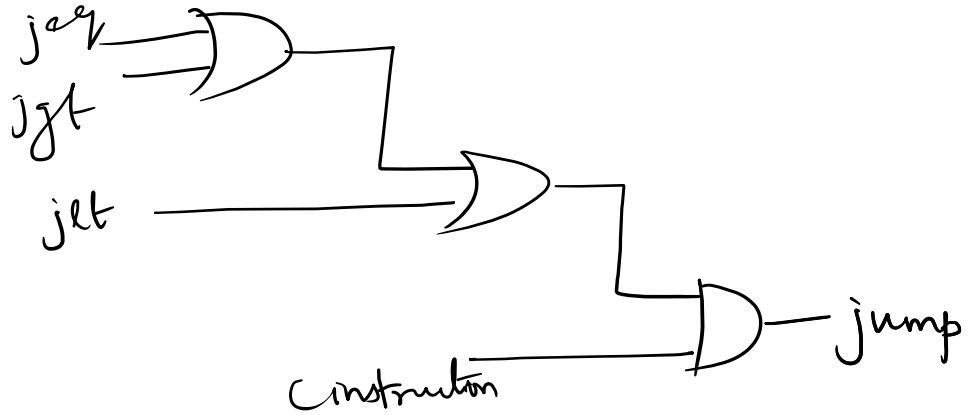
ng → Indicates o/p is -ve

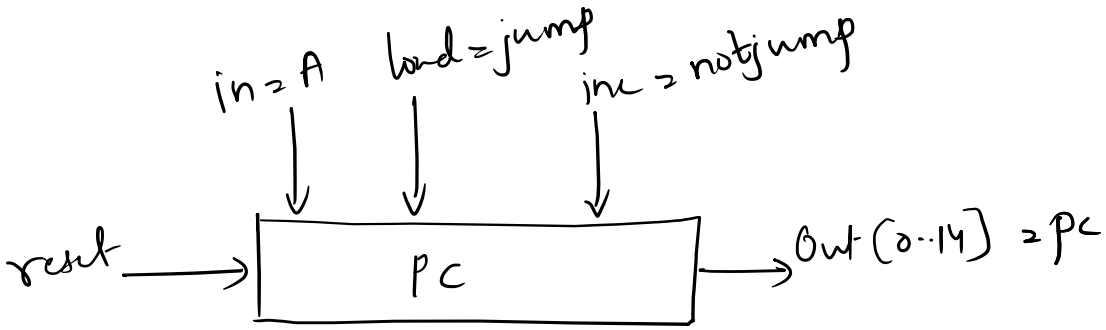




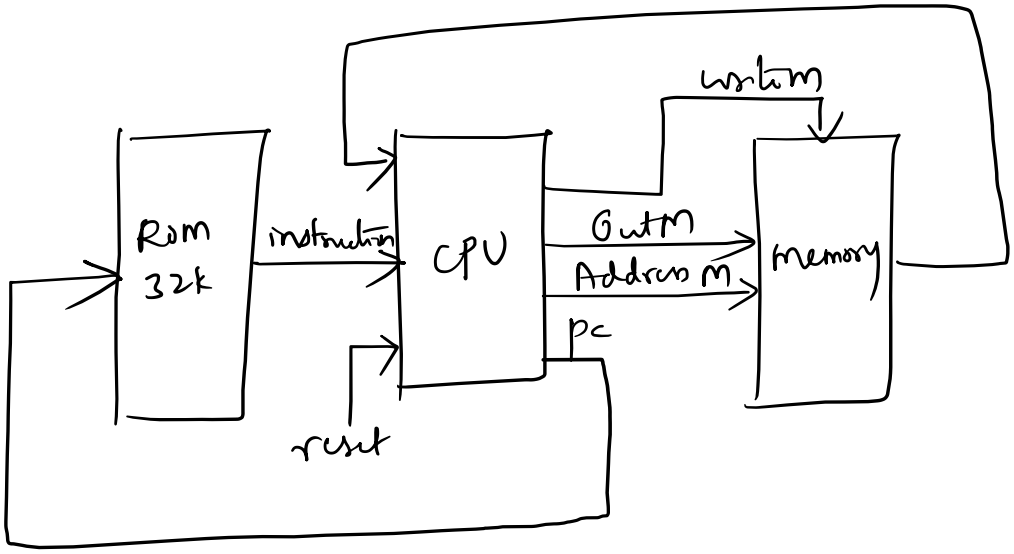


Jump if jeq or jgt or jlt is 1 & it is a C instruction


jeq
jgt
jlt
instruction
jump


in = A
load = jump
inc = notjump
reset
PC
Out (0..14) = pc

③ Computer


writeM
ROM
32k
instruction
CPU
outM
addressM
memory
pc
reset