CS230 Winter 2015 – Assignment 4

Due Date: Wednesday, March 18, 2015, 12:00 p.m. (noon)
Weight: 5% of the course grade

- All submissions are to be completed through Markus.
- The files submitted must be either plain text files (.txt), or pdf files. Some questions on this and future assignments **must** be submitted as a plain text file.
- When creating plain text files outside of Unix, choose a good, simple editor. Programs like Microsoft Word are not a good choice for generating plain text files.
- You may submit scanned, hand-written solutions, however the source material must be readable by the markers.
- In some cases, it is very difficult to create solutions in plain text format. However, the markup in Markus works best on plain text files. Where you are able, it is recommended that you submit solutions in plain text.
- Where you are allowed more than one file format for a question, Markus is configured to expect two files for each question. In this case, you should get a message from Markus indicating that you have not submitted all the required files. This is not a problem as long as you have successfully submitted one file for each of the questions.
- Read the policy described on the course website https://www.student.cs.uwaterloo.ca/~cs230/assignments regarding late submissions.
- Solutions for assignments will not be posted.

1. **[5 marks]** Show your work in the computation for the following questions involving CPI and performance.

   a) Suppose you have two processors with the following specifications:
      Processor 1 runs at 2.4 GHz and has a CPI of 1.2
      Processor 2 runs at 3.8 GHz and has a CPI of 2.5

   Which processor will perform better given any instruction set? How much better is the performance?

   b) Consider the following situation:
   - you are running a program on a 2.4 GHz processor
   - the program instructions can be divided into two categories: memory access and non-memory access
   - memory access instructions take 6 times as long as non-memory access instructions
   - 25% of your instructions involve memory access

   Assume you have found an optimization technique that will keep the same number of instructions, but reduce the amount of memory access instructions to 20%. What is the improvement in the overall performance of the optimized program compared to the original?

   Submit the file `a4q1.txt` or `a4q1.pdf`.

2. **[10 marks]** For this question, refer to the following MIPS assembly code.

```
          lis $3
          .word 0xffff000c
          addi $4, $0, 0x60
          addi $5, $0, 0x7b
          lw $6, 0($1)
loop:     beq $6, $0, endloop
          slt $7, $4, $6
          slt $8, $6, $5
          mult $7, $8
          mflo $7
          beq $7, $0, endif
          addi $6, $6, -32
endif:    sw $6, 0($3)
          addi $1, $1, 4
          lw $6, 0($1)
          bne $6, $0, loop
endloop:  addi $7, $0, 0xa
          sw $7, 0($3)
          jr $31
```

a) Identify all data and control pipeline hazards under the assumption that no pipeline optimizations have been implemented – including forwarding. The code has been numbered so that you can refer to these line numbers in your explanation.

b) Suppose you ran the program using the array frontend, and entered 20 as the size of the array, and then entered the following 20 numbers as array values:
`67 83 50 51 48 32 105 115 32 116 104 101 32 98 101 115 116 33 33 0`
How many stalls would be saved if there was forwarding in the pipeline? Explain your answer. How many stalls would be saved if there was no forwarding, but you used static branch prediction? Explain your answer.

Submit the file `a4q2.txt` or `a4q2.pdf`.

3. **[10 marks]** For this question, you are permitted to use any online or offline source available to you to carry out literature research. Make sure to list all sources that you use to prepare your answers. Answer the question below **in your own words**.

In the lectures we discussed a few optimization techniques, including reordering of MIPS code to avoid stalls in pipelining, and branch prediction. There are many other optimization techniques that exist in hardware and software. Optimization techniques can involve reducing the instruction count and/or reducing the CPI. One optimization technique is known constant folding and propagation. In one or two paragraphs, explain how this approach works. The level of your explanation should be so that a student in CS230, who has never heard of this technique can understand it. In other words, you can assume that your audience understands the basics of MIPS code and the concept of registers etc. You should not assume any particular mathematical background for your audience, since many students in CS230 are not in the Math faculty.

Using the following pseudocode:

```
a = 7
b = a * 52 * 10
x = 0
for c = 1 to 100
   if (c + a * 100) < b
      x = x + c
return x
```

explain how constant folding and propagation will improve the run time of this code. Be specific by describing the difference in the MIPS code instructions that would be executed in the case where there is no optimization versus where there is optimization.

The purpose of this question is for you to do a small amount of research yourself. Do not ask questions in Piazza to clarify how constant folding and propagation works.

Submit the file `a4q3.txt` or `a4q3.pdf`.

4. **[10 marks]** Assume a memory model where you have a cache size of 4 blocks, a block size of 16 words/block, and addresses from 0 through 511 in main memory.

   a) What addresses would be in the same block as address 201 in main memory?

   Assume that the cache is empty before the start of the sequence in each case. Assume that the 2-way associative cache uses a LRU (last recently used) eviction strategy. Consider the following sequence of memory accesses:

   ```
   18, 70, 4, 16, 65, 10, 84, 470, 12, 475, 90, 70
   ```

   For each style of cache below, identify if each memory access in the sequence would be a hit or a miss. Include a table that shows the state of the cache after all accesses are complete.

   b) for a direct mapped cache (use the table on slide 68 as a template)
   c) for a 2-way associative cache (use the table on slide 69 as a template)

   Submit the file `a4q4.txt` or `a4q4.pdf`.