Computer Organization II, 4th Year, Winter 2010
Lab 1: CPU Sim

Due date: 2 weeks of your lab date

1 Introduction

CPU Sim is an interactive computer simulation package where you specify the details of the CPU to be simulated, including the register set, memory, the microinstruction set, machine instruction set, and assembly language instructions.

CPU Sim executes a program through repeated execution of machine cycles. One cycle has two parts: the fetch sequence, which is a sequence of microinstructions that typically loads the next instruction to be executed into an instruction register and decodes it, and the execute sequence, which is the sequence of microinstructions associated with the newly-decoded instruction. Execution of machine cycles halts when a halt bit is set to 1.

In this lab, you will identify the specifications of the hypothetical machine you want to simulate on CPU Sim. Using these specifications, you will create the machine hardware components, microinstructions, and instruction set. You will write a program for the hypothetical machine and the assembly language you created, run the program, and debug through it.

2 Objectives

1. To enumerate different specifications of a machine.
2. To create a new machine on CPU Sim with a given set of specifications in a correct way.
3. To create an instruction set on a simulated machine on CPU Sim in a correct way.
4. To write an assembly language program using a given instruction set on a given machine on CPU Sim in a correct way.
5. To debug through a program on CPU Sim correctly.

3 Install CPU Sim

1. Download the CPUSim3.4.X.zip file onto your hard disk from the CPU Sim web page (http://www.cs.colby.edu/~djskrien/CPUSim/).
2. Unzip the CPUSim3.4.X.zip file. The unzipping process will create a folder called "CPUSim3.4.X" containing 8 documents and a folder with 3 more documents.
3. Install Java 1.5 or later on your machine if it is not already there. Go to the web site http://java.sun.com/ to get the software and to read installation instructions.
4. Fix, if necessary, your PATH system variable so that it includes the directory (installed as part of the Java installation) in which the java interpreter application is located.

4 Procedure

You will use CPU Sim to run a program on a hypothetical machine. You will (a) create a hypothetical machine, Wombat1, and save it to a file, (b) write an assembly language program for the Wombat1, (c) assemble the program, (d) load it into the Wombat1's memory, and (e) run it.

4.1 Start up CPU Sim

Double-click on the "CPUSim.bat" file.
The window that will appear is the main "desktop" window for CPU Sim.

4.2 Create a New Machine

You will construct a hypothetical machine, name it Wombat1, with the basic structure shown in Figure 1 (The arrows in the figure indicate the movement of data by the microinstructions).

4.2.1 Identify specifications

1. **Registers**: Six registers as following:
   acc (accumulator), 16 bits.
   pc (program counter), 12 bits.
   ir (instruction register), 16 bits.
   mar (memory address register), 12 bits.

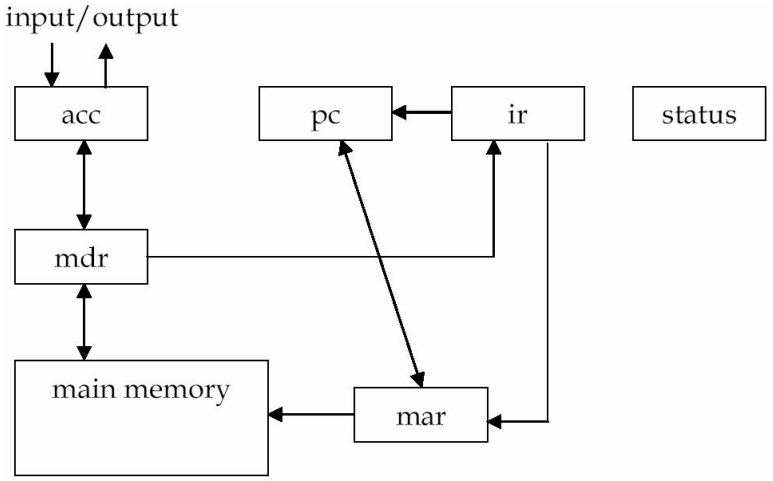

Figure 1: Diagram of data movement in the Wombat1.

mdr (memory data register), 16 bits.
status (status register), 3 bits.

2. **Main memory**: 128 bytes.

3. **Instruction set**: 12 machine instructions, each associated with a 4-bit opcode. They are shown in Table 1.

4.2.2 Create hardware modules

1. Choose "New machine" from the File menu. The default new machine has no registers, no memory, and no machine instructions.

2. Choose "Hardware Modules..." from the Modify menu. The dialog box that appears allows you to edit the registers, register arrays, condition bits, and memory of a machine.

3. Select "Register" as the type of module and then click the "New" button six times. Edit the name and width of each of the six registers so that they match the specifications of the Wombat1 given in the previous section (see Figure 2).

4. Select "RAM" from the popup menu at the top of the Edit Modules dialog. Click "New" and set the name to "Main" and length to 128 for the new memory component.

Table 1: Instruction set of the hypothetical machine to create

Mnemonic	**Description**
HALT	Stop.
READ	Get input from the user.
WRITE	Send output to the user.
LOAD	Transfer data from the main memory to the acc register.
STORE	Transfer data from the acc register to the main memory.
ADD	Add a value from the main memory to the value in the acc register, putting the result in the acc register.
SUBTRACT	Similar to ADD.
MULTIPLY	Similar to ADD.
DIVIDE	Similar to ADD.
JMPZ	If the value in the acc register is 0, jump to a new location to obtain the next instruction to be executed.
JMPN	If the value in the acc register is less than 0, jump to a new location to obtain the next instruction to be executed.
JUMP	Jump to a new location unconditionally.

5. Select "ConditionBit" from the popup menu at the top of the dialog. Click "New" and then set the name of the condition bit to "halt-bit", the register to the status register, the bit to 0, and check the "halt" box.

6. Close the Edit Modules dialog box by clicking the "OK" button.

4.2.3 Create microinstructions

Now we need to construct the necessary microinstructions that will be used to implement the machine instructions. There are 7 transfer, 4 arithmetic, 2 test, 1 increment, 1 decode, 2 io, 2 memory access, and 1 set condition-bit microinstructions that need to be created. The 7 transfer microinstructions are displayed by arrows between the registers in Figure 1.

Create the transfer microinstructions:
You will create "TransferRtoR" type microinstructions. They are for transferring a contiguous set of bits from one register to another register.

1. Choose the menu item "Microinstructions" from the Modify menu.

2. Choose "TransferRtoR" in the popup menu at the top of the dialog that appears (see Figure 3).

3. Click the "New" button.

4. Edit this microinstruction so that it transfers the contents of the pc to the mar. Click in the table cell with the "?" in it (in the column headed "name") and type in "pc->mar" (you can give the microinstructions any

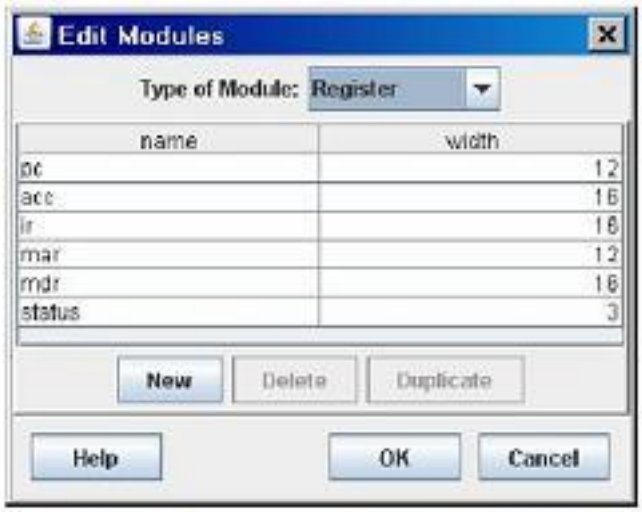

Figure 2: The dialog for editing the hardware modules.

name you want, but it helps to choose something descriptive). Select the "pc" as the "source" register and the "mar" as the "destination" register, with bit 0 as the start bit for both registers and give numBits the value 12. This means that we want to transfer all 12 bits between the registers.

5. Create the remaining 6 transfer microinstructions in the Wombat1 by adding new microinstructions in the same manner (see Figure 3).

6. The microinstructions of the other types needed for the Wombat1 can be created similarly by first selecting the appropriate type of microinstruction in the popup menu at the top of the dialog and then creating the microinstructions of that type.

7. When you are finished, close the Edit Microinstructions dialog box by clicking "OK".

Question 1: `In a table, for every microinstruction of the remaining 11 you create, record the values you use for its parameters.`

4.2.4 Construct the fetch sequence

1. Choose "Fetch Sequence" from the Modify menu.

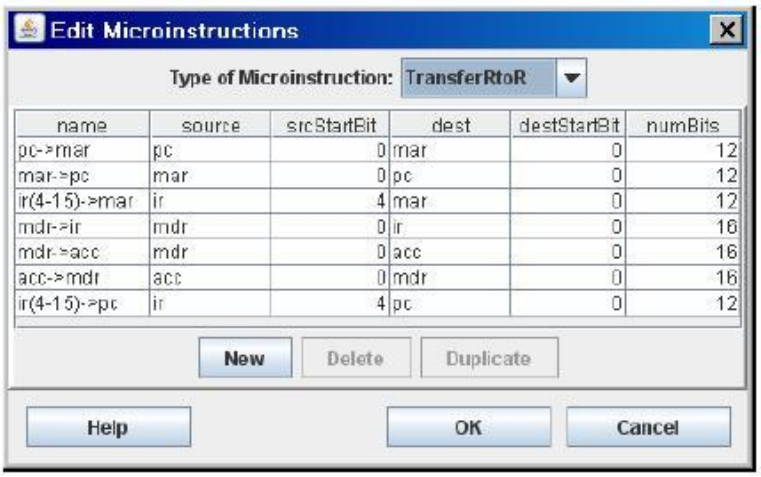

Figure 3: The dialog for editing microinstructions.

2. The current fetch sequence list is displayed in the left scroll box entitled "Implementation". To add microinstructions to it, first display the microinstructions of the type you want in the tree of microinstructions in the right side of the dialog by clicking on the dial to the right of the folder corresponding to the desired type. Then highlight the microinstruction you want by clicking on it.

3. To specify where the new microinstruction is to be inserted into the fetch sequence, click on the microinstruction in the left scroll box before which you want to insert the new microinstruction and click "<<insert<<" (see Figure 4).

4. To remove a microinstruction from the fetch sequence, click on it in the left scroll box to highlight it, and then click ">>delete". You can reorder the microinstructions in the list on the left by dragging them up or down to a new position.

5. When you are done, click "OK".

4.2.5 Create the instruction set

The name you specify for a machine instruction is used in assembly language programs to execute that instruction. The opcode is specified in hexadecimal notation. The field lengths give the number of bits in each field of the instruction, including the opcode field (the first field) and all operand fields. For example, if an instruction has field lengths of 4 and 12, then the opcode occupies

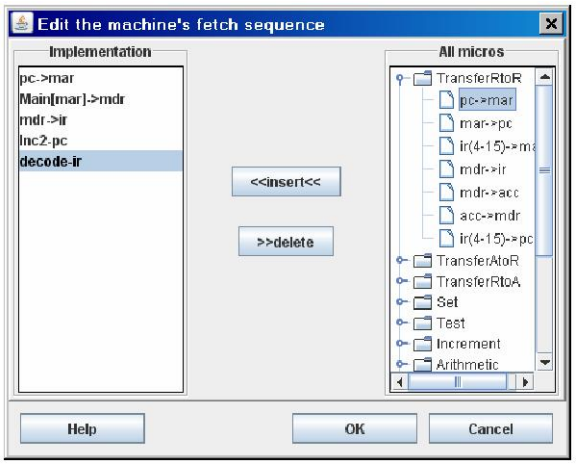

Figure 4: The dialog for editing the fetch sequence.

the leftmost 4 bits and the remaining 12 bits of the instruction form the only operand. All Wombat1 machine instructions have 4-bit and 12-bit fields, but for some of the instructions, the 12-bit field is ignored (such fields are denoted by putting parentheses around them in the fieldLengths column).

1. Choose "Machine Instructions..." from the Modify menu. The dialog box that appears allows you to edit machine instructions, including the name, opcode, field lengths, and the list of microinstructions that form the execution sequence of each instruction.

2. To create a new machine instruction, click "New" and then edit the parameters of that instruction (see Figure 5). The "Opcode" is a non-negative integer (in hexadecimal) that fits in the first field of the instruction. The "Field Lengths" is a list of 1 or more positive integers whose sum is a multiple of 8.

3. Once all machine instructions have been created, click "OK".

Question 2: `In a table, for every instruction you create, record the values you use for its parameters.`

4.2.6 Save your machine to a file

1. Choose "Save machine" or "Save machine as" from the File menu. Use the filename: Wombat1.cpu.

2. CPU Sim saves the details of the machine as text in an XML file.

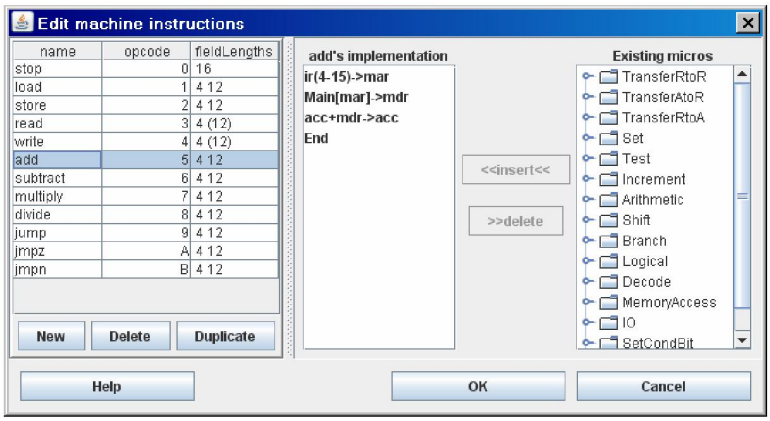

Figure 5: The dialog for editing machine instructions.

3. Close CPU Sim.

To see all the details of the machine in a more user-friendly format than XML, choose "Save machine in HTML" from the File menu. You will be asked to type in the name of the HTML file into which the data will be saved. This file can be opened and viewed with any web browser.

4.3 Run a Program

1. Start up CPU Sim.
2. Choose "Open machine" from the File menu.
3. Select and open the text file you named for the Wombat1 machine. If the machine loads without error, then the title of the main display should become "Wombat1" and two windows should appear inside the desktop (see Figure 1).
4. Choose "New text" from the File menu.
5. In the text window, write the following program.

```
; This program reads in integers and adds them together
; until a negative number is read in.  Then it outputs
; the sum (not including the last number).
start:    read          ; read n -> acc
          jmpn done     ; jump to done if n < 0
          add sum       ; add sum to the acc
          store sum     ; store the new sum
```

```
          jump start   ; go back & read in next number
done:     load sum     ; load the final sum
          write        ; write the final sum
          stop         ; stop
sum:      .data 2 0    ; 2-byte location where sum is stored
```

6. Save your program by selecting "Save text" from the File menu. Name it W1.

7. Choose "Assemble & Load" from the Execute menu. You should see numbers and comments appear in the first few rows of the table in the RAM window. The numbers in the data column of the RAM window are the machine language instructions generated by the assembler from the assembly language program. The comments column of the RAM window displays the lines of assembly code from which the corresponding machine instruction was generated.

8. Make sure all the registers have been cleared (set to 0). If some of them are not 0, then either edit the values to make them 0 or choose "Clear all registers & arrays" from the Execute menu.

9. Choose "Run" from the Execute menu. The program will begin execution with the instruction whose address (namely, 0) is stored in the program counter pc. The machine runs by repeatedly executing machine cycles consisting of the fetch sequence, which loads into the ir the instruction whose address is in the pc and then decodes the instruction, followed by the execute sequence, which executes the machine instruction that was just decoded.

10. When the console panel asks you for input, type a positive integer and press the enter key. Repeat this process several times and then type in a negative number.

11. When execution is complete, watch the final state of the registers and RAM in the registers and RAM windows.
Question 3: `Record these results in a table.`

12. To rerun the program with different inputs, select "Clear, assemble, load & run" from the Execute menu.

13. To quit CPU Sim, choose "Quit" from the File menu.

4.4 Debug a Program

1. Load the Wombat1 machine and the W1 program.

2. Select "Clear everything" from the Execute menu.

Figure 6: The debugging toolbar.

3. Choose "Assemble & load" from the Execute menu.
4. Select "Debug Mode" from the Execute menu. You will see a toolbar appear at the top of the display (see Figure 6).
5. On its right end, the debugging toolbar says that the next instruction is the fetch sequence. Click "Step by Instr" from the toolbar, then a complete machine cycle will be executed.
6. The "read" instruction will be fetched, decoded, and executed. Type the inputs in the console and press enter.
7. Each click of "Step by Instr" causes a full machine cycle to be executed. **Question 4:** `In a table, record the values of the registers after executing every instruction.`
8. To quit CPU Sim, just choose "Quit" from the File menu.

5 Deliverables

In your lab report, answer **Questions** 1 through 4 mentioned in the Procedure section. Submit your report to the TA within 2 weeks of your lab date. Submit just one lab report for your group.