CS220 Quiz#4

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 *

sohamg@iitk.ac.in

Your name *

Soham Ghosal

6/6

Excellent!

Your roll number *

180771

Q1. Suppose a processor uses the I-format of MIPS to encode conditional branch instructions. The only difference is that the immediate field has a length of 18 bits. What is the maximum size of a loop in terms of the number of instructions that can be run on this processor if a backward conditional branch instruction is used to check loop termination? Assume that each instruction is of four bytes size. [1 point]

Immediate field has a length of 18 bits.
We use a backward conditional branch instruction(negative offset)
The negative value with the largest magnitude for this conditional instruction would be:
-2^17=0xFFFE0000. This is because the maximum negative value that can be stored in a n bit field is -2^(n-1)[1 bit reserved for sign]

So, the maximum size of the loop could have been 2^17 instructions.
In terms of bytes: 2^17<<2=2^19 bytes.

1

Q2. What are the byte addresses accessed by the lh instruction in the following sequence of instructions separated by semi-colons on a 32-bit MIPS processor? Note that the immediate field is 16 bits long. {lui $1, 0xf000; addi $1, $1, 0xefef; lh $2, 0xdfff($1)} [2 point]

lh instruction would access 2 bytes.

lui will set the upper 16 bits of $1 to 0xf000, and the lower 16 bits remain 0.

0xefef will be sign extended, after which it becomes 0xFFFFEFEF
This now gets added to 0xF0000000.
We get: 0xEFFFEFEF. This is stored in $1

0xdfff is sign extended.
We get 0xFFFFDFFF

Adding this to the contents of $1, we get the new base address as:
0xEFFFCFEE.

2

So, the byte addresses accessed by the lh instruction would be: 0xEFFFCFEE and 0xEFFFCFEF
(Note: MIPS is big endian)(we also note that the base address is 2 aligned)

Q3. Suppose a function f starts at address 0xdefcdefc. Write a sequence of 32-bit MIPS instructions that would call f. Your instruction sequence must work irrespective of where in memory this sequence is placed. [1 points]

```
lui $t0,0xdefc
ori $t0,$t0,0xdefc//preparing the address to jump to in a register
jalr $t0//jumping to that address, and also storing the value of PC+4 in $31[Linking]
```

1

Inside the function, there must have been a command like "jr $31", which enables control to return to the caller function.

Q4. Suppose two variables x and y of type float are allocated in registers $f6 and $f7. Write a sequence of 32-bit MIPS instructions to translate the C statement x = y + 1.375. The sequence cannot use any load/store instructions. Assume that y has already been loaded in $f7. [2 points]

The hexadecimal IEEE754 representation of +1.375 is 0x3fb00000

2

```
lui $t0,0x3fb0 //loading the IEEE754 representation in a general purpose register
mtc1 $t0,$f8 //moving this to a floating point register from a general purpose register
add.s $f6,$f7,$f8 //adding 2 single precision floats,and storing the result in $f6
```