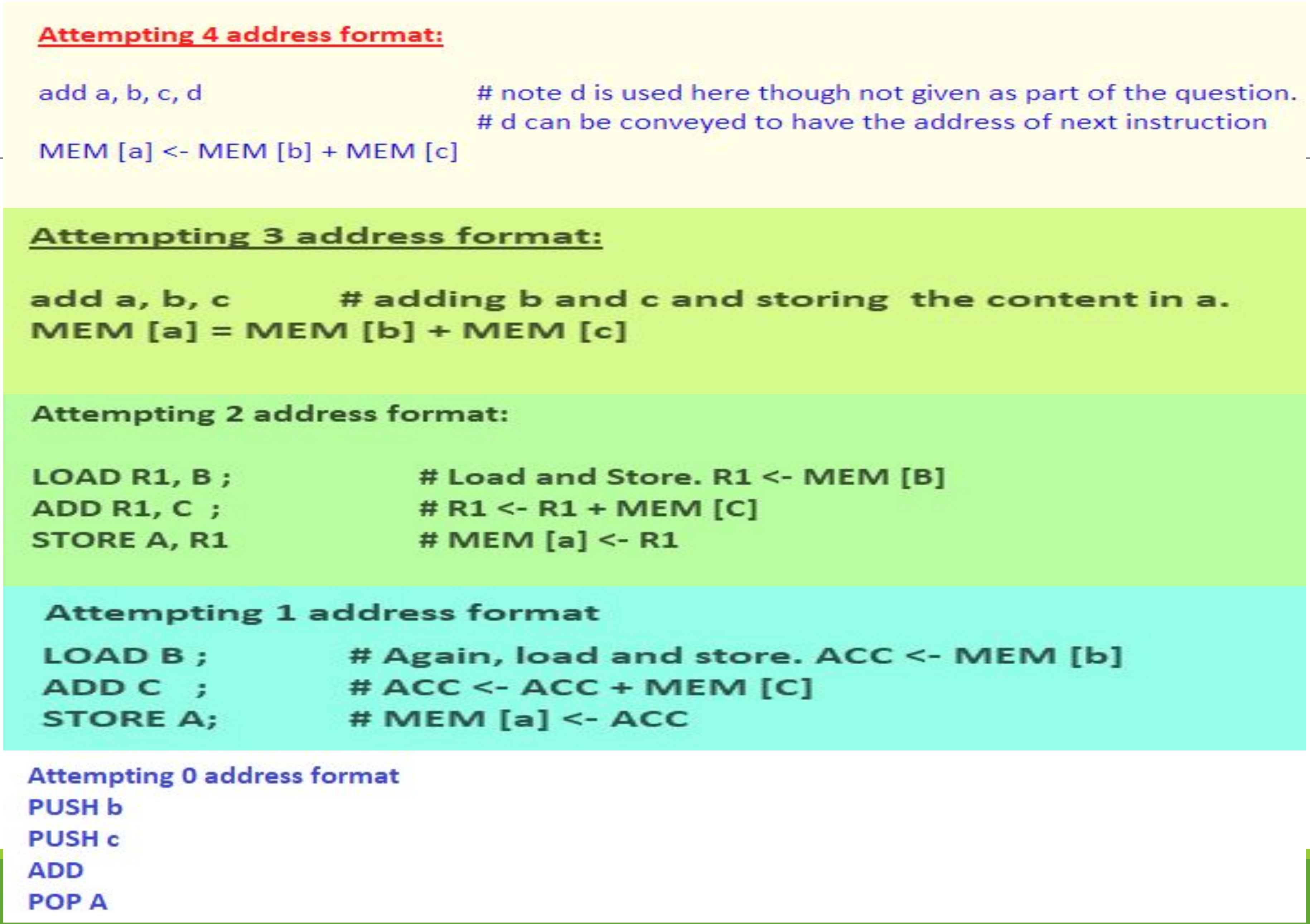
Computer Organization and Architecture – Let’s solve these.

SHRIRAM K VASUDEVAN

SESSION – 15

Q.No. 1

Write the following expression using 4 address format, 3address format, 2 address format,1 address format and 0 address format. a = b + c ; # this is the question folks.



Attempting 4 address format:

```
add a, b, c, d                # note d is used here though not given as part of the question.
                              # d can be conveyed to have the address of next instruction
MEM [a] <- MEM [b] + MEM [c]
```

Attempting 3 address format:

```
add a, b, c          # adding b and c and storing  the content in a.
MEM [a] = MEM [b] + MEM [c]
```

Attempting 2 address format:

```
LOAD R1, B ;              # Load and Store. R1 <- MEM [B]
ADD R1, C  ;              # R1 <- R1 + MEM [C]
STORE A, R1               # MEM [a] <- R1
```

Attempting 1 address format

```
LOAD B ;           # Again, load and store. ACC <- MEM [b]
ADD C   ;          # ACC <- ACC + MEM [C]
STORE A;           # MEM [a] <- ACC
```

Attempting 0 address format

```
PUSH b
PUSH c
ADD
POP A
```

You work this out!

Write the following expression using 4 address format, 3address format, 2 address format,1 address format and 0 address format x=(y+d)/(d+3)

- You folks try this and get me answer.

Q.NO. 2

Give the register transfer notations for the following MIPS Instructions. (It could be asked as RTN or RTL also)

LW $t2,100($s2)	$t2<-mem[100+$s2]
SW $t3,32($s2)	mem[32+$s2]<-$t3
beq $t1,$t3,8	if $t1==$t3 goto (pc+4)+8
addi $t1,$t2,32	$t1<-$t2+32

Q.NO. 3

For the following MIPS INSTRUCTIONS give the RTL equivalents, addressing modes.

or $s2,$t1,$t2	$s2<-$t1 OR $t2	-	Register addressing mode
sw $t3,12($t0)	mem[12+$t0]<- $t3	-	Base/displacement addressing

registers $s0 to $s7 map onto registers 16 to 23, and **registers $t0 to $t7 map onto registers 8 to 15.**

Q.NO. 4

Decode the instruction $(010f8020)_{16}$

0000 0001 0000 1111 1000 0000 0010 0000 = 32 bits (Length)

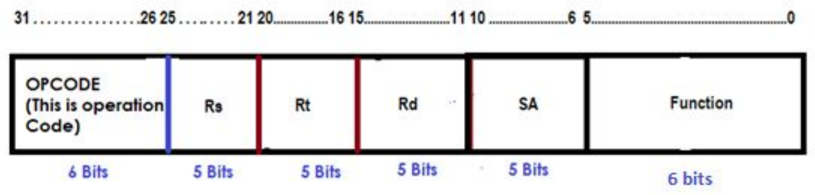
Let us decode:

0000 0001 0000 1111 1000 0000 0010 0000

Ans: add $16, $8, $15 == add $s0, $t0, $t7

Legend:

- **Red – Opcode**
- **Green – Rs**
- **Purple – Rt**
- **Maroon – Rd**
- **Light Blue – SA**
- **Black – Function code**



MIPS supports 3 address instructions as we know. **(2 Source Registers and 1 destination register)**

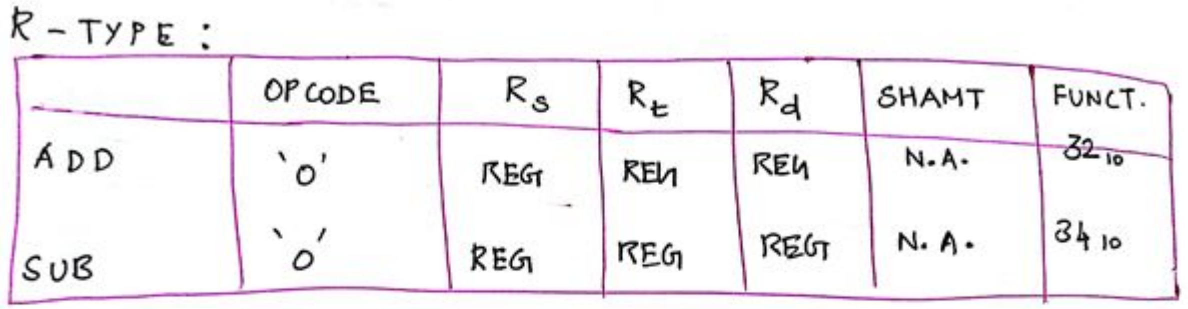
Simple Example: Add instruction to be referred.

- add $s1, $s1, $s2

rd rs1 rs2

Opcode - Operation Code
Rs - Source Register
Rt - Can be used as Source / Destination.
Rd - Destination
Function code - 6 bits. Specifies function.
SA - Shift amount.

R – TYPE :



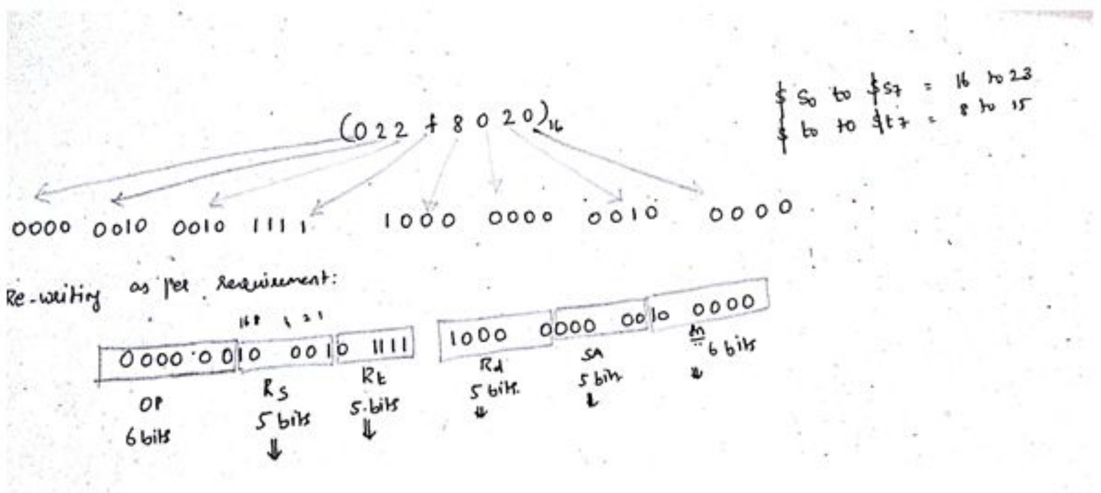
	OPCODE	R_s	R_t	R_d	SHAMT	FUNCT.
ADD	'0'	REG	REG	REG	N.A.	32_{10}
SUB	'0'	REG	REG	REG	N.A.	34_{10}

registers $s0 to $s7 map onto registers 16 to 23, and **registers** $t0 to $t7 map onto registers 8 to 15.

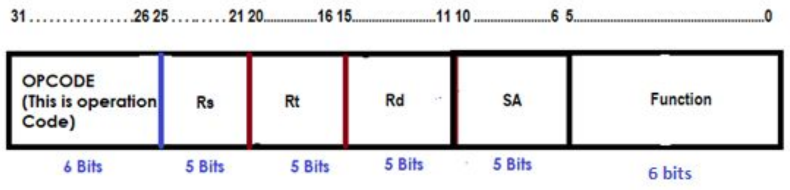
Q. NO. 5

Decode it yourself.

Decode the instruction $(022f8020)_{16}$



ADD $16, $17, $15 = ADD $s0, $s1, $t7



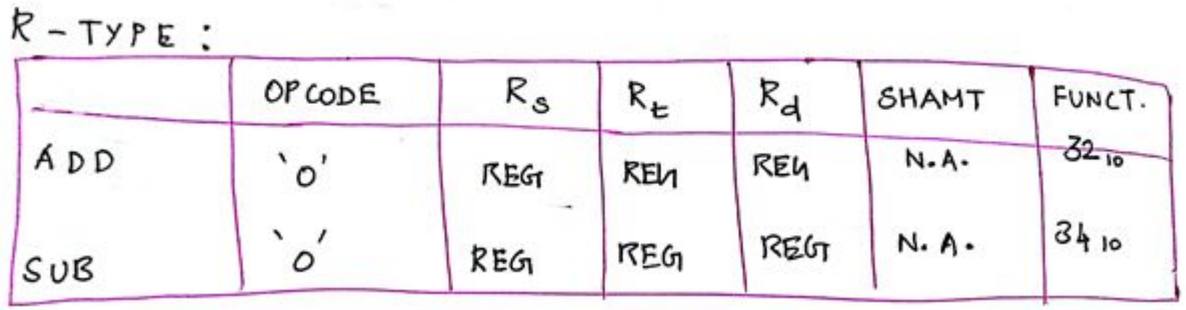
MIPS supports 3 address instructions as we know. **(2 Source Registers and 1 destination register)**

Simple Example: Add instruction to be referred.

- **add $s1, $s1, $s2**
  rd rs1 rs2

Opcode - Operation Code
Rs - Source Register
Rt - Can be used as Source / Destination.
Rd - Destination
Function code - 6 bits. Specifies function.
SA - Shift amount.

R - TYPE :



	OPCODE	R_s	R_t	R_d	SHAMT	FUNCT.
ADD	'0'	REG	REG	REG	N.A.	32_{10}
SUB	'0'	REG	REG	REG	N.A.	34_{10}

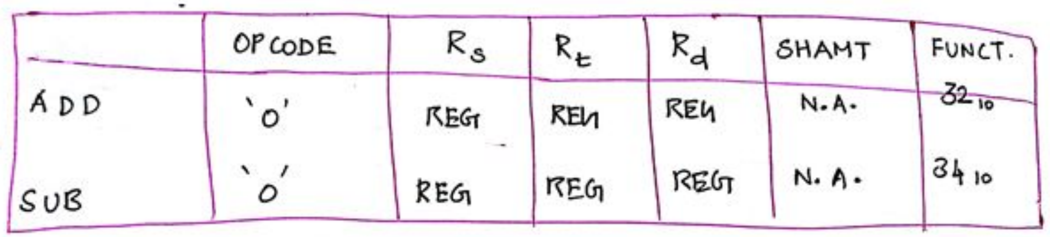
Q. NO. 6

Encode the following instructions in binary. Also write the RTL for each instruction

add $t1,$s0,$s1 **$t1<-$s0+$s1**

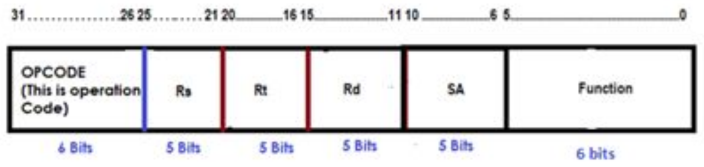
---

registers $s0 to $s7 map onto registers 16 to 23, and **registers** $t0 to $t7 map onto registers 8 to 15.



	OPCODE	R_s	R_t	R_d	SHAMT	FUNCT.
ADD	'0'	REG	REG	REG	N.A.	32_{10}
SUB	'0'	REG	REG	REG	N.A.	34_{10}

add $t1,$s0,$s1

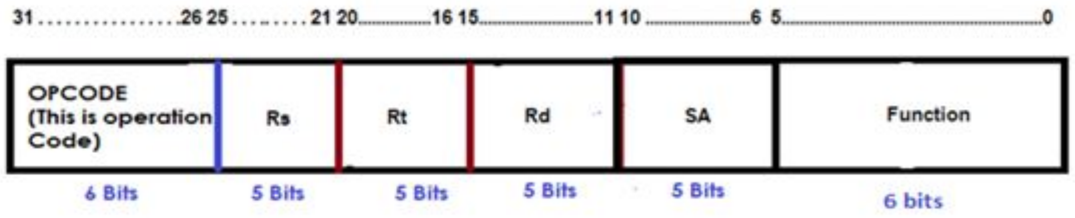


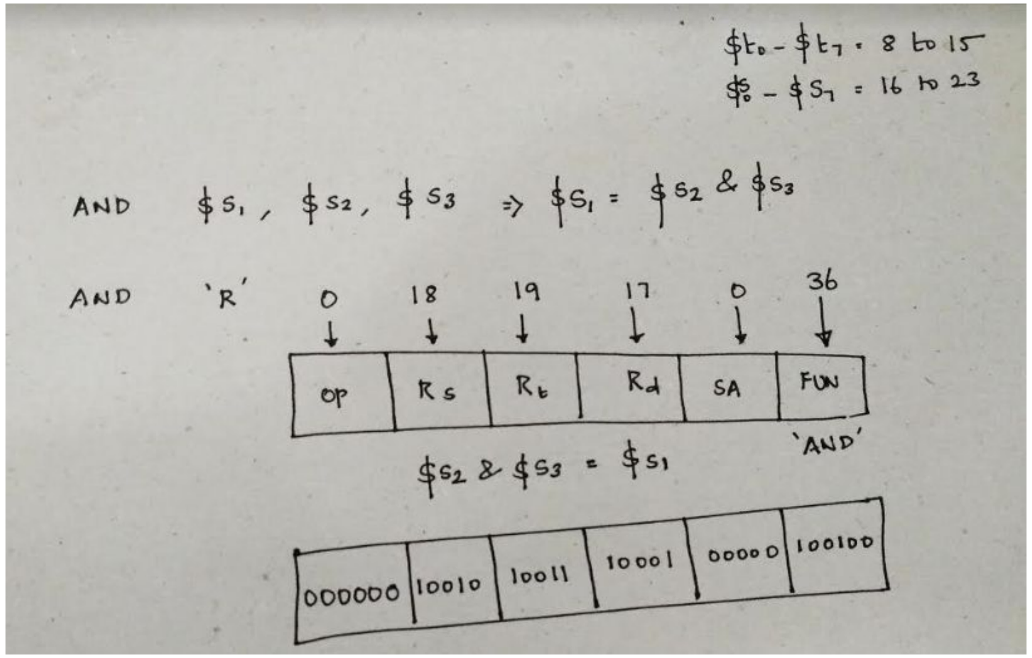
OPCODE	Rs	Rt	Rd	SA	Function
000000	10000	10001	01001	00000	100000

Q. NO. 7

Encode the following instructions in binary. Also write the RTL for the instruction presented.

and $s1,$t1,$t2 **RTL : $s1 <-reg $t1 & reg $t2**





Q.No.8

Mention the status of control signals for the below operation

ADD $s1, $s2, $s3

RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOP 1	ALUOP 2
1	0	0	1	0	0	0	1	0

Q.No. 9

Mention the status of the control signals for the below instruction

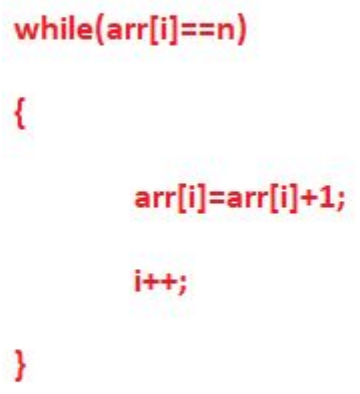
LW $t0, 32($t1)

RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOP 1	ALUOP 2
0	1	1	1	1	0	0	0	0

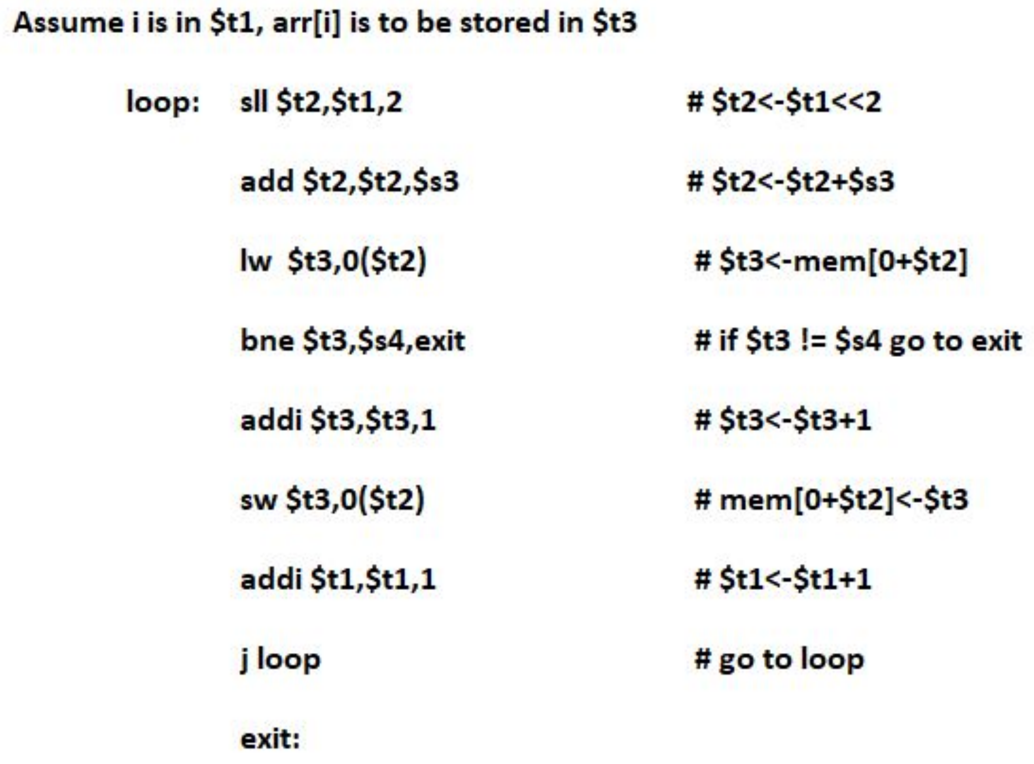
For you to solve!
Find the status of control signals for any SW operation.

Q. No. 10

Write the MIPS equivalent for the following C Code.



```c
while(arr[i]==n)
{
        arr[i]=arr[i]+1;
        i++;
}
```



Assume i is in $t1, arr[i] is to be stored in $t3

```
loop:   sll $t2,$t1,2           # $t2<-$t1<<2
        add $t2,$t2,$s3         # $t2<-$t2+$s3
        lw  $t3,0($t2)          # $t3<-mem[0+$t2]
        bne $t3,$s4,exit        # if $t3 != $s4 go to exit
        addi $t3,$t3,1          # $t3<-$t3+1
        sw $t3,0($t2)           # mem[0+$t2]<-$t3
        addi $t1,$t1,1          # $t1<-$t1+1
        j loop                  # go to loop
        exit:
```

Q. NO. 10

You work this out.

Write the MIPS equivalent for the following C Code.

1. a=(b-d)+2;

2. while(a!=b)

```
{
  c=c+a;
  a++;
}
```