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 *

palad@iitk.ac.in

Your name *

Adarsh Pal

5.5/6

Very good!

Your roll number *

180032

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 20 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]

1

Maximum negative value in 2's complement representation for 20 bits is -2^19, so maximum negative offset can be -2^19 from the current address. So, address of first instruction of loop is current address+negative offset<<2. This is equivalent to 2^19 instructions backward. Hence, the number of instructions in the loop (if we consider the last instruction inside the loop) is 2^19.

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, 0x8000; addi $1, $1, 0x8888; lh $2, 0xe888($1)} [2 point]

lui $1, 0x8000;
addi $1, $1, 0x8888;
lh $2, 0xe888($1);

2

lui loads 0x8000 to $1 making it 0x80000000. addi adds 0xffff8888 to $1 making it 0x7fff8888.
lh now loads two bytes of data starting from the memory address $1+0xe888 which is equivalent to 0x7fff8888+0xffffe888 (sign extended)=0x7fff7110.

Q3. Suppose a function f starts at address 0xfcdefcde. 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 $1, 0xfcde;
ori $1, 0xfcde;
jr $1;

0.5

jalr

Q4. Suppose two variables x and y of type float are allocated in registers $f8 and $f9. Write a sequence of 32-bit MIPS instructions to translate the C statement x = y + 1.625. The sequence cannot use any load/store instructions. Assume that y has already been loaded in $f9. [2 points]

1.625 is 0x3fd00000 in IEEE754 representation for floats.
y is loaded into $f9.
$1 is a general register.

Instructions are as follows:
lui $1, 0x3fd0;
ori $1, 0x0000;
mtc1 $1, $f8;
add.s $f8, $f8, $f9;

2