CS220 Quiz#3

General instructions: Please write brief explanation for your answers. If you submit multiple times, your last submission will be used for grading. Please provide an email address below where your responses can be sent.

Email address *

palad@iitk.ac.in

Your name *

Adarsh Pal 4.5/6

Your roll number *

180032

Q1. Write the branch instruction names of the MIPS ISA that will be generated by a reasonably optimized compiler from this C statement: if ((x != y) || (x <= 0)). [1 point]

Suppose address of x is at $3 and address of y is at $4

```
lw $1, 0($3)
lw $2, 0($4)
bne $1, $2, Label1
blez $1, Label1
<Statements to be executed if the condition is false>
j Label2
Label1: <Statements to be executed if the condition is true>
Label 2: <Statements that will always be executed>
```

1

Q2. Write the branch instruction names of the MIPS ISA that will be generated by a reasonably optimized compiler from this C statement: if ((x != y) && (x < 0)). [1 point]

Suppose address of x is at $3 and address of y is at $4.

```
lw $1, 0($3)
lw $2, 0($4)
beq $1, $2, Label1
bgez $1, Label1
<Statements to be executed if the condition is true>
j Label2
Label1: <Statements to be executed if the condition is false>
Label 2: <Statements that will always be executed>
```

1

Q3. Consider a function f written using the C language. The function f calls another function g having twelve arguments all of type integer. The function f can allocate all its local variables in registers without spilling. The structure of f is as follows: {... return g (a, b, c, d, e, ...);}. How much stack space in bytes should be allocated to f when compiling for 32-bit MIPS? [1 point]

The function f requires 8 integer variables (that have been sent as arguments to g) to be sent to the stack space as the max no of arguments that can be sent using registers is only 4 and 12-4=8. Also, f needs to store its return address $ra as it may get modified by g. Hence in total, the stack space in bytes that should be allocated to f when compiling for 32-bit MIPS is 9*4=36 bytes.

1

Q4. Consider the following sequence of 32-bit MIPS instructions separated by semi-colons: [addi $t0, $0, 0xf2; sll $t0, $t0, 0x18; addi $t1, $0, 0x2; srav $t0, $t0, $t1]. What is the final hexadecimal value in $t0? [1 point]

addi $t0, $0, 0xf2; (1)
sll $t0, $t0, 0x18; (2)
addi $t1, $0, 0x2; (3)
srav $t0, $t0, $t1; (4)

0.5

(1) $t0 has 0xf2;
(2) 0x18 in hex is 24 in decimal.
Shifting 0xf2 by 24 bits in decimal gives 0xf2000000
(3) $t1 becomes 0x2
(4) 0x2 is 2 in decimal which shifts t0 by 2 bits giving 0xc8000000 as shifting the values to the left by 2 bits leads to discarding of the leftmost two bits since we can have only 32 bits in 32-bit MIPS.

Q5. Consider the following segment of C code: [for(i=0;i<20;i++) { if (i%2==0) { // Some non-branch statements } else if (i%3==0) { // Some non-branch statements } }]. This code is translated to 32-bit MIPS such that the translated code has the minimum number of branch/jump instructions. When the translated code is executed, calculate how many forward branches/jumps and how many backward branches/jumps are executed. [1+1 points]

for(i=0;i<20;i++) { if (i%2==0) { // Some non-branch statements } else if (i%3==0) { // Some non-branch statements } }

//These are just pseudo instructions for understanding
i=0
Label5: slt and bne will jump to outside of the loop to Label 3 (forward jump)
  mod i,2 equal to zero then jump (forward jump to Label 1)
  mod i,3 equal to zero then continue
Label 2 statements and a forward jump to Label 4
Label 1 statements
Label 4: backward jump to Label5 after updating value of i
Label 3: Final Statements outside the loop

0+1

So, total number of forward jumps= 10+6-3+1(for the false case)=14 and total number of backward jumps=20