Project 5

COMPUTER ARCHITECTURE

In this section, we will combine all the pieces & build a CPU, memory section & combine both to create a computer.

To perform any action, a computer must perform the fetch & execute cycle.

① Put the location of next instruction in the memory address input

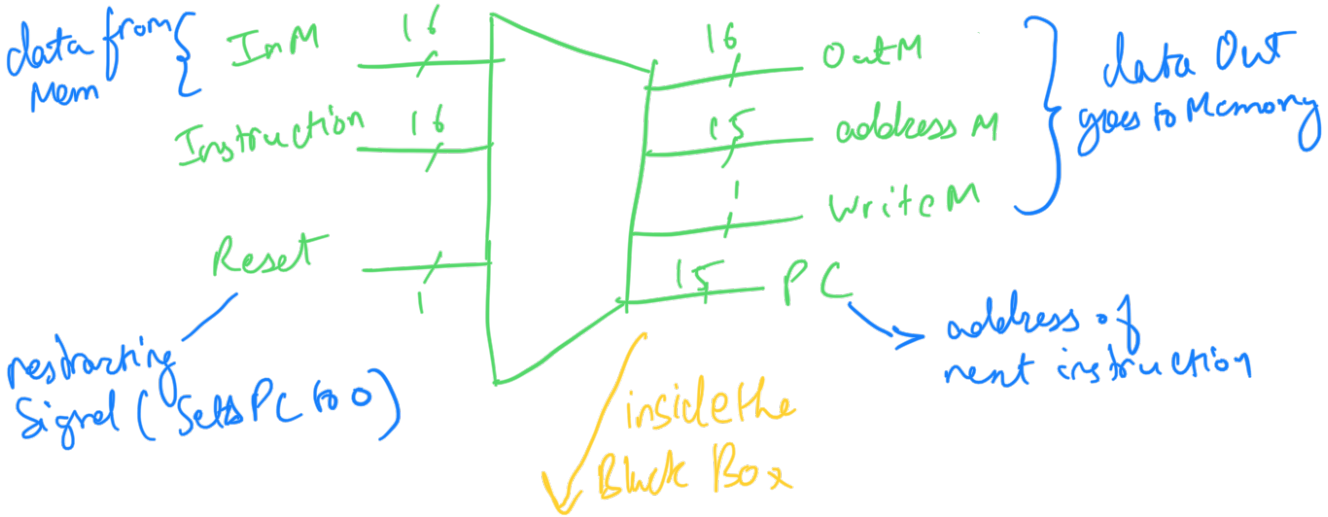
② Get the instruction code by reading the contents at the memory location.

Execute
the instruction specifies which instruction to execute, which memory to access etc.

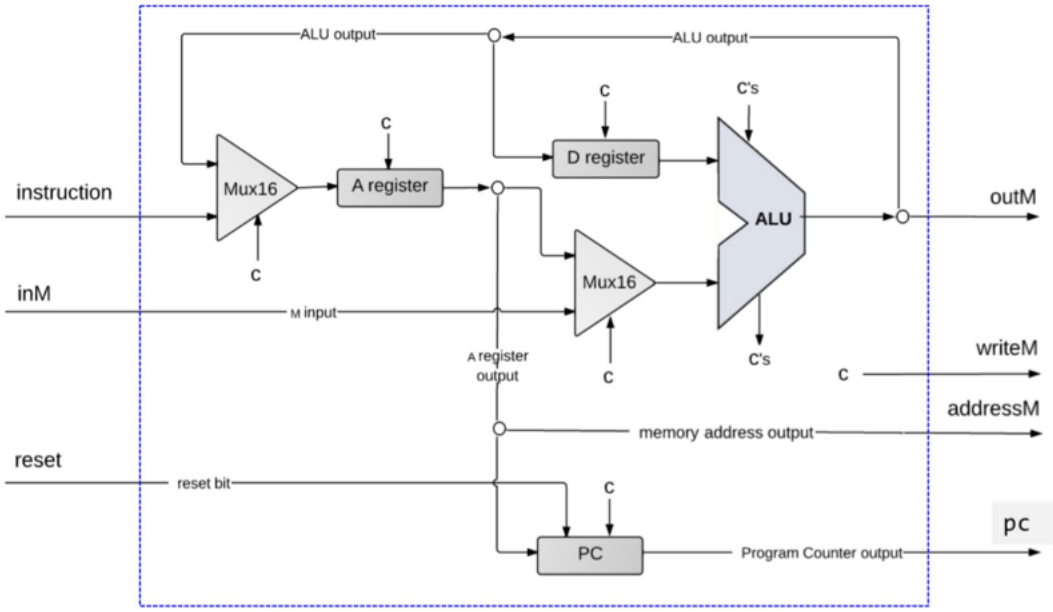
We must decide which one to do as we can only perform either fetch or execute. We do this using a multiplexor

---

HACK CPU



Hack CPU Implementation



(each "C" symbol represents a control bit)

C-instruction specification

ALU C's

C for A register

C for D register

C for Memory

to choose A or M

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

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

Op Code

jump Conditions

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

Examples:	Symbolic:	Binary:
	MD=D+1	1110011111011000

The above diagram & table would help you a lot to write the HDL code. The code with explanation is presented below

```
CHIP CPU {

    IN  inM[16],         // M value input  (M = contents of RAM[A])
        instruction[16], // Instruction for execution
        reset;           // Signals whether to re-start the current
                         // program (reset==1) or continue executing
                         // the current program (reset==0).

    OUT outM[16],        // M value output
        writeM,          // Write to M?
        addressM[15],    // Address in data memory (of M)
        pc[15];          // address of next instruction

    PARTS:
    // Put your code here:

        //A REGISTER
        Mux16(a=instruction, b=ALUout1, sel=instruction[15],out=Ain); //IF C INSTRUCITON, SET ALU OUTPUT AS INPUT FOR A REGISTER OTHERWISE USE INSTRUCITON AS A VALUE AND IN PUT FOR A
        Mux(a=true,b=instruction[5],sel=instruction[15],out=loadA); //IF C INSTRUCITON USE 'd1' TO DECIDE LOAD FOR A REGISTER, IF A INSTRUCTION LOAD IS TRUE
        ARegister(in=Ain,load=loadA,out=Aout1,out[0..14]=addressM); // A REGISTER


        //D REGISTER
        Mux(a=false,b=instruction[4],sel=instruction[15],out=loadD);  //IF C INSTRUCTION USE 'd2' TO DECIDE THE LOAD FOR D, ELSE LOAD IS FALSE
        DRegister(in=ALUout1,load=loadD,out=Dout); //D REGISTER


        // A OR M
        Mux16(a=Aout1, b=inM, sel=instruction[12],out=AMuxout); //USE 'a'. IF 0, USE A REGISTER ELSE USE 'inM'


        //ALU
        ALU(x=Dout,y=AMuxout,zx=instruction[11],nx=instruction[10],zy=instruction[9],ny=instruction[8],f=instruction[7],no=instruction[6],out=ALUout1,out=outM,zr=zr,ng=ng,out[15]=ALUout2);


        //WRITE M
        Mux(a=false,b=true,sel=instruction[3],out=tempwriteM);  //USE 'd3'. IF 0 OUTPUT 0 ELSE 1, THIS OUTPUTS A TEMP BECUASE WE ARE NOT SURE YET
        Mux(a=false,b=tempwriteM,sel=instruction[15],out=writeM);  //IF INSTRUCTION A WRITE M IS FALSE ELSE ITS TEMPWRITEM - WHATEVER THAT HAPPENS TO BE FROM ABOVE OUTPUT


        // PROGRAM COUNTER
        ////////// ALL JUMP CONDITIONS

        Or16Way(in=ALUout1,out=JNE);  //OUTPUTS 0 IS ALL 16 BITS OF ALUOUT IS 0 ELSE 1 [JNE]
        Not(in=JNE,out=JEQ);          // OUTPUTS NOT OF JNE, NOT OF JNE IS [JEQ]
        DMux(in=true,sel=ALUout2,a=JGTtemp,b=JLT);  //ALUout2 IS THE 16th BIT. IF ITS 0, OUTPUT IS 0,1, WHICH MEANS JGT MAYBE 1 (NOT SURE YET) AND JLT IS 0. ELSE JGT IS 0 AND JLT IS 1
        And(a=JGTtemp,b=JNE,out=JGT);           // IF JGTtemp AND JNE BOTH ARE TRUE, IT MEANS JGT IS 1 i.e. ALUOUT IS > 0
        Or(a=JGT,b=JEQ,out=JGE);                // THIS CHECKS FOR >= OR JGE
        Or(a=JLT,b=JEQ,out=JLE);                // THIS CHECKS FOR <= OR JLE

        Mux8Way(a=false,b=JGT,c=JEQ,d=JGE,e=JLT,f=JNE,g=JLE,h=true,sel=instruction[0..2],out=jmptemp); //CHOOSING BETWEEN 8 JUMP CONDITIONS. I CREATED Mux8Way FOR THIS. OUT IS STILL A TEMP BCOZ WE ARE NOT SURE

        /////////

        Mux(a=false,b=jmptemp,sel=instruction[15],out=jmp); //IF C INSTRUCTION USE jmptemp ELSE JMP IS FALSE

        Mux16(a=Aout1,b=false,sel=reset,out=pcin);      //IF RESET IS 0 USE A REGISTER OUTPUT ELSE 0 FOR PC INPUT
        Mux(a=jmp,b=true,sel=reset,out=pcload);         // IF RESET IS 0 USE JMP OR JMP IS NECESSARY. SET OUTPUT TO PCLOAD

        PC(in=pcin,load=pcload,inc=true,out[0..14]=pc);     //PROGRAM COUNTER

}
```

if the above image is not readable, see code in the project.

Key Points (to help you write code for task)

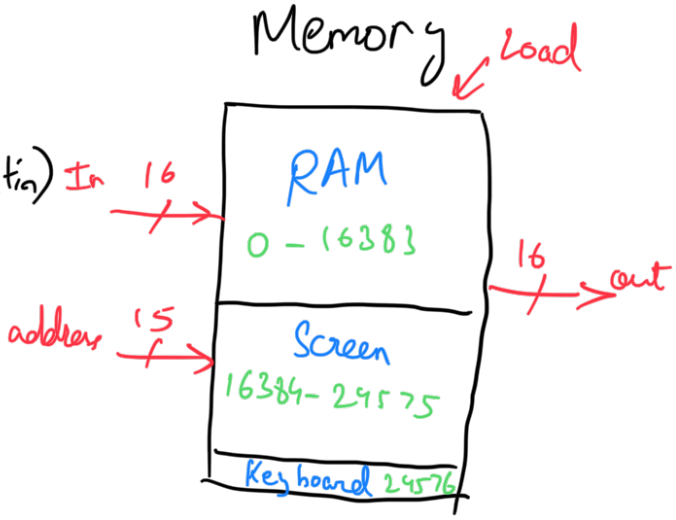
* Check for C & A instruction using MUx for all registers, PC
* arrange C's for ALU according to ALU & CPU implementation
* write code for 8 jump conditions & choose one using MUX
* when reset is 1, we don't care about anything set PC in to 0 & load to 1. else if reset is 0, take A register as input & check for instructions.

You should be able to code this yourself.

MEMORY

In this we need to use RAM16K, Screen (Built in) & Keyboard (Built in)

Our job is to create input to correct address.



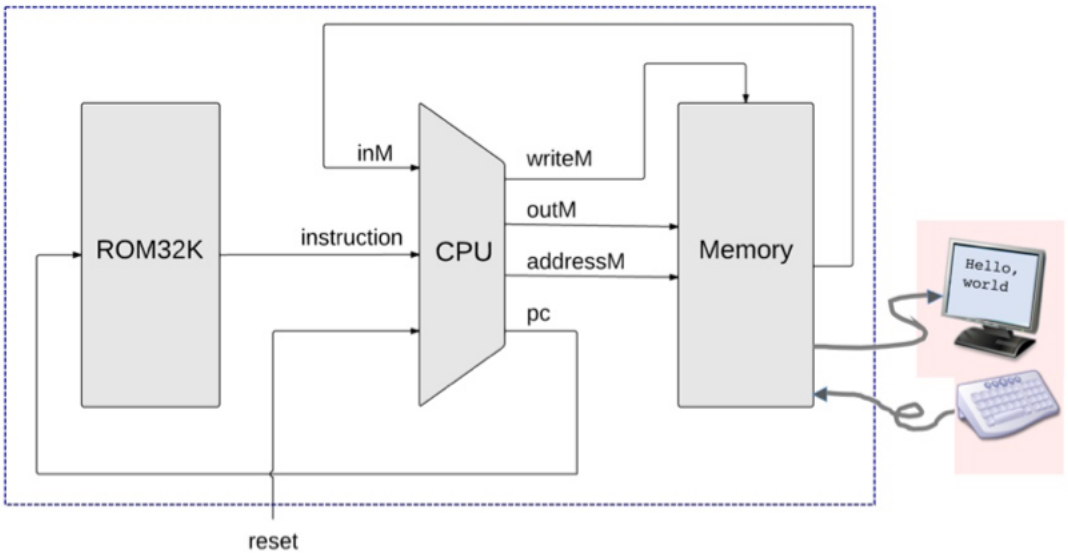
```
CHIP Memory {
    IN in[16], load, address[15];
    OUT out[16];

    PARTS:

        Keyboard(out=tempout1);   //KEYBOARD INPUT
        Or16Way(in=tempout1,out=k);  //IF INPUT(16) HAS ANY ONE'S OUTPUT WILL BE 1

        DMux(in=load,sel=address[14],a=a,b=b);  //TO DECIDE BETWEEN SCREEN AND RAM16, WE USE THE LAST BIT IN THE ADDRESS

        RAM16K(in=in,load=a,address=address[0..13],out=ram);   //RAM16
        Screen(in=in,load=b,address=address[0..12],out=screen); //SCREEN

        Mux16(a=ram,b=screen,sel=address[14],out=tempout2); //CHOOSING BETWEEN RAM AND SCREEN

        Mux16(a=tempout2,b=tempout1,sel=k,out=out);  //CHOOSING BETWEEN KEYBOARD OUTPUT AND PREVIOUS OUTPUT. IF KEYBOARD HAS 1, KEYBOARD OUT WILL BE OUTPUTED


}
```


```
CHIP Memory {
    IN in[16], load, address[15];
    OUT out[16];

    PARTS:

        Keyboard(out=tempout1);   //KEYBOARD INPUT
        Or16Way(in=tempout1,out=k);  //IF INPUT(16) HAS ANY ONE'S OUTPUT WILL BE 1

        DMux(in=load,sel=address[14],a=a,b=b);  //TO DECIDE BETWEEN SCREEN AND RAM16, WE USE THE LAST BIT IN THE ADDRESS

        RAM16K(in=in,load=a,address=address[0..13],out=ram);   //RAM16
        Screen(in=in,load=b,address=address[0..12],out=screen); //SCREEN

        Mux16(a=ram,b=screen,sel=address[14],out=tempout2); //CHOOSING BETWEEN RAM AND SCREEN

        Mux16(a=tempout2,b=tempout1,sel=k,out=out);  //CHOOSING BETWEEN KEYBOARD OUTPUT AND PREVIOUS OUTPUT. IF KEYBOARD HAS 1, KEYBOARD OUT WILL BE OUTPUTED


}
```


COMPUTER

Combine ROM, CPU, & Memory to create a Computer. This is a simple task, we have all the pieces.

Hack Computer implementation



That's it!

just take the output of 1 chip & make it the input of another. use the diagram above