COMP2421 Computer Organization Homework 1

ZHOU Siyu Feb. 19

Part 1

1.1 Answer: 103

X in base 10: $X_{10} = 1 \times 6^2 + 2 \times 6^1 + 4\ \times 6^0 = 52$

X in base 7: $X_7 = \mathbf{103_7}$

1.2

1) Answer: 1 1110 1101

19 = 0 0001 0011   -19 = 1 1110 1100 (flip) +1 = **1 1110 1101**

2) Answer: -87

1 1010 1001 – 1 = 1 1010 1000   Flip: 0 0101 0111 = **87**

1.3

1) Answer: 0110 (in 2's complement form), no overflow

0111 + 1111 = 1 0110 → 7+(-1) = 6

No overflow, because positive + negative = positive is possible here, then we ignore the 1.

2) Answer: 1 0110(in 2's complement), overflow

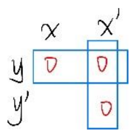
1110 + 1000 = 1 0110 → (-2) + (-8) != 6   **Original result:** 1 0110 = -10

Overflow, because negative + negative = positive is not possible, and we need to consider 1 in the front.

1.4 Answer: address B(0x00FA0700) can be used.

Because 0x0000000 to 0x00400000 is reserved. So memory adress A is not possible to used as the address for the instruction. But address B is not reserved, so address B(0x00FA0700) can be used as the address for an instruction.

1.5 Answer: $f = x' + y$



$f = x'y' + xy + x'y = x'y' + xy + x'y + x'y = x'(y + y') + y(x + x') = x' + y$

Part 2

2.1

Execution of A: $6 = 34 + $0 = 34, $7 = (-34) + $0 = -34

If $6 < $7 in 2's complement form, set $8 as 1; else set $8 as 0. Value in register $6 is larger. Then **after execution of A, value in $8 = 0, because $6 > $7 in 2's complement form.**

Execution of B: $6 = 34 + $0 = 34, $7 = (-34) + $0 = -34

If $ 6 < $7 in unsigned integers, set $8 as 1; else set $8 as 0. And value in register $7 is larger positive value, so $8 = 1. **After execution of B, value of $8 = 1, because value in $6 <value in $7 in unsigned form.**

2.2

(1)

```
lui $t2, 0x1234          # copy 0x1234 into upper 16 bits of $t2, lower 16 bits are 0, then $t2 is 0x12340000
ori $t3, $0, 0xabcd      # load every bit of 0xabcd immediately to $t3, then $t3 is 0xabcd
addu $t1, $t2, $t3       # add 0x12340000 and 0x0000abcd together, and save the result into $t1
```

(2)

```
slt $t3, $t2, $t1        # $t2 and $t1 contains signed integer in 2's complement
                         # Set $t3 as 1 if $t2 < $t1; else, set $t3 = 0 (result is $t3 = 1)
bne $t3, $0, Label       # Branch to address  if $t3 != $0 (result: branch to address)
```

Part 3

(1)

```
addu $t3, $t1, $t2    # $t3 stores value of b + c
lw $t4, 12($t6)       # $t4 stores value of V[3]
addu $t0, $t3, $t4    # $t0 stores value of b + c + V[3] (by adding value from $t3 and $t4)
```

(2)

```
lw $t3, 12($t6)       # $t3 stores value of V[3]
sll $t3, $t3, 2       # $t3 stores value of V[3] >> 2
addu $t3, $t3, $t5    # $t3 stores value of U[0] + V[3] >> 2
lw $t3, 0($t3)        # $t3 stores value of U[U[0] + V[3] >> 2] (U[V[3]])
addu $t0, $t1, $t3    # a is assigned to the value of b + U[V[3]] (by adding value of $t1 and $t3)
```

Part 4

(1)

ble \$1, \$v1 implements: if less then or equal flow-control statement.

(2)

For register \$t0, it stores the pointer that points the array's element.

For register \$t1, it stores the current element in the array.

For register \$v0, it stores the address of smallest element in array.

For register \$v1, it stores value of the smallest element.

(3)

move \$v0, \$t0: It means update memory address of smallest element(\$t0) to \$v0.

move \$v1, \$t1: It means update memory address of smallest element(\$t1) to \$v1.

(4)

bne \$t0, \$a1, loop: this instruction means loop and stop when the value stored in \$t0 is equal to the value stored in \$a1. It is used when all element in array S are used up then loop will stop.

(5)

This function is to find out smallest value and smallest value's address in the array S.

(6)

\$v0: 0x20060004

\$v1: -29