CSC 462/562 Computer Architecture
Homework #1: Due Wednesday, January 30

This assignment covers chapter 1 and appendix A. Undergraduate students answer any 5 of the 7 problems. Graduate students answer all 7. All answers are to be word processed. Submit by hardcopy if possible.

1. Let's see why Amdahl's Law promotes the common case. Write a program that will step through F = 20%, 30%, 40%, …, 90% with k = 2. Next, step through for F = 40%, k = 7, 9, 11, …, 21. Do the same for F = 75% and k = 1.4, 1.6, 1.8, …, 2.8. Show the results of each of these pairs of values. Provide a brief explanation of what you found that demonstrates that the common case has a greater impact than the enhancement's speedup (k).

2. The CPI of a processor is as follows:
   Loads = 6
   Stores = 5
   ALU operations = 4
   Conditional branches = 4
   Unconditional branches = 3

   Architects are considering one of two enhancements. First, they are considering improving all ALU operations to be reduced to 3. Second, they are considering reducing all branches (conditional and unconditional) to 2. Using the sjeng benchmark from figure A.29 (p. A-42), compute the average CPI before the enhancement and then with each enhancement and compute each enhancement's speedup over the original. The speedup is old CPI / new CPI. Redo this with the mcf benchmark. Compare the speedups between the two benchmarks and offer a suggestion of which speedup might be preferred.

3. Architects want to add ALU operations that can use a datum from memory as one of its source operands, such as `add x1, x2, (5000)` to add the datum at location 5000 and the datum in x2 storing the result in x1, or store the result into memory as its destination as in `add (5000), x1, x2` to add the data in x1 and x2 and place the result at location 5000. This type of ALU operation will eliminate some loads and stores. Let's assume 30% of all loads and 15% of all stores will be eliminated from a program and the new instruction will be used in place of an existing ALU operation. The result is a reduction in IC. These new ALU operations have a CPI of 7 instead of 4. Use the CPIs from #2 for the remainder of the instructions. Note that the ALU operations that operate solely on registers (and immediate data) still have a CPI of 4.
   a. Will this change result in a speedup or slowdown? In order to determine this, compute the current machine's CPI * IC using the astar benchmark (figure A.29 on page A-42) and then the new CPI * IC. Determine the amount of speedup/slowdown.
   b. Redo part a assuming that this change requires also lengthening the clock cycle by 10%. Now determine what the speedup or slowdown will be.

4. Older processors performed multiplication in software rather than hardware. The following is the unsigned multiplication algorithm (as covered in CSC 362).
    1. A ← 0, Q ← X, C ← 0, M ← Y
    2. For I = 1 to n do // n is the number of bits in A, Q and M)
        a. If Q0 = 1 then A = A + M, move any carry out to C
        b. Right shift C, A and Q

    The product is stored in A and Q combined

   Assume step 1 takes 2 cycles (one to load X into Q, one to load Y into M), the for loop mechanism itself takes 1 cycle per iteration, and steps 2a and 2b take 1 cycle apiece per iteration. Assume a benchmark consists of 5% multiplications. We want to determine the speedup of using a hardware multiplier with a CPI of 8. If the benchmark is a 16-bit program (n = 16), what is the speedup of using the hardware multiplier? Repeat assuming a 64-bit program (n = 64). Use Amdahl's Law.

5. In RISC-V, x0 always stores 0.
    a. x0 cannot be used to store a result so seems of limited use. Provide two uses of x0. Show example RISC-V code of your uses.
    b. If RISC-V did not dedicate x0 to store 0, show how you would accomplish your two examples from part a using different code.

6. Rewrite the following C code in RISC-V assembly language assuming array a is an int array starting at memory location 10000 and has 500 elements in it and that x, y and z are stored at memory locations 9988, 9992, 9996 respectively. Store i in a register so that you do not need to access memory for it.

```
for(i=0;i<500;i++)
    if(a[i] >= x) y++;
    else z++;
```

7. Implement the code given in problem A.20 (bottom of page A-53) in RISC-V assuming that arrays A and B are long int arrays starting at locations 5000 and 6000 respectively, and C and D are long ints stored at locations 7000 and 7008 respectively. Store i in a register so that so that you do not need to access memory for it.