High Performance Computing

Homework #1

Due: Tuesday February 9 2016 by 11:59 PM (Midnight)

Email-based help Cutoff: 5:00 PM on Mon, Feb 8 2016

Maximum Points: 30

Submission Instructions

This homework assignment must be turned-in electronically via Niihka. Type in your responses to each question (right after the question in the space provided) in this document. You may use as much space as you need to respond to a given question. Once you have completed the assignment upload:

1. The MS-Word document (duly filled) and saved as a PDF file named with the convention `MUid.pdf` (example: **`raodm.pdf`**)
2. The three C++ programs developed as part of questions 9-11.

Note that copy-pasting from electronic resources is plagiarism. Consequently, you must suitably paraphrase the material in your own words when answering the following questions.

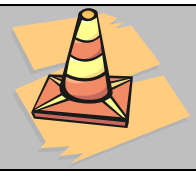
Name: Henry Ni

Objective

The objective of this homework is to review the necessary background information about computer architecture, related concepts, and terminology. The terminology will be often used in the course.

Read chapters 4, 5, and 6 of the E-book titled "How Computers Work" (all students have free access to the electronic book) and then answer the first set of questions.

Once you have answered the first set of questions, read chapters 1, and 2 of the E-book titled "C++ How to Program" (all students have free access to the electronic book) and then answer the following questions.



Although the Safari E-books are available to all students there are only a limited number of concurrent licenses to access the books. Consequently, do not procrastinate working on this homework or you may not be able to access the E-books due to other users accessing books.

1. Briefly describe (1 or 2 sentences will suffice) each one of the following terms: [**3 points**]
   a. ALU

The chip responsible for performing math instructions

   b. Address bus (or Address lines)

Marks the location on memory where data can be stored with a physically etched mark on a RAM chip

   c. CISC

Complex Instruction Set Computing: a processor design where complex instructions are turned into a series of simpler statements before any work is done

   d. RISC

Reduced Instruction Set Computing: the opposite of CISC, only simple instructions are used to perform tasks

   e. Register

Used by a CPU to store data used in calculations, small size and limited space

   f. Using the image in section titled "How the Processor Uses Registers" in Chapter 6, mention name of three registers (such as: MDR/ACCUMULATOR) and write one line description of the function of each one of the 3 registers.

Memory Data register: stores values to be written to RAM
Program Counter register: holds memory location of next value processor will work on
Accumulation register: stores returned values that still will be used in future operations

2. Using the detailed image in section titled "How a Microprocessor Moves Data", describe the various phases involved in processing an instruction. Your description must briefly (1 complete sentence will suffice) cover the following: BIU, BTB, μops, ALU, cache use, speculative execution, JEU, and retirement unit. [**2 points**]

Instructions flow through the bus interface unit and sends a copy of the program code to the level 1 cache of the CPU. A separate decoder on the CPU will work on processing the data in the cache while a branch target buffer finds and predicts a path the program will take if a branching instruction is given. Decoded instructions are broken up into micro-ops to speed up execution (several small instructions are faster to execute than one complex instruction). All μops are sent to a reorder buffer which feed into two ALUs that do the actual computations. To reduce idle time the execute unit goes though the buffer of μops that it can act on, this process of checking/executing is called speculative execution. A delayed μop is completed, the execute unit will compare the results to the prediction made by the BTB; in the case of an incorrect prediction the jump execution unit moves the end marker to the spot the incorrect μop was in, showing all the μops after that spot are now irrelevant. The prediction information is also given to the BTB so it may make more accurate predictions in the future. At the end of all this, a retirement unit checks if the first μop in the circular buffer is executed, and then the second and third. If all three are complete, then the maximum is sent to the store buffer where the prediction unit checks a final time before sending the end result to RAM.

3. Provide suitable response to each of the following questions regarding a modern PC.

   a. What is the expansion of the acronym RAM? [**0.5 points**]

Random Access Memory

   b. What is the expansion of the acronym ROM? What is the difference between RAM and ROM? [**0.5 points**]

Read only memory – is non-volatile so data saved to ROM will not be lost if the computer is turned off, unlike RAM

   c. What is a microprocessor aka CPU? List at least 3 major subcomponents of a CPU? [**1 points**]

CPUs are the master processor in a computer, contains ALU(s), control unit and registers

   d. What is a CPU core? What is a dual-core or multi-core CPU? [**0.5 point**]

A CPU core contains all the parts needed to execute instructions, and n-core CPUs have that many processors sharing the same cache, memory, etc.

   e. What is the difference between a CPU Core and an ALU? [**0.5 point**]

An ALU is a small component in a CPU core

4. Briefly (about 3-5 sentences each) describe and contrast the following high performance features in modern CPUs: [**6 points**]
   a. Superscalar operation

Brief description (3-5 sentences):
Executes multiple instructions per clock cycle by sending different instructions to different execution units within the CPU (but not different cores). Requires that each instruction doesn't have any data dependent on pending operations and that instructions are sent from a sequential stream.

Contrast with others (2 sentences):
Not truly parallel, since any instructions with data dependence won't be executed. Also, multiple instructions may be processed at once instead of having multiple portions of a single instruction being computed with SIMD/SSE.

   b. SIMD operation

Brief description (3-5 sentences):
A form of parallelism where multiple (similar) operations within a single instruction are performed at once on different data points.

Contrast with others (2 sentences):
SIMD compatible operations are done in special registers (usually 256 bits wide) and are limited to a single instruction.

   c. SSE operations

Brief description (3-5 sentences):
Extends SIMD instructions to work with single precision floating point data. Further inclusions and revisions are named SSE2, SSE3, SSSE3 (supplemental streaming SIMD extension ver. 3) and SSE4.

Contrast with others (2 sentences):
SSE reuses SIMD registers, but its capabilities are expanded and can operate on multiple 256-bit blocks of data at a time. Data can also be fetched while an operation is being performed, unlike superscalar operation.

5. What is overclocking? What are the two commonly used approaches for overclocking? [**1 points**]

Overclocking, in short, allows your CPU to operate faster than the advertised speed by either increasing the clock multiplier or increasing the speed the front side bus operates at.

Ensure you have read chapters 1 and 2 of the E-book titled "C++ How to Program" (all students have free access to the electronic book) and then answer the following questions.

6. Briefly (1 to 2 sentences for each phase) describe the following phases involved in creating and running a C++ program: creating, preprocessing, compiling, linking, loading, and execution. [**2 points**]

From your IDE/text editor/console, you type your source code (creating). Once you are ready to compile, you issue a build command which causes a preprocessor to analyze your code for any directives that need to be addressed before compilation: including other files, text replacements, etc. Compilation occurs after preprocessing and translates source code into machine code which leads to linking, the stage which patches any holes that references to outside functions/data may lead to errors. If a program compiles and completes the linking phase successfully, a .exe file is created. The .exe is transferred to memory by the loader and only then can the CPU begin to execute the file.

7. What is the significance of the main function in a C++ program returning the value of zero? [**1 point**]

After the processor reads a 0 value, it knows the program has completed successfully.

8. High Performance Computing (HPC) aims to improve efficiency of programs by (1) reducing program runtime, (2) reducing memory (or resource) usage, and (3) efficiently utilize energy. However, depending on the nature of the application, programmers have to balance the priority (or order of importance) of these three key components of efficiency. In each of the following scenario discuss/justify which one of the factors plays the most significant role. A solution is already provided for the first question to illustrate an example. [**3 points**]
    a. A HPC programmer (i.e., *yours truly*) is optimizing a program to run on Ohio's Supercomputer (https://www.osc.edu/supercomputing/computing/oakley) for forecasting the spread of Zika virus (see: http://in.reuters.com/article/health-zika-who-idINKCN0V61JD) to inform multinational policies. Which one of the aforementioned 3 efficiency factors should the programmer primarily focus on?

    In this situation getting results quickly is critical to combat the ongoing epidemic that is affecting > 10 million people. Since the program is running on a Supercomputer memory is not an issue. Energy concerns is unequivocally superseded by the need to quickly help millions of people.

    Decision: In this situation, the programmer must strive to minimize runtime of the program while trading off memory and energy.

    b. A HPC programmer is developing a program to run on a bridge monitoring device. The device has special accelerometers to quickly measure vibrations and compactly store it permanently in a large flash memory. Since the bridge is in a remote location a small battery will be used to power the device. Which one of the aforementioned 3 efficiency factors should the programmer primarily focus on?

    Since storage is taken care of by a large flash memory device, the programmer should strive to optimize energy usage, since it will be difficult to replace/recharge the battery if it dies. Program runtime is not as large of a priority in this case, since the practicality of supporting the device trumps resource usage concerns.

c. A HPC programmer is developing a program for a car collision avoidance system to rapidly (in microseconds) respond to head on collision situations. The program is run on a dedicated (different from navigation/entertainment system) Intel Atom CPU (@2.5 Ghz) with 8 GB of RAM. Which one of the aforementioned 3 efficiency factors should the programmer primarily focus on?

Response time is the most important factor in avoiding car accidents, so reducing runtime is the clear goal. Storage is taken care of with the 8GB of ram, and power draw will come from the car itself so managing energy efficiently is not as much of an issue.

d. A HPC programmer is developing a program to report temperature (requires 1 second to report temperature) once an hour on an embedded device based on ATmega328 (http://www.atmel.com/devices/atmega328.aspx?tab=parameters). The device is powered from the main power supply. Which one of the aforementioned 3 efficiency factors should the programmer primarily focus on?

Since the program is running on an embedded device, optimizing memory use would be the best decision out of the three options. Power is delivered from a reliable source and runtime is not nearly as much of an issue since only one report will need to be generated per hour.

9. Develop a complete C++ program to that prompts and obtains two integers from the user and prints the sum, product, difference, and quotient of the two numbers as shown in the adjacent sample output (user inputs are shown in **red**). Your program should compile without any errors or warnings and successfully run under Linux. Ensure your C++ source file is named with the convention *MUid*_hw1_9.cpp, where *MUid* is your unique ID. In addition to uploading the C++ source file to Niihka, copy-paste the source code in the space below. [**3 points**]

```
Enter 2 integers: 9 4
sum       : 13
product   : 36
difference: 5
quotient  : 2
```

```
#include <iostream>

using namespace std;


int main() {
  int a = 0, b = 0;

  cout << "Enter two integers: ";
  cin >> a >> b;

  cout << "\nsum = " << a + b << endl;
  cout << "difference =  " << a - b << endl;
  cout << "product = " << a * b << endl;
  cout << "quotient = " << a / b << endl;
  return 0;

}
```

10. Develop a complete C++ program to that prompts and obtains three integers from the user and prints the smallest and largest of the three numbers as shown in the adjacent sample output (user inputs are shown in **red**). Using `std::min` and `std::max` functions can simplify the code. Your program should compile without any errors or warnings and successfully run under Linux. Ensure your C++ source file is named with the convention *`MUid`*`_hw1_10.cpp`, where *`MUid`* is your unique ID. In addition to uploading the C++ source file to Niihka, copy-paste the source code in the space below. [**3 points**]

```
Enter 3 integers: -5 -9 -7
smallest: -9
largest : -5
```

```
#include <iostream>

using namespace std;

int main() {
  int a, b, c;

  cout << "Enter 3 integers: ";
  cin >> a >> b >> c;

  cout << "largest = " << max(max(a, b), c) << endl;
  cout << "smallest = " << min(min(a, b), c) << endl;

  return 0;
}
```

11. Develop a complete C++ program to that reads a month name ("`january`" through "`december`") from the user and prints the corresponding numerical month value (1 through 12) as shown in the adjacent sample outputs (user inputs are shown in **red**). You may assume the user enters only valid month names in all lowercase letters. Here are a few tips that may come in handy:

```
Enter month: august
Month: 8
Enter month: december
Month: 12
```

    a. The `std::string` objects in C++ can be compared with each other using equal (`s1 == s2`) and not-equal (`s1 != s2`) operators just like integers.
    b. You may use the stream extraction operator (>>) to read a word from the user as shown below:

```
std::string month;
std::cin >> month;
```

Your program should compile without any errors or warnings and successfully run under Linux. Ensure your C++ source file is named with the convention *`MUid`*`_hw1_11.cpp`, where *`MUid`* is your unique ID. In addition to uploading the C++ source file to Niihka, copy-paste the source code in the space below. [**3 points**]

```
#include <iostream>
using namespace std;

int main() {
  string month;

  cout << "Enter a month: ";
  cin >> month;
  if (month == "january")
    cout << 1;
  else if (month == "february")
    cout << 2;
  else if (month == "march")
    cout << 3;
  else if (month == "april")
    cout <<  4;
  else if (month == "may")
    cout <<  5;
  else if (month == "june")
    cout <<  6;
  else if (month == "july")
    cout <<  7;
  else if (month == "august")
    cout <<  8;
  else if (month == "september")
    cout <<  9;
  else if (month == "october")
    cout <<  10;
  else if (month == "november")
    cout <<  11;
  else
    cout <<  12;
  return 0;
}
```