COMP 2421 Computer Organization
Assignment 1

Firm Deadline: 11:59 pm, Feb. 25 (Sunday), 2024.
Note: please submit a PDF or Word file with file name: "Name_ID.pdf".

1 Questions with Short Answers [10 pts]

For each of the question, you need to write down the detailed steps.

1. Assuming 12-bit 2's complement representation, what is the decimal value of the hexadecimal number FD2? [**2 pts**]

   Solution: -46

2. What is the decimal value of the following floating point number represented in binary form (8 bits for biased exp and 23 bits for significand)? [**2 pts**]

   1 1000 1001 111 1100 1101 0000 0000 0000

   Sign bit = 1 (negative)
   Biased Exponent = 1000 1001 = 137
   Exponent Value = 137 − 127 = 10
   Value = $-(1.11111001101000000000000)_2 \times 2^{10}$
   $= -(11111100110.1000000000000)_2$
   Decimal Value = -2022.5

3. Suppose that we use **4 bits** to represent integers in 2's complement form. Consider the following two cases of binary addition. First, calculate the result. Second, decide whether there is overflow and explain why. [**2 pts**]

   1) Case 1: $0111 + 1111$
   2) Case 2: $1110 + 1000$

   Solution:

   1) Case 1: $0111 + 1111 = 0110$, no overflow. Reason: the two operands have different signs.
   2) Case 2: $1110 + 1000 = 0110$, overflow. Reason: the two operands are both negative but the results is positive.

4. Examine the following program fragment:

   ```
   addiu $t1,$0,-13
   addiu $t2,$0,23
   ```
   <u>the third instruction</u>

   What is the third instruction to set \$t3 to 1 [**2 pts**]? Requirement: using 'sltu'.

   sltu \$t3,\$t2,\$t1.

5. Complete the following program to push the two values stored in \$s1 and \$s2 into stack (note: \$s1 is pushed first) [**2 pts**]:
   addi \$sp, \$sp, _
   sw \$s1, _(\$sp)
   sw \$s2, _(\$sp)

   -8; 4; 0

2 MIPS: Translate Pseudo-instructions [10 pts]

For each of the following Pseduo-instruction with comment, translate it into actual MIPS instructions as specified.

(1) [**3 pts**]

```
ror $t1, 7
#rotate right: rotate the bits in $t1 to the right by 7 positions.
#here, "rotate" means that the bits on the right side are filled into
#the vacated bits on the left (see Fig.1)
Requirement: use a sequence of three instructions: srl, sll, or,
             and another register.
```

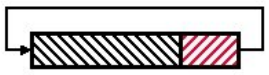

Figure 1: Question 2-(1): rotate right.

(2) [**3 pts**]

```
multiply $t1, $t2, 31
#multuply $t2 with a constant 31 and store the result in $t1
#suppose there's no overflow
Requirement: use a sequence of two instructions: sll, sub.
```

(3) [**4 pts**]

```
lw $t4, 0x00010002($t1)
#load the word stored at memery address $t1 + 0x00010002 into register $t4,
#where $t1 stores the base address
Requirement: use a sequence of four instructions: lui, ori, add, lw,
             and another resigter
```

```
(1)
srl $t2, $t1, 7    # first, shift $t1 to the right by 7 positions,
store the result in $t2
sll $t1, $t1, 25    # second, shift $t1 to the left by (32 - 7) = 25 positions
or $t1, $t1, $t2    # take or operation between $t1 and $t2

(2)
sll $t1, $t2, 5 # $t1 = 32* $t2
sub $t1, $t1, $t2 # $t1 = $t1 - $t2
```

```
(3)
lui $t2, 0x0001
ori $t2, $t2, 0x0002  # load 0x 00010002 into $t2
add $t2, $t2, $t1 # now, $t2 stores the target address $t1 + 0x00010002
lw $t4, 0($t2)
```

3 MIPS: Translate MIPS program into C program [12pts]

Read the following MIPS code segment and comments. Translate it into C code. Specifically, the registers `$s0, $s1, $s2, $s3` store signed integers x, y, z, w, respectively. Complete the C code using expressions of x, y, z, w. Do not care too much about the grammars of C. You will get full marks as long as the meanings are correct.

MIPS code segment:

```
bge $s0, $s1, L1   # go to L1 if $s0 >= $s1
bgt $s2, $s3, skip # go to skip if $s2 > $s3
L1:
bne $s0, $s2, skip # go to skip if $s0 != $s2
L2:                    # inner if statement
bne $s2, $s3, L3   # go to L3 if $s2 != $s3
addu $s0, $s1, $s2
j skip
L3:
addiu $s2, $s1, -2
skip:
```

C code you need to complete:

```
if (some condition 1  [5pts]){
if (some condition 2   [3pts])
{
some code 1   [2pt]
}else{
some code 2  [2pt]
}
}
```

Solution:

```
bge $s0, $s1, L1   # branch if x >= y
bgt $s2, $s3, skip # branch if z > w
L1:
bne $s0, $s2, skip # branch if x != z
L2:                    # inner if statement
bne $s2, $s3, L3   # branch if z !=w
addu $s0, $s1, $s2 # execute inner if
j skip             #
L3:
addiu $s2, $s1, -2 # execute inner else
skip:
```

5 pts "some condition 1": Consider the condition to enter L2. There are two ways: (1) branch in instruction 1 - do not branch in instruction 3; (2) do not branch in instruction 1 - do not branch in instruction 2 - do not branch in instruction 3.

Thus, the corresponding condition is $(x >= y \&\& x == z) || (x < y \&\& z <= w \&\& x == z)$ — Equation 1.
It can be simplified to $(x >= y || x < y \&\& z <= w) \&\& x == z$ — Equation 2.
It can be further simplified to $(x >= y || z <= w) \&\& x == z$ — Equation 3.

Rules of marking: 5pts if got Equation 3 correctly. 4 pts if got Equation 2 correctly. 3pts it got Equation 1 correctly.

3 pts "some condition 2": do not branch. So the condition is $z == w$

2 pts "some code 1": x = y + z

2 pts "some code 2": w = y -2

4 MIPS: Understand MIPS Code [18 pts]

An array of integers S is defined in the following code. Try to understand the code and answer the following questions.

```
S: .word 14, -29, 18, 30, -12, 12, 106, -7

la $a0, S   # load address of S into $a0; suppose $a0 = 0x20060000
addi $a1, $a0, 28
move $v0, $a0  #move the value of $a0 into $v0
lw $v1, 0($v0)
move $t0, $a0
loop: addi $t0, $t0, 4
lw $t1, 0($t0)
ble $t1, $v1, skip # go to skip if $t1 <= $v1
move $v0, $t0
move $v1, $t1
skip: bne $t0, $a1, loop
```

(1) What flow-control statement does `ble $t1, $v1` implement? [**1 pts**]

(2) To show that you fully understand the function of this program, briefly explain the usage of the following 4 registers in the program. That is, what are these registers used for in the program. For example, for register `$a1`, it stores the address of the last element of array, indicating the end of array. [**4 pts**]

Registers: `$t0, $t1, $v0, $v1`

(3) Briefly explain the usage of the two instructions `move $v0, $t0; move $v1, $t1`. [**4 pts**]

(4) Briefly explain the usage of the instruction `bne $t0, $a1, loop`. [**2 pts**]

(5) Briefly explain the function of this program, suppose the desired outputs of the program are the contents of the registers `$v0` and `$v1`. [**3 pts**]

(6) Determine the contents of the registers `$v0` and `$v1` after executing the code. [**4 pts**]

Answer:

```
S: .word 14, -29, 18, 30, -12, 12, 106, -7

la $a0, S # suppose $a0 = 0x20060000
addi $a1, $a0, 28 # $a1 stores the address of the last element S[7]
move $v0, $a0 # $v0 stores the address of the current maximum element
lw $v1, 0($v0) # $v1 stores the value of the current maximum element
move $t0, $a0 # initialize $t0 to be the address of the first element
loop: addi $t0, $t0, 4 # update $t0 to the address of the
                       # current element to be compared
lw $t1, 0($t0) # $t1 stores the value of the current element to be compared
ble $t1, $v1, skip # go to skip if $t1 <= current maximum
move $v0, $t0   # if $t1 is the current maximum, update $v0 and $v1
move $v1, $t1
skip: bne $t0, $a1, loop # check if reached the end of the array
```

1. if
2. `$t0` stores the address of the current array element to be compared
   `$t1` stores the value of the current array element to be compared
   `$v0` stores the address of the maximum element
   `$v1` stores the value of the maximum element
3. since a larger element is found, update the address and value of the current maximum element
4. by checking the address, determine if the program has reached the end of the array
5. find the address and value of the maximum element in the array.
6. `$v0:  0x20060018`
   `$v1:  decimal value 106`