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 *

mainakc@cse.iitk.ac.in

Your name *

Mainak Chaudhuri

Your roll number *

0

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]

The largest magnitude negative number representable using 18 bits is -131072. So, if the PC of the backward branch instruction is x, the PC of the first instruction in the loop is at least x-(131072*4). Therefore, the maximum number of instructions in the loop is 131073 including the backward branch instruction.

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]

```
lui  $1, 0xf000       // $1 <-- 0xf0000000
addi $1, $1, 0xefef   //  $1 <-- 0xf0000000 + 0xffffefef = 0xefffefef
lh   $2, 0xdfff($1)   //  Address = 0xffffdfff + 0xefffefef = 0xefffcfee
```

Byte addresses accessed = 0xefffcfee, 0xefffcfef

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]

```
addi $sp, $sp, -4
sw   $ra, 0($sp)
lui    $at, 0xdefc
ori    $at, $at, 0xdefc
jalr   $at
```

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]

First 1.375 (which is 1.011 in binary) needs to be represented in single-precision IEEE 754 format and copied into a floating-point register (say, $f8). Now, 1.011 binary is represented as 0_01111111_01100000000000000000000. The instruction sequence is shown below.

```
lui      $at, 0x3fb0
mtc1  $at, $f8
add.s $f6, $f7, $f8
```