Surprise Quiz 3

Q1. Consider a MIPS instruction that aims to perform a certain calculation using an immediate value(12 bits).

```
addi $t0, $t0, i
```

Where **i** is the immediate value, the initial value of t0 is **0x 0000 0060**. What should be the value of i (in hexadecimal) such that the value of t0 becomes **0x 0000 0B51** ?

(Note: If your answer is 0xABCD enter ABCD as the answer and use upper case.)

Ans: AF1

Solution:

i = 0x 0000 0B51 – 0x 0000 0060 = 0x AF1 (12 bits)

Q2. Consider the MIPS code snippet, initial value of **$t0** is **128** and **$s0** is **0.** All the numbers are in decimal format (*Base-10*).

```
1  label:
2        xor $s0, $s0, $t0
3        srl $t0, $t0, 1
4        bgtz $t0, label
5        j exit
6
7  exit:
8
```

What is the value of **$s0** after jumping to **exit**?
Note: srl -> shift right logical, bgtz -> branch on greater than zero, j ->jump
Note : Write the answer in Decimal (Base-10) number format.

Ans: 255

Solution:

Line No.	Operation	t0	s0
Initially		**128**	**0**
1.	-		
2.	s0 = s0 xor t0	128	128
3.	Shift content of t0 right by 1 bit	64	128
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	64	192
3.	Shift content of t0 right by 1 bit	32	192
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	32	224
3.	Shift content of t0 right by 1 bit	16	224
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	16	240
3.	Shift content of t0 right by 1 bit	8	240
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	8	248

3.	Shift content of t0 right by 1 bit	4	248
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	4	252
3.	Shift content of t0 right by 1 bit	2	252
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	2	254
3.	Shift content of t0 right by 1 bit	1	254
4.	t0 > 0 goto label		
1.	-		
2.	s0 = s0 xor t0	1	255
3.	Shift content of t0 right by 1 bit	0	255
4.	t0 = 0 goto line 5		
5.	Jump to exit	0	255
7.	-	0	**255**

Final value of **s0** is **255.**

Q3. Consider the following MIPS program:

```
1  fun1:
2      bgez $a0, fun2
3      jr $ra
4  fun2:
5      addi $sp, $sp, -4
6      sw $ra, 0($sp)
7      addi $a0, $a0, -1
8      jal fun1
9      lw $ra, 0($sp)
10     mul $v0, $v0, $a1
11     addi $sp, $sp, 4
12     jr $ra
13 main:
14     li $v0, 1
15     li $a0, 2
16     li $a1, 2
17     jal fun1
```

If the program execution begins from main and the initial value of $sp is **0xA210**.
Then what is the least possible value of **$sp (in hexadecimal format)**?
Note: bgez-> branch on greater than or equal to zero, jr jump register, jal->jump and link, sw-> store word, lw->load word.

(Note: If your answer is 0xABCD enter ABCD as the answer and use upper case. If your answer is 0x00AB12 enter AB12 as the answer.)

Ans:_____

Ans: A204

Soln:

Tracing the sequence of execution:

Line No.	**Operation**	**Register Values**	**Call Frame**
14. li $v0, 1	Set register $v0 =	v0 = 1,	main

	2	sp = 0xA210	
15. li $a0, 2	Set register $a0 = 2	v0 = 1. a0 = 2, sp = 0xA210	main
16. li $a1, 2	Set register $a1 = 2	v0 = 1, a0 = 2, a1 = 2, sp = 0xA210	main

17. jal fun1	Jump and link to the label fun. Moves execution to line 2.	v0 = 1, a0 = 2, a1 = 2, sp = 0xA210	$main_{17}$->fun1
2. bgez $a0, fun2	If $a0 >= 0 goto line 5 (fun2). Since $a0 is 2, execution goes to line 5.	v0 = 1, a0 = 2, a1 = 2, sp = 0xA210	$main_{17}$->fun1
5. addi $sp,$sp,-4	Add -4 to register $sp.	v0 = 1, a0 = 2, a1 = 2, sp = 0xA20C	$main_{17}$->fun1
6. sw $ra,0($sp)	Store the value of register $ra in the memory address stored in $sp.	v0 = 1, a0 = 2, a1 = 2, sp = 0xA20C	$main_{17}$->fun1
7. addi $a0,$a0,-1	Add -1 to register $a0	v0 = 1, a0 = 1, a1 = 2, sp = 0xA20C	$main_{17}$->fun1
8. jal fun	Jump and link to	v0 = 1,	$main_{17}$->$fun1_{8}$

	the label fun. Moves execution to line 2.	a0 = 1, a1 = 2, sp = 0xA20C	->fun1
2. bgez $a0, fun2	If $a0 >= 0 goto line 5 (fun2). Since $a0 is 1, execution goes to line 5.	v0 = 1, a0 = 1, a1 = 2, sp = 0xA20C	$main_{17}$->$fun1_{8}$ ->fun1
5. addi $sp,$sp,-4	Add -4 to register $sp.	v0 = 1, a0 = 1, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->fun1
6. sw $ra,0($sp)	Store the value of register $ra in the memory address stored in $sp.	v0 = 1, a0 = 1, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->fun1
7. addi $a0,$a0,-1	Add -1 to register $a0	v0 = 1, a0 = 0, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->fun1
8. jal fun	Jump and link to the label fun. Moves execution to line 2.	v0 = 1, a0 = 0, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$->fun1
2. bgez $a0, fun2	If $a0 >= 0 goto line 5 (fun2). Since $a0 is 0, execution goes to line 5.	v0 = 1, a0 = 0, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$->fun1
5. addi $sp,$sp,-4	Add -4 to register $sp.	v0 = 1, a0 = 0, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$->fun1

6. sw $ra,0($sp)	Store the value of register $ra in the memory address stored in $sp.	v0 = 1, a0 = 0, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$->fun1
7. addi $a0,$a0,-1	Add -1 to register $a0	v0 = 1, a0 = -1, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$->fun1
8. jal fun	Jump and link to the label fun. Moves execution to line 2.	v0 = 1, a0 = -1, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$-> $fun1_{8}$->fun1
2. bgez $a0, fun2	If $a0 >= 0 goto line 5 (fun2). Since $a0 is -1, execution goes to line 3.	v0 = 1, a0 = -1, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$-> $fun1_{8}$->fun1
6. jr $ra	Return execution to previous call frame. Moves execution to line 9.	v0 = 1, a0 = -1, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$-> fun1
9. lw $ra, 0($sp)	Restore return address of previous call frame from stack.	v0 = 1, a0 = -1, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$-> fun1
10. mul $v0, $v0, $a1	Multiply $v0 by $a1 and store in $v0	v0 = 2, a0 = -1, a1 = 2, sp = 0xA204	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$-> fun1
11. addi $sp, $sp, 4	Add 4 to register $sp	v0 = 2, a0 = -1, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->$fun1_{8}$-> fun1

12. jr $ra	Return execution to previous call frame. Moves execution to line 9.	v0 = 2, a0 = -1, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->fun1
9. lw $ra, 0($sp)	Restore return address of previous call frame from stack.	v0 = 2, a0 = -1, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->fun1
10. mul $v0, $v0, $a1	Multiply $v0 by $a1 and store in $v0	v0 = 4, a0 = -1, a1 = 2, sp = 0xA208	$main_{17}$->$fun1_{8}$ ->fun1
11. addi $sp, $sp, 4	Add 4 to register $sp	v0 = 4, a0 = -1, a1 = 2, sp = 0xA20C	$main_{17}$->$fun1_{8}$ ->fun1
12. jr $ra	Return execution to previous call frame. Moves execution to line 9.	v0 = 4, a0 = -1, a1 = 2, sp = 0xA20C	$main_{17}$->fun1
9. lw $ra, 0($sp)	Restore return address of previous call frame from stack.	v0 = 4, a0 = -1, a1 = 2, sp = 0xA20C	$main_{17}$->fun1
10. mul $v0, $v0, $a1	Multiply $v0 by $a1 and store in $v0	v0 = 8, a0 = -1, a1 = 2, sp = 0xA20C	$main_{17}$->fun1
11. addi $sp, $sp, 4	Add 4 to register $sp	v0 = 8, a0 = -1, a1 = 2, sp = 0xA210	$main_{17}$->fun1

12. jr $ra	Return execution to previous call frame. There is no more code line in main, so execution is halted.	v0 = 8, a0 = -1, a1 = 2, sp = 0xA210	main

Minimum value of $sp is 0xA204.