CS230: Digital Logic Design and Computer Architecture
Tutorial 03 [Mon 02 Sep, Tue 03 Sep, Thu 05 Sep]

Concepts tested: Computer Arithmetic in MIPS, Computer Performance Quantification, Single Cycle Implementation

1. Consider the following Java code fragment:

   ```
   for(i = k; i < N; i++) { a[i] = b[i] + c; }
   ```

   Assume that 'a' and 'b' are arrays of words. Also assume that the base address of 'a' and 'b' are in $a0 and $a1 respectively. Register $t0 is associated with 'i', $t1 with 'k', $t2 with 'N', and register $s0 with the value of 'c'. Assume that the (maximum allocated) lengths of arrays 'a' and 'b' are in $a2 and $a3 respectively.

   (a) Translate the above code into MIPS assembly code. Comment your code appropriately, and use intuitive label names. You have to check for memory out-of-bounds exception before each array reference. If there is any out-of-bounds exception, you should break out of the loop by jumping to a label called 'OutOfBounds'. Write the code without any optimization taking advantage of MIPS's 2's complement representation.

   (b) What optimization can you make in the above code to take advantage of 2's complement representation, while doing the out-of-bounds check?

   (c) Suppose you want to further optimize the above code by extending the MIPS ISA. To the ISA, we want to add instructions lw_inc and sw_inc, which increment something in addition to loading/storing. What is this something which must be incremented by the instruction? And what should be the default increment value?

   (d) Which of sw_inc and/or lw_inc can you use in the above code to optimize it? How?

2. **MIPS code performance analysis**
   For this question, take the code segment from the previous question. And assume that the 2's complement-based optimization in part (b) of the above question always applies.

   (a) How many instructions does the optimized code from (b) execute, assuming that there are no out-of-bounds exception? Give your answer in terms of appropriate symbols, as necessary.

   (b) What is the new instruction execution count after using sw_inc and/or lw_inc in (d)?

   (c) Suppose that the addition of sw_inc/lw_inc to the ISA results in an factor of 'x' increase in the clock period. For what values of 'x' does the use of sw_inc/lw_inc improve the performance of the above code segment? For what values of 'x' does it degrade the performance?

3. [**Based on Q4.11 from the text**] Consider program P which runs on a 1 Ghz machine M in 10 sec. An optimization is made to P, replacing all instances of multiplying a value by 4 (mult X, X, 4) with two instructions that set X to X+X twice (add X, X, X; add X, X, X). Call this optimized program P1. The CPI of a multiply instruction is 4, and the CPI of an add is 1. After recompiling, the program now runs in 9 sec on M.

   (a) How many executed multiplies were replaced by the compiler?

   (b) What is the minimum possible increase in static code size when going from P to P1?

4. The following is a C++ implementation of a method for computing the square root of a given number (assumed positive).

   ```
   float sqrt(float x) {
     float est = 1;
     while ((est*est) != x) { est = 0.5*(est + x/est); }
     return est;
   }
   ```

   (a) What is wrong with the above code and how can you correct it?

   (b) What is the name (in Mathematics) of the algorithm logically implemented by the given algorithm ? (Up to 3HP for the first three who identify the name and tell the instructor during the tutorial).

5. Draw the hardware diagram for the single cycle implementation of the MIPS instruction subset add, sub, and, or, slt, lw, sw, beq.