CS 3194 Design Project

Due on or before April 26, 2010, 4:30 p.m.

February 1, 2010

This problem is to be done on a group basis following the Trinity University Academic Integrity Policy or Trinity University Honor Code.

Academic Integrity and Honor Code

All students are covered by a policy that prohibits dishonesty in academic work. The Academic Integrity Policy (AIP) covers all students who entered Trinity before the Fall of 2004. The Academic Honor Code covers all those who entered the Fall of 2004 or later. The Integrity Policy and the Code share many features: each asserts that the academic community is based on honesty and trust; each contains the same violations; each provides for a procedure to determine if a violation has occurred and what the punishment will be; each provides for an appeal process. The main difference is that the faculty implements the AIP while the Honor Code is implemented by the Academic Honor Council. Under the Academic Integrity Policy, the faculty member determines whether a violation has occurred as well as the punishment for the violation (if any) within certain guidelines. Under the Honor Code, a faculty member will (or a student may) report an alleged violation to the Academic Honor Council. It is the task of the Council to investigate, adjudicate, and assign a punishment within certain guidelines if a violation has been verified. Students who are under the Honor Code are required to pledge all written work that is submitted for a grade: On my honor, I have neither given nor received any unauthorized assistance on this work and heir signature. The pledge may be abbreviated pledged with a signature.

Laboratory problems should be submitted electronically (e-mail to cs3194@leda.cs.trinity.edu) on or before the due date and should contain a problem write-up, source code to any programs and data sets used in solving the problem. The submitted files should be ASCII text files having Unix end-of-line characters (please convert all Windows and Mac text files to Unix format—I have found that Emacs or the Stone Text Tool seem to do a reasonable job of such conversions). If several files need to be submitted, put them in a directory having name your-last-name-problem-set-number and create a tar archive of this file system and attach it to your e-mail problem submission.

Introduction

A computer is an interpreter for the machine language of its processor. A processor emulator is a computer program which interprets the machine language of some processor design. A processor emulator operates on bit patterns stored in memory which represent program instructions and data in a fashion which is identical to the way the actual processor would operate on those same instructions and data.

Computer emulation software has played an important role in the history of computing. When designing new computers, it has been common practice to build a software emulator for a new processor so that software development for the new machine can precede the availability of the new hardware. Also, software implications of hardware design trade-offs may be evaluated with emulation without having to actually build each option.

Emulation technology has been used to build unusual computer designs using an existing processor design. For example, in 1975, IBM introduced the IBM 5110 portable computer. This machine directly interpreted the APL programming language. This APL machine was constructed by using a micro computer version of the IBM 360 computer and emulation software, stored in read-only memory, which interpreted the APL programming language. The APL interpreter used in the 5110 computer was the IBM 360 mainframe APL interpreter. Another version of the IBM 5100 series of computers directly interpreted the BASIC programming language using similar emulation technology.

Apple Computer has successfully used emulation technology twice in the history of various Macintosh computer designs. The original Macintosh computer used a Motorola 68000 micro processor which was an early 32 bit design. Later, Apple, IBM, and Motorola formed a joint effort to design and manufacture the PowerPC processor which was
based on the IBM Power architecture. They formed the Somerse t Design Center in Austin Texas where IBM, Apple, and Motorola engineers worked together on the project. Apple’s plan was to switch from using the 68000 chip in Macintosh computer to using the PowerPC chip in new Mac computers.

Apple built a 68000 emulator into the Mac operating system so that new Mac computers could not only run old 68000 programs, but also run new PowerPC programs in a seamless fashion. In fact, during a transition period, much of the MacOS operating system itself ran on the emulator of PowerPC Mac computers. Later, when the operating system had been converted to PowerPC code, the performance of Mac computers improved significantly. Microsoft was slow in converting its Mac Office products and during the transition period, those programs ran slowly.

A few years later, Apple made the decision to switch from the PowerPC processor to Intel processors because the IBM and Motorola PowerPC chips lagged far behind Intel in power consumption. Apple again used (and still uses today) emulation technology to manage the transition from PowerPC code to Intel code. Intel Mac computers are able to seamlessly run PowerPC MacOS X programs under the emulator. Again, Microsoft was slow in providing Intel version of their Mac Office software and the Office programs ran slowly until Intel versions of the programs were available.

There are numerous other examples of successful application of emulation technology.

The Junior design problem involves the development of an emulator for IEEE single precision and double precision floating point operations of addition, subtraction, multiplication, and division.

There will be two teams; Group 1 will produce emulation of single precision floating point operations and Group 2 will produce emulation of double precision floating point operations.

Here are the teams (as determined by J):

```bash
names =: 'Christopher Cavin';'Ansell Chui';'Christopher Gunadi';'Anthony Hechler'
names =: names,'Philip Jensen';'Onyekwere Ogba';'John Pavlick'
names =: names,'Patricia Perez';'Philip Repsher';'Cameron Swords';'Elise Thrasher'
names =: names,'Zachary Tupper';'Aaron Welch';'Douglas Wolf'
groups =: 'Group 1';'Group 2'
(// :) (16$ 1 0 0 0 0 0 0) { ((14?14) {>names),>groups

Group 1
Philip Repsher
Philip Jensen
Aaron Welch
Ansell Chui
John Pavlick
Douglas Wolf
Cameron Swords
Groups
Group 2
Christopher Gunadi
Zachary Tupper
Patricia Perez
Anthony Hechler
Christopher Cavin
Elise Thrasher
Onyekwere Ogba
```

The floating point emulator projects will consist of at least the following phases:

1. Requirements Assessment and Analysis
2. Design
3. Software Selection
4. Prototype Implementation
5. Testing
6. Documentation
Requirements Assessment and Analysis

A requirements document must be submitted to the instructor by February 15, 2010. The floating point emulators must operate on the bit patterns of floating point operands in single and double precision formats and produce single and double precision results according to the IEEE standard. The emulators should include details such as guard bits and rounding bits as specified by the IEEE standard.

Design

Preliminary design documents should be submitted to the instructor by e-mail cs3194@leda.cs.trinity.edu by March 1, 2010. You should include some design narrative and the following UML diagrams:

1. use-case
2. analysis
3. class

Your design should not include a graphical user-interface or other elaborate user features. Use a simple command-line interface which includes commands to display and change memory cells and registers and initiate program operation.

Software Selection

Software selection involves deciding on an appropriate implementation language.

Prototype Implementation

Your floating point emulator should run on available CS Department lab machines.

Testing

Test the prototype emulator in preparation for Junior project presentation on April 5, 2010.

Design Groups

The class should subdivide itself into appropriate groups to accomplish the various tasks above. A project manager must be identified.

Documentation

System documentation must be prepared and submitted by April 26, 2010.

Concluding Remarks