Dalhousie University
Department of Electrical and Computer Engineering

ECED 3403 – Computer Architecture

Lab 3: More debugger commands

1 Objectives

In Lab 1 you wrote a loader and a memory display tool to show the contents of either instruction or data memory. In this lab, you are to add several new commands to the debugger.

The tools developed for this lab can be used with your assignments.

2 Specifications

Lab 3 requires you to design, implement, and test the following debugger commands:

1. A command to display the contents (values) of the eight registers in the register file. The register number and its two-byte hexadecimal value should be displayed.
2. A command to change the contents (value) of a register in the register file. The register number should be specified, and the new value should be entered as a hexadecimal value. This is a useful way of changing the program counter (R7) to begin execution at a different address.
3. A command to change the contents (value) of a memory location in the instruction or data memory. The command will require the memory (instruction or data), address, and new two-byte word value. Both the address and word value should be supplied as hexadecimal numbers.
4. The ability to set a breakpoint at any address in the machine's instruction memory. When the program counter equals the breakpoint value, execution should stop. This tool could be used with Lab 2 to stop execution rather than searching for an instruction with a zero value.

3 Suggestions

Assume that the debugger can see any location in the CPU. There is no need for it to use the instruction or data memory controllers.

The new debugger commands can be added to your loader module from Lab 1.

If your emulator can execute the MOVx instructions (MOVL, MOVLS, MOVLZ, and MOVH), the debugger can inspect the register file to determine if the instructions executed correctly.

You will need a register file, which can be created as:

```
#define REGFILE  8    /* Size of register file */
#define REGCON   2    /* RC bit: REG=0 or CONST=1 */
unsigned short reg_file[REGCON][REGFILE];
```

4 Submission and Marking

4.1 Submission

The assignment will be marked using the following marking scheme:

Design: A brief design is required, consisting of a data dictionary of the major structures and algorithms for decoding and storing.

Software: A fully commented, indented, magic-numberless, tidy piece of software that meets the requirements described above and follows the design description.

Testing: Sufficient test results must be supplied to demonstrate that your implementation meets the specifications. For example, it will be necessary to demonstrate that the different opcodes and operands are recognized.

Your solution must be demonstrated in the lab for it to be marked.

Your design, software, and tests must be submitted electronically to the course website.

4.2 Marking

Marking is as follows:

3: All requirements (design, software, and testing) met.

2: One requirement missing or coding problem.

1: Two or three requirements missing or coding problems.

0: Not submitted or more than three requirements missing or coding problems.

Undemonstrated solutions will not be marked.

5 Important Dates

Available: 5 June 2024 at noon, Atlantic.

Submission due no later than: 7 June 2024 at 11:59pm, Atlantic.

6 Miscellaneous

This lab is to be completed individually.

If you are having *any* difficulty with this or any part of the course, *please* contact Dr. Hughes or Emad as soon as possible.