10/05/2023

Noah Semashkewich

COSC 211 Lab 3 Written

i) 2.8: 0xabcdef12

$$0xabcdef12 = (10\times16^7)+(11\times10^6)+(12\times16^5) + (13\times16^4)+(14\times16^3)+(15\times16^2)+(1\times16^1) + (2\times16^0)$$

$$\boxed{=(2,882,400,018)_{10}}$$

ii) 2.12.1: \$s0 = 0x80000000, \$s1 = 0xD0000000

\$s1 = 1000
\$s0 = 1101

```
   1101
 + 1000
  11101  ← overflow
```

= \$150000000, which when converted back to decimal is 90,316,602,816. This value is too large for MIPS, $(2^{31}-1)$, hence an overflow.

iii) 2.12.2:

The value in \$t0 is <u>not</u> the desired result, so an overflow has occured.

iv) 2.12.3: sub \$t0, \$s0, \$s1

0x80000000 − 0xD0000000 = 0xFF000000

$= -1(2^{32}) = -4,294,967,296$

$$\boxed{\$t0 = -4,294,967,296}$$

v) 2.12.4: There has not been an overflow. Min value is -2,147,483,648, and the result falls in the range for a signed integer.

vi) 2.12.5: add $t0, $s0, $s1
add $t0, $t0, $s1

From earlier question, the binary result of $t0 is

0101 0000 0000 0000 0000 0000 0000 0000
which resulted in an over flow
so adding $s0

$s0 = 0101 0000 0000 0000 0000 0000 0000 0000

The second instruction $= 1101\ \underbrace{0000}_{\times 7} \ldots\ldots$

overflow has already occured and this is not desired result.

vii) 2.12.6: This is not the desired result because overflow occurs in the first instruction

viii) 2.14:
0000 0010 0001 0001 1000 0000 0010 0000

So...

$\underbrace{000000}_{\text{opcode (empty)}}, \underbrace{10000}_{\text{rs (\$s0)}} \underbrace{10000}_{\text{rt (\$s0)}}, \underbrace{10000}_{\text{rd (\$s0)}}, \underbrace{00000}_{\text{shamt (unused)}}, \underbrace{100000}_{\text{funct (add, via greensheet)}}$

So the instruction is add $\boxed{\$s0, \$s0, \$s0 \text{ (R-Type)}}$

ix) 2.15: sw \$t1, 32(\$t2)

$\underbrace{\text{sw}}_{\substack{101011 \\ \text{(opcode for sw)}}}, \underbrace{\text{base (\$t2)}}_{\substack{01010 \\ \text{(10, decimal)}}} \quad \text{rt (\$t1)}, \quad \underline{\text{offset}} \text{ (32)}$

01001 (9, decimal)   0000000000010000

So Binary equivalent is

$\boxed{1010\ 1101\ 0100\ 1001\ 0000\ 0000\ 00010000}$

x) 2.19.1: \$t0 = 0xAAAAAAAA, \$t1 = 0x12345678

① Register \$t0 will be shifted left by 44, but the sll instruction will give an error because MIPS registers are 32 bit, so not possible.

② The or \$t2, \$t2, \$t1 will be a bitwise operation. Because the previous instruction failed, \$t2 has no value, so \$t2 is equal to \$t1

$\boxed{\$t2 = 0x12345678}$

xi) 2.19.2:

sll \$t2, \$t0, 4

- This instruction shifts the bits in \$t0 left by 4 bits.

0xAAAAAAAA as binary : 10101010101010101010101010101010

So after sll , we get - 10101010101010101010101010100000

$\boxed{\$t2 = 0xAAAAAAA0}$ (signed two's complement)

andi \$t2, \$t2, -1

\$t2 will bitwise AND -1's value.

So the answer will be same as previous instruction

$\boxed{\$t2 = 0xAAAAAAA0}$ = $\boxed{-1,431,655,776}$

xii) 2.19.3:

srl \$t2, \$t0, 4

- shifts the bits in \$t0 right by 3 bits

\$t0 = 0xAAAAAAAA will become 0x15555555

andi \$t2, \$t2, 0xFFEF → Andi operation

$\boxed{\$t2 = 0x00005545}$