CPSC 261 Sample Midterm 1
February 2016

Name: ______________________ Student ID: ______________________

Signature: ______________________

- You have 60 minutes to write the 6 questions on this examination. A total of 45 marks are available.
- **Justify all of your answers.**
- You are allowed to bring in one hand-written, double-sided 8.5 x 11in sheet of notes, and nothing else.
- Keep your answers short. If you run out of space for a question, you have written too much.
- The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.
- Use the back of the pages for your rough work.
- **Good luck!**

Question	Marks
1	
2	
3	
4	
5	
6	
Total	

UNIVERSITY REGULATIONS:

- Each candidate should be prepared to produce, upon request, his/her UBC card.
- No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.
- CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.
  1. Having at the place of writing, or making use of, any books, papers or memoranda, electronic equipment, or other memory aid or communication devices, other than those authorised by the examiners.
  2. Speaking or communicating with other candidates.
  3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.
- Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

[8] 1. Short Answers

[2] a) List two techniques for coping with complexity, and briefly describe each of them (one or two sentences suffice).

[2] b) Give two reasons why coherency and atomicity are both desirable properties of the *memory* abstraction that are difficult to achieve.

[2] c) When evaluating the performance of a pipelined CPU, should you look at the latency of the pipeline, or at its throughput? Why?

[2] d) What would happen to your program if you ran it on a CPU whose pipeline violates sequential consistency?

[8] 2. More short answers

[2] a) What technique does the Intel core CPUs utilize when confronted with a very complicated instruction, such as `movq 0x40(%rsi, %rdi, 4), %rbx`, in order to make it easier to execute on its pipeline?

[2] b) Core memories and DRAM have one property in common, that relates to the way in which they work. What is it?

[2] c) Most of the memory in a computer is much slower than the CPU. What property of well-written programs allows us to make it appear as if most of it were actually fast?

[2] d) Which area of memory does the memory returned by a call to `malloc` belong to, and who manages it?

[5] 3. Each of the following two pieces of code has a bug that will cause it to *possibly* not behave the way the comments indicate it should, or fail to interact properly with the rest of the program. In each case, explain (1) what the problem is and (2) how you would fix the bug. Recall that registers `%rbx`, `%rbp`, and `%r12` to `%r15` are callee saved. All other registers are caller saved.

[3] a) `test1:`

```
    leaq (%rdi, %rdi, 2), %rdi # multiply parameter "n" by 3
    call test3                 # compute something with 3n
    addq %rdi, %rax            # add 3n to test3's return value
    ret
```

[2] b) `test2:`

```
    leaq (%rdi, %rdi, 2), %r15 # save 3n into %r15
    movq (%rsi, %r15, 8), %rdi # load array[3n] into %rdi
    call test4                 # compute something with it
    addq %r15, %rax            # add 3n to test4's return value
    ret
```

[10] 4. Using what you know about pipelined processors, explain why:

[2] a. conditional branches decrease throughput.

[2] b. dependencies between instructions decrease throughput.

[2] c. a pipeline with more stages is capable of executing more instructions per second than one with fewer stages (under the right conditions).

[2] d. if we keep adding pipeline stages, we will eventually get to the point where additional stages no longer increase throughput.

[2] e. using the pipeline to execute two threads (alternating between them) increases throughput compared to first executing one thread, and then the other.

[7] 5. Consider the following C function:

```
void do_something()
{
    char *mtl = malloc(5 * sizeof(char));
    char *qc;
    char **cdn = &qc;

    mtl[0] = ’Y’;
    mtl[4] = ’\0’;
    qc = mtl + 2;
    *qc = ’A’;
    qc[-1] = ’P’;
    *cdn = qc + 1;
    **cdn = ’Z’;
    *(qc - 1) = ’E’;
    printf("%s\n", mtl);
}
```

[5] a) What characters will the call to `printf` print to the screen? Justify your answer by drawing pictures of the array and of the pointers.

[2] b) What additional statement(s) should be added at the end of this function, and why? Note: the answer is not “a return statement”.

[7] 6. Consider the following sequence of instructions written in x86_64 assembly language:

```
1  function:
2    movq   %rdi, %rax
3    subq   %rsi, %rax
4    jle    done
5    movq   %rsi, %rbp
6    movq   $2,   %rax
7    addq   %rdi, %rbp
8    subq   %rax, %rbp
9    movq   0x1000(%rsi), %rax
10   andq   %rax, %rbp
11   pushq  %rdi
12   movq   %rbp, %rdi
13   call   function
14   popq   %rdi
15   call   otherfunction
16 done:
17   popq   %rbp
18   ret
```

List the seven of the eleven hazards that are present in this sequence of instructions. For each hazard:

- If it is a data hazard, write D, and both the number of the instruction that uses the data value, and the number of the instruction that computes it. If an instruction uses several values computed by earlier instructions, list only the number of the most recent such instruction. One of the data hazards is fairly hard to find.

  Do not worry about the possible existence of data hazards where only one of the two instructions involved is shown here (for instance, if the second instruction is in `otherfunction`).
- If it is a control hazard, write C and the number of the instruction where the hazard occurs.

(1)

(2)

(3)

(4)

(5)

(6)

(7)