Virginia Tech
Bradley Department of Electrical and Computer Engineering
ECE 3574: Applied Software Engineering ◆ Spring 2013

Homework 5

Submission Details

The solutions for this homework must be submitted as an electronic submission. Your submission will be a tarred and gzipped file containing the source files for the programming problem. These files must be in a directory entitled HW5_*LAST_FIRST* where LAST is your last or family name and FIRST is your first or given name. Your program will be run to evaluate its correctness, and the source code will be reviewed for adherence to Qt programming style. Your program must run on the latest version of Ubuntu and be compiled/built using the GNU C/C++ compiler and qmake. The following information must be included in your source files: your full name, your student ID number, your email address, class (ECE 3574), title of the assignment (Homework 5), and date of submission. You must submit the solutions using the "Submission" button on HW 5 assignment page on the Scholar class web site. The file must be given a name of the following form: HW5_*LAST_FIRST*.tar.gz where LAST and FIRST are as specified above. You can make multiple submissions. Paper and email submissions will not be accepted. All work must be submitted by the announced due date/time.

Honor Code

As stated in the syllabus, in working on homework and projects, discussion and cooperative learning are allowed. However, copying or otherwise using another person's detailed solutions to assigned problems is an honor code violation. See syllabus for details.

Homework 5:

For this homework you will extend the computer simulator that you built for HW 4. This extension consists of two parts. The first part is to add some functionality to the simulator. The second part is to add a cache memory to your computer model. These changes are described in the next two sections.

Added Functionality

Keep the same computer simulation functionality as specified in HW 4. In addition to this functionality, add the following instructions to the instruction set architecture.

LOADA

STOREA <hex-address>

The definitions of each instruction is the following:

- The LOADA instruction treats the value in the accumulator as an address in data memory and loads the value at this address into the accumulator. For example, if the value in the accumulator was 12, and the value stored in data memory location 12 was 129, executing the instruction "LOADA" would result in the value 129 being loaded into the accumulator.
- The STOREA instruction takes the value stored at a data memory address and treats this value as an address (call this value A). The instruction then stores the value in the accumulator at the address given by the value A. For example, suppose the accumulator had the value 13, and that memory location 11 (in hex) had the value 39 in it. The instruction "STOREA 011" would store the value 13 in data memory location 39.

In addition to adding these two instructions to your simulator, you should include the following three features to your simulator GUI.

- You should add a "Reset" capability to your simulator. The "Reset" reinitializes data memory to all zeros, sets the program counter back to 0, sets the accumulator to zero, and takes the computer out of the "halted" state. The "Reset" function allows the user to rerun the program that is stored in program memory.
- You should add a "Clear" capability to you simulator. The "Clear" capability does what "Reset" does and, in addition, clears the instruction memory. For example, after a "Clear", one can read in a new program from the "File" menu and run this new program.
- You should another capability of your choice to your simulator. Examples of possible capabilities include being able to set a "breakpoint" or edit the data or instruction memory values. The capability that you add should be something that would improve the usefulness of the simulator. You need to state what the capability is that you added is in a "README.TXT" file that is included in your program directory.

Adding a Data Memory Cache

The second part of the assignment is to add a data memory cache to the simulator. The data memory cache is an interface between the data memory and the computer. This cache memory should have an associated GUI that displays the state of the cache. The cache GUI and the cache implementation should have the following properties:

1. The cache memory should be configurable from the cache GUI. The cache size should be able to be set to 16, 32, or 64 words of memory. The cache line should be able to be set to 2, 4, or 8 words.
2. The GUI should display the current values of data in the cache. The cache is initially empty. The view of the cache should show the "tags" for each block of cache memory. The view should also show the value of the "dirty" bit for each block of cache.
3. The cache should be direct mapped. Data should be written back to data memory only when a block is replaced (and it is marked as dirty) or when the HALT instruction occurs (in which case the entire cache should be flushed).
4. The cache GUI window should display the current cache efficiency (i.e., the "hit" ratio, the ratio of the number data references that are in the cache divided by the total number of data references in the program).
5. As the program executes instructions, the instructions that interact with memory, e.g., LOAD, STORE, LOADA, STOREA, and ADD, should be displayed in red if it is a cache miss, and in green if it is a cache hit after they execute.

The overall design of this GUI is completely up to you. You may either implement your QUI completely in Qt code, or you can use Qt Designer to design the layout.

Grading Policy

- QT style programming and indentation – 20%.
- The code complies and runs – 0%.
- The code and GUI satisfy the specifications – 80%.