CS220: Computer Organization
Quiz#4

Name:
Roll No.:

1. Suppose Booth's algorithm is used in a multiplication where the multiplicand and the multiplier are represented in two's complement and their respective values are 0xcdef01ab and 0xef01abcd. Count the number of addition and subtraction operations. **(1+1 points)**

Solution: The number of addition and subtraction operations depends on the multiplier only. The multiplier is 1110 1111 0000 0001 1010 1011 1100 1101 0. I have also appended a zero on the least significant side as required by Booth's algorithm. The number of additions is equal to the number of transitions from 1 to 0 and the number of subtractions is equal to the number of transitions from 0 to 1 while scanning the multiplier from right to left. So, the **number of additions is 7** and the **number of subtractions is 8**.

2. Consider a program that has 30% load/store instructions, 25% conditional branch instructions, 20% other types of control transfer instructions, and 25% arithmetic and logic instructions. The program is executed on a processor with average CPI of load/store 2, of conditional branch 3.5, of other types of control transfer instructions 3, and of arithmetic and logic instructions 3. Rank these four categories of instructions from most important to least important for optimizing the overall performance of the program. Assume that the clock frequency of the processor which the program runs on is kept constant during the optimization process. **(2 points)**

Solution: For each instruction, average cycle contribution of loads/stores = $0.3 \times 2 = 0.6$, average cycle contribution of conditional branches = $0.25 \times 3.5 = 0.875$, average cycle contribution of other types of control transfer instructions = $0.2 \times 3 = 0.6$, average cycle contribution of ALU operations = $0.25 \times 3 = 0.75$. Therefore, the desired order of importance is **Conditional branches $>$ ALU operations $>$ Loads/Stores = Other types of control transfer instructions**.

Grading policy: 0.5 point if only the most important one or the least important one is correct. 1.0 point if both most important and least important ones are correct. 2.0 points if the entire order is correct.

3. Suppose the variables x, y, z are of signed type of length 32 bits and we would like to compute $z = x - y$. If x is 0xffff0bca, what is the range of permissible values of y so that no overflow occurs in the subtraction used to compute z? Express the upper and lower bounds in hexadecimal of length 32 bits. **(3 points)**

Solution: The representation range using 32-bit two's complement format is $[-2^{31}, 2^{31} - 1]$. Therefore, this is the allowed range for z i.e., $-2^{31} \leq x - y \leq 2^{31} - 1$. While one can compute the range for y given x from these inequalities, we can simplify matters by observing that x is a negative number and hence, y can be any negative number without causing an overflow because in a subtraction an overflow cannot occur when both operands are of the same sign. Therefore, $y \geq -2^{31}$. To find how big y can be without causing an overflow, we use the allowed range of $x - y$ deduced above to get $y \leq x + 2^{31}$ (this is an unsigned addition with the carry out ignored). Therefore, 0x80000000 $\leq y \leq$ 0x7fff0bca.

Grading policy: One point for correct lower bound and two points for correct upper bound.

4. Suppose 1011010 (in binary) is divided by 1001 (in binary). We would like to calculate the number of additions and subtractions when using the restoring, non-performing restoring, and non-restoring algorithms. Fill out the six entries in the table below. **(0.5×6 points)**

Table 1. Count of additions and subtractions in division algorithms

Algorithm	Number of additions	Number of subtractions
Restoring	2	4
Non-performing restoring	0	4
Non-restoring	2	3

Solution: The quotient is 1010 and the remainder is 0. The filled table is shown above. The general rule in restoring division is that the number of subtractions is equal to the number of iterations and the number of additions is equal to the number of zeros in the quotient. For non-performing restoring division, the number of additions is zero and the number of subtractions is equal to the number of iterations. I have shown the steps involved in the non-restoring division below.

```
Remainder = 1011010
Divider   = 1001000  Subtract  [Iteration 1]
-----------------------------------------
Remainder = 0010010
Divider   = 0100100  Subtract  [Iteration 2]
-----------------------------------------
Remainder = 1101110  [Negative]
Divider   = 0010010  Add       [Iteration 3]
-----------------------------------------
Remainder = 0000000  [Non-negative]
Divider   = 0001001  Subtract  [Iteration 4]
-----------------------------------------
Remainder = 1110111  [Negative]
Divider   = 0001001  Add       [Extra iteration, cannot shift divider any more]
-----------------------------------------
Remainder = 0000000  [Non-negative]
```