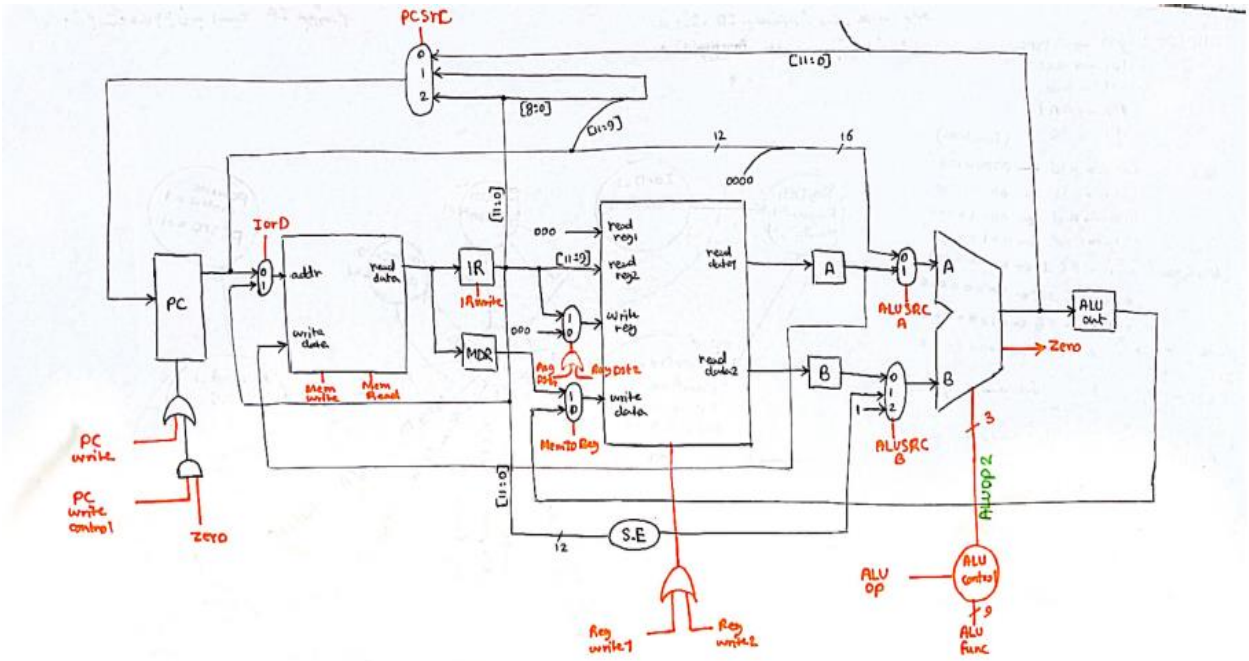
تمرین کامپیوتری شماره چهار معماری کامپیوتر

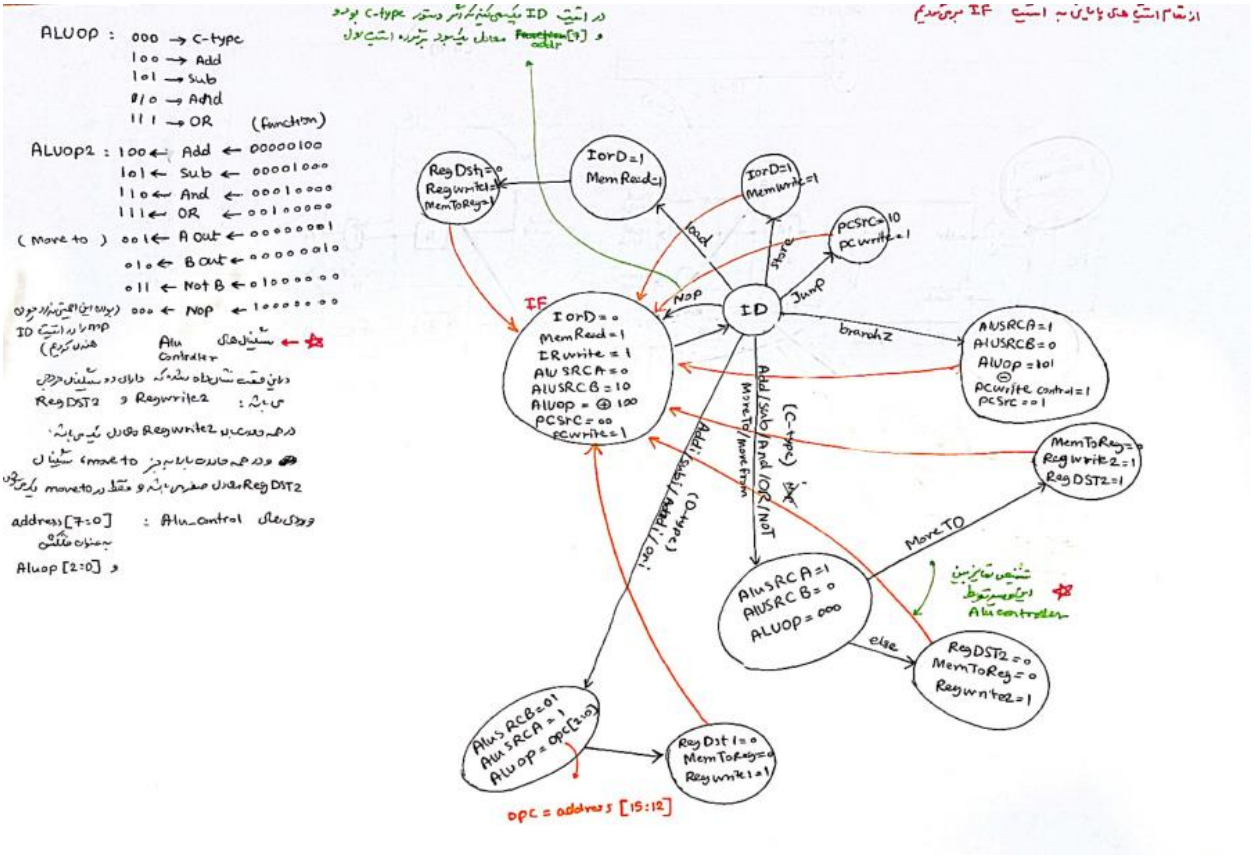
مریم جعفرآبادی آشتیانی 810199549

سنا ساری نوایی 810199435

Datapath:



Controller:



خروجی state های کنترلی نیز به شرح زیر می‌باشند:

	IorD	pcSRC	pcWrite	pcWriteCtrl	MemRead	MemWrite	MemToReg	Irwrite	RegDst1	RegDst2	RegWrite1	RegWrite2	ALUsrc_A	ALUsrc_B	ALUop	Zero_in
IF	0	2'b00	1		1			1					0	2'b10	3'b100(+)	
ID			0		0	0		0								
LOAD1	1				1											
LOAD2							1		0		1					
STORE	1					1										
JUMP		2'b10	1													
BRANCH		2'b01		1									1	2'b00	3'b101(-)	Zero_out
CTYPE													1	2'b00	3'b000	
CTYPE1							0			inst[0]		1				
DTYPE													1	2'b01	opcode[2:0]	
D1							0		0		1					

C++ Code:

```cpp
int main()
{
    vector<int> numbers = {1,2,3,4,5,6,7,8,9,10,20,12,13,14,15,16,17,18,19,11};
    int maxx = numbers[0];
    int index = 0;
    for(int i=0;i<20;i++)
    {
        if(maxx<numbers[i])
        {
            maxx = numbers[i];
            index = i;
        }
    }
}
```

```
@000
0000000000000001
0000000000000010
0000000000000011
0000000000000100
0000000000000101
0000000000000110
0000000000000111
0000000000001000
0000000000001001
0000000000001010
0000000000010100
0000000000001100
0000000000001101
0000000000001110
0000000000001111
0000000000010000
0000000000010001
0000000000010010
0000000000010011
0000000000001011
```

```
//moveto R7 : R7 <- R0
//addi 2048 : R0 = R0 + 2048
//add R0 : R0 = R0+R0
//add R0 : R0 = R0+R0
//add R0 : R0 = R0+R0
//add R0 : R0 = R0+R0
//moveto R5 : R5 = R0(R5=10..00)
//load 0 : R0 = M(0)
//moveto R6 : R6 = R0 (max)
//load 1 : R0 = M(1)
//moveto R1 : R1 = temp (M(1))
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5 (if R0>0: R0=0)
//branchz R4,225 (if R0==0 : R1>R6 swap needed)
//jump 230

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 1 : R0 = 1 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 2 : R0 = M(2)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,236
//jump 241

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 2 : R0 = 2 + R0
//moveto R7 : R7 <- R0 (update_index)
```

```
//load 3 : R0 = M(3)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,247
//jump 252

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 3 : R0 = 3 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 4 : R0 = M(4)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,258
//jump 263

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 4 : R0 = 4 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 5 : R0 = M(5)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,269
//jump 274
```

```
//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 5 : R0 = 5 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 6 : R0 = M(6)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,280
//jump 285

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 6 : R0 = 6 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 7 : R0 = M(7)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,291
//jump 296

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 7 : R0 = 7 + R0
//moveto R7 : R7 <- R0 (update_index)
```

```
//load 8 : R0 = M(8)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,302
//jump 307

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 8 : R0 = 8 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 9 : R0 = M(9)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,313
//jump 318

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 9 : R0 = 9 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 10 : R0 = M(10)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,324
//jump 329
```

```
//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 10 : R0 = 10 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 11 : R0 = M(11)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,335
//jump 340

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 11 : R0 = 11 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 12 : R0 = M(12)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,346
//jump 351

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 12 : R0 = 12 + R0
//moveto R7 : R7 <- R0 (update_index)
```

```
//load 13 : R0 = M(13)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,357
//jump 362

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 13 : R0 = 13 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 14 : R0 = M(14)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,368
//jump 373

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 14 : R0 = 14 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 15 : R0 = M(15)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,379
//jump 384
```

```
//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 15 : R0 = 15 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 16 : R0 = M(16)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,390
//jump 395

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 16 : R0 = 16 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 17 : R0 = M(17)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,401
//jump 406

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 17 : R0 = 17 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 18 : R0 = M(18)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,412
//jump 417
```

```
//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 18 : R0 = 18 + R0
//moveto R7 : R7 <- R0 (update_index)

//load 19 : R0 = M(19)
//moveto R1 : R1 <- R0
//sub R6 : R0 = R0 - R6
//and R5 : R0 = R0&R5
//branchz R4,423
//jump 428

//movefrom R1 : R0 <- R1
//moveto R6 : R6 <- R0 (update max)
//movefrom R4 : R0 <- R4=0
//addi 19 : R0 = 19 + R0
//moveto R7 : R7 <- R0 (update_index)

//movefrom R6
//store 200
//movefrom R7
//store 204
```

عناصر آرایه از 1 تا 20 انتخاب شده اند. بزرگترین عدد رقم 20 می‌باشد که index مربوط به آن، 10 است.

data	1	2	3	4	5	6	7	8	9	10	20	12	13	14	15	16	17	18	19	11
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

شکل موج خروجی نیز مانند زیر است. همانطور که مشاهده می‌شود، بزرگترین عدد رقم 20 و مقدار index آن نیز 10 می‌باشد که با داده های ما یکسان است.