2.12 Assume that registers $s0 and $s1 hold the values 0x80000000 and

0xD0000000, respectively.

2.12.1 What is the value of $t0 for the following assembly code?

add $t0, $s0, $s1

1000 0000 0000 0000 0000 0000 0000 0000

+1101 0000 0000 0000 0000 0000 0000 0000

-----------------------------------------------------

1 0101 0000 0000 0000 0000 0000 0000 0000 = 0x50000000

2.12.2

The result is not the desired result and there has been an overflow.

2.12.3

For the contents of registers $s0 and $s1 as specified above,

what is the value of $t0 for the following assembly code?

sub $t0, $s0, $s1

1101 0000 0000 0000 0000 0000 0000 0000 2’s complement

0010 1111 1111 1111 1111 1111 1111 1111

1

------------------------------------------------------

0011 0000 0000 0000 0000 0000 0000 0000

1000 0000 0000 0000 0000 0000 0000 0000

+ 0011 0000 0000 0000 0000 0000 0000 0000

-----------------------------------------------------

1011 0000 0000 0000 0000 0000 0000 0000 = 0xB0000000

2.12.4

This is the desired result and there has not been any overflow.

2.12.5 For the contents of registers $s0 and $s1 as specified above,

what is the value of $t0 for the following assembly code?

add $t0, $s0, $s1

add $t0, $t0, $s0

1000 0000 0000 0000 0000 0000 0000 0000

+1101 0000 0000 0000 0000 0000 0000 0000

-----------------------------------------------------

0101 0000 0000 0000 0000 0000 0000 0000

+1000 0000 0000 0000 0000 0000 0000 0000

-----------------------------------------------------

1101 0000 0000 0000 0000 0000 0000 0000

2.12.6

There was overflow in during the first addition and this is not there desired result.

3.1 What is 5ED4 - 07A4 when these values represent unsigned 16-

bit hexadecimal numbers? The result should be written in hexadecimal. Show your

work.

0101 1110 1101 0100

0000 0111 1010 0100

--------------------------

0101 0111 0011 0000

5 7 3 0

3.12

Using a table similar to that shown in Figure 3.6, calculate the product of the octal unsigned 6-bit integers 62 and 12 using the hardware described in Figure 3.3. You should show the contents of each register on each step.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Step | Multiplier | Multiplicand | Product |
| 0 | 1.No operation  2. SL mcand  3. SR multiplier | 001 010  001 010  000 101 | 000 110 010  001 100 100  001 100 100 | 000 000 000  000 000 000  000 000 000 |
| 1 | 1a.prod=prod+mcand  2. SL mcand  3. SR multiplier | 000 101  000 101  000 010 | 001 100 100  011 001 000  011 001 000 | 001 100 100  001 100 100  001 100 100 |
| 2 | 1.No operation  2. SL mcand  3. SR multiplier | 000 010  000 010  000 001 | 011 001 000  110 010 000  110 010 000 | 001 100 100  001 100 100  001 100 100 |
| 3 | 1.prod=prod+mcand  2.SL mcand  3.SR multiplier | 000 001  000 001  000 000 | 110 010 000  100 100 000  100 100 000 | 111 110 100  111 110 100  111 110 100 |
| 4 | 1.no operation  2.SL mcand  3.SR multiplier | 000 000  000 000  000 000 | 000 100 000  000 000 000  000 000 000 | 111 110 100  111 110 100  111 110 100  7 6 4 |

3.21 If the bit pattern 0×0C000000 is placed into the Instruction Register, what MIPS instruction will be executed?

0000 1100 0000 0000 0000 0000 0000 0000

The instruction is jal 0. The register $ra would become pc + 8 and the new pc would become 0x00000000.

3.22 What decimal number does the bit pattern 0×0C000000 represent if it is a floating point number? Use the IEEE 754 standard.

0000 1100 0000 0000 0000 0000 0000 0000

Exponent = 0001 1000 = 24 –bias = 24-127 = -103

(-1^0) \* (1+0) \* (2^-103)

Given the following 32-bit binary sequences representing single precision IEEE 754 floating point numbers

 a = 0100 0110 1101 1000 0000 0000 0000 0000

 b = 1011 1110 1110 0000 0000 0000 0000 0000

a)     a + b

a = (-1^0) \* (1+.1011) \*2^2 = 1.1011\*(2^14)

b = (-1^1) \* (1+.11) \* (2^-2) = -1.11\*(2^-2)

=-.000000000000000111\*(2^14)

1.101100000000000000

- .000000000000000111

---------------------------------

1.101011111111111001

b)     a × b

exponent = 14 -2 = 12

1.1011

1.11

----------

11011

11011

11011

------------

10111101

=10.111101 \* 2^12 = 1.0111101 \* 2^13