Introduction

Overflow:

Overflow occurs when there are insufficient bits in a binary number representation to portray the result of an arithmetic operation.

Overflow occurs because computer arithmetic is not closed with respect to addition, subtraction, multiplication, or division.

When an Arithmetic operation e.g addition, subtraction, multiplication involves two operands of same sign and the result of the operation is of opposite sign then, we say an overflow has occurred.

This tutorial contains five solved questions on the topic and one assignment question.

Question-1

Q1. Given the instructions below discuss the output .

a) li $s0,-2147483648
li $s1 ,1
add $s0,$s0,$s1

b) li $s0,2147483647
li $s1 ,1
add $s0,$s0,$s1

c) li $s0, 2147483648
li $s1 ,1
add $s0,$s0,$s1

Question-1

b) li $s0,2147483647
li $s1 ,1
add $s0,$s0,$s1

b) signed number addition .
Arithmetic overflow.
Incorrect output is obtained.

Question-1

c)
```
li $s0,2147483648
li $s1 ,1
add $s0,$s0,$s1
```

c) -2147483647.

(No carry, but resulting in wrong output .
Result of the signed addition operation is out of range of the numbers that can be represented with 32 bits.)

Question-2

In what cases can the operation overflow?
In the 'Addition' and 'Subtraction' columns of the following table use a 'Y' to indicate that the operation can overflow, and 'N' to indicate that the operation can not overflow.

Possible cases of overflow for integer addition and subtraction

Operand_1	Operand_2	Addition	Subtraction
Positive[b] Strictly greater than zero	positive		
positive	Negative[c] Strictly lesser than zero		
negative	positive		
negative	negative		
0	positive		
0	negative		
positive	0		
negative	0		

Question-2

In what cases can the operation overflow?
In the 'Addition' and 'Subtraction' columns of the following table use a 'Y' to indicate that the operation can overflow, and 'N' to indicate that the operation can not overflow.

Possible cases of overflow for integer addition and subtraction

Operand_1	Operand_2	Addition	Subtraction (Op2 –op1)
Positive[b] Strictly greater than zero	positive	Y	N
positive	Negative[c] Strictly lesser than zero	N	Y
negative	positive	N	Y
negative	negative	Y	N
0	positive	N	N
0	negative	N	y
positive	0	N	N
negative	0	N	N

Question-3

Given the following pairs of instructions , find the number of bits used to represent the result ,check if overflow occurs for signed or unsigned multiplication.

a)mult Rsrc1, Rsrc2

b)multu Rsrc1, Rsrc2

Question-3

Given the following pairs of instructions , find the number of bits used to represent the result ,check if overflow occurs for signed or unsigned multiplication.

a)mult Rsrc1, Rsrc2

Sol:

a) It is a Native instruction , the multiplication instruction results in the lower order 32 bits to be stored in "lo"" register. The higher order 32 bits of the result to be stored in " Hi" register. 2*n bits are used for representing the product in 2's complement representation . There is no case for overflow because the product never takes more than 2*n bits.

Question-3

b)multu Rsrc1, Rsrc2

b) It is a Native instruction, the unsigned multiplication instruction results in 2 * n bits of product where lower 32 bits are stored in " lo" register and higher 32 bits are stored in "Hi" register.2*n bits are used for representing the unsigned product .There is no case for CARRY.

Question-4

Given the following pairs of instructions , find the number of bits used to represent the result ,check if overflow occurs for signed or unsigned multiplication.

a) li $s0, 1073741824
li $s1 ,2
mul $s0,$s1,$s0

b) li $s0, 1073741824
li $s1 ,4
mul $s0,$s1,$s0

Question-4

a) li $s0, 1073741824
li $s1 ,2
mul $s0,$s1,$s0

a) It is a pseudo instruction. It is a signed number operation. Both Numbers are positive .Only 32 bits of the destination register (s0) are used to represent the result, in this case the result of the operation is out of range of the positive numbers that can be represented with 32 bits .hence there is an overflow generated. The o/p would be -2147483648 which is –ve .hence error.

Question-4

b) li $s0, 1073741824
li $s1 ,4
mul $s0,$s1,$s0

b) It is a pseudo instruction.This is a signed number operation.Only 32 bits of the destination register (s0) are used to represent the result,in this case the result cannot be held in 32 bits hence there is an overflow generated. The o/p i.e s0 register would contain Zero (0) .Hence error.

Question-5

Explain the result of the following set of instructions .

a)
```
li $s0, -2147483648
li $s1 ,1
div $s0,$s1
```

b)
```
lui $s0 ,32768
lui $s1, 1
divu $s0,$s1
```

Question-5

a)
```
li $s0, -2147483648
li $s1 ,1
div $s0,$s1
```

a) This operation is a signed integer division.The result of the division is represented/Held by the "Lo" Register which is where the quotient is stored for division instruction. The actual o/p displayed is -2147483648 for quotient and 0 for remainder. The o/p of the division should be -2147483648 for quotient and 0 for remainder.

No overflow.

Q6, Explain the result of the following set of instructions ?

a)
```
li  $s0, -2147483648
li $s1 , 2147483647
div $s2 $s0,$s1
rem $s3,$s0,$s1
```

b)
```
li  $s0, -2147483648
li $s1 , -2147483647
div $s2 $s0,$s1
rem $s3,$s0,$s1
```

c)
```
lui  $s0, 32768
lui  $s1 ,1
divu $s2 $s0,$s1
remu $s3,$s0,$s1
```

Question-5

b)
```
lui  $s0 ,32768
lui  $s1, 1
divu $s0,$s1
```

b) This is an unsigned division .
The output is
quotient 32768 and remainder 0.
No carry.