CS220: Computer Organization
Quiz#4 Solutions

Name:
Roll No.:

1. Consider translating the following C statement where the value of `label` is 0x0 and `label1` is 2^{26} instructions away. This information is available at the time of compilation of the statement. Show the MIPS translation of this C statement using minimum number of instructions. **(3 points)**

```
label: goto label1
```

Solution: The value of `label1` is 2^{28} (i.e., 0x10000000) since each instruction is four bytes. This is beyond the range representable by a `j` instruction. The instruction sequence is shown below.

```
lui $at, 0x1000
jr $at
```

2. 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 0xaabbccdd and 0xabbccdda. 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 1010 1011 1011 1100 1100 1101 1101 1010 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 9** and the **number of subtractions is 10**.

3. Consider a program that has 15% load/store instructions, 25% conditional branch instructions, 10% other types of control transfer instructions, and 50% arithmetic and logic instructions. The program is executed on a processor with average CPI of load/store 10, of conditional branch 4, of other types of control transfer instructions 3, and of arithmetic and logic instructions 2. What is the maximum speedup achievable by optimizing the CPI of the load/store instructions? Assume that the clock frequency of the processor which the program runs on is kept constant during the optimization process. **(2 points)**

Solution: Since the clock frequency and instruction count remain unchanged during the optimization, the speedup can be calculated by the ratio of baseline CPI to optimized CPI. Baseline CPI = $0.15 \times 10 + 0.25 \times 4 + 0.10 \times 3 + 0.50 \times 2 = 3.80$. Optimized CPI (obtained by setting the CPI of load/store instructions to zero) = $0.25 \times 4 + 0.10 \times 3 + 0.50 \times 2 = 2.30$. **Speedup = 3.8/2.3 = 1.65.**

4. Suppose 1001010 (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$\times$6 points)**

Table 1. Count of additions and subtractions in division algorithms

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

Solution: The quotient is 1000 and the remainder is 10. 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 = 1001010
Divider   = 1001000  Subtract  [Iteration 1]
-------------------------------------------
Remainder = 0000010
Divider   = 0100100  Subtract  [Iteration 2]
-------------------------------------------
Remainder = 1011110  [Negative]
Divider   = 0010010  Add       [Iteration 3]
-------------------------------------------
Remainder = 1110000  [Negative]
Divider   = 0001001  Add       [Iteration 4]
-------------------------------------------
Remainder = 1111001  [Negative]
Divider   = 0001001  Add       [Extra iteration, cannot shift divider any more]
-------------------------------------------
Remainder = 0000010  [Positive]
```