# CMPE 325 - Computer Architecture II

## EXPERIMENT 1

### Introduction to PCSpim (MIPS R2000 Simulator)

# 1.Introduction

PCSpim is a simulator that runs programs for the MIPS R2000/R3000 RISC computers. SPIM can read and immediately execute files containing assembly language or MIPS executable files. SPIM is a self-contained system for running MIPS programs.

In the experiments, we will use a simulator instead of a workstation with a MIPS processor, because a simulator provides us miscellaneous features in understanding the instruction set as well as in debugging. Moreover, a MIPS simulator is available for almost any computer- and operating-system. Furthermore, the simulator can be updated to include the new features, instructions or pseudo-instructions developed in later versions of the processor for almost without any additional cost.

The MIPS simulator is called SPIM, and available as a freeware from **www.mkp.com/cod2e.htm**. SPIM is written for most of the commonly used workstations such as SUN, VAX, IBM-PC compatibles, and Windows. PC-SPIM is a combined assembler-loader-processor-memory and I/O simulation for MIPS-2000. It is a sophisticated program with miscellaneous command line options and in-line assembler commands. It is easy to install in Windows 32-bit operating systems.

#### MIPS ASSEMBLER SYNTAX

**Comments** in assembler files begin with a sharp sign " # ". Everything starting from the sharp sign to the end of the line is ignored.

**Identifiers** are a sequence of alphanumeric characters, underbars " _ ", and dots " . " that do not begin with a number. Instruction opcodes are reserved words that cannot be used as identifiers. Labels are declared by putting them at the beginning of a line followed by a colon, for example:

```
      .data
item: .word 1
      .text
      .globl main   # must be global.
main: lw $t01,item  # loads temp.reg. $t01 with item.
      .....
```

Numbers are base 10 by default. If they are preceded by 0x, they are interpreted as hexadecimal. Hence, 256 and 0x100 denote the same value.

Strings are enclosed in doublequote "...". Special characters in strings follow the C convention: i.e., newline is \n, tab \t, and quote \"

Some important SPIM (and also MIPS) assembler directives:

**`.byte b1,...,bn`** # store n specified values to the memory.

**`.data <address>`** # set data segment address.

SPIM uses 0x10000000 as the beginning of the data segment. Set it to 0x10000000 to have correctly matching data labels to their addresses.

**`.globl sym`** # makes label globally accessible.

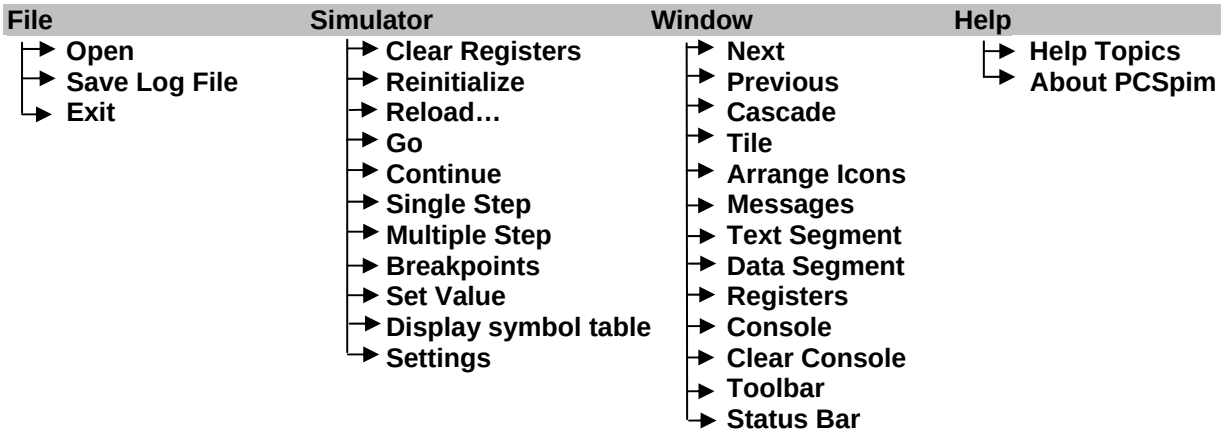
**`.space n`** # allocates n bytes of space in the current segment.

**`.text <address>`** # subsequent items are put in the user text segment.

The items in text segment may be only words, or instructions.

**`.word n`** # stores the listed values of words into the memory.

The PCSpim simulator program has a pull down menu appearance as shown below:




## How to Set The Simulator:

In the first part of this experiment you will use SPIM to simulate

① a bare MIPS machine,

② without allowing pseudo-codes, and

③ no mapped I/O option.

④ without loading any trap-features.

In this mode, the assembler will not allow any pseudo-codes (i.e., **`li`**, **`mul`**, blt, ... etc. and also any long offset fields in the lw and sw instructions) to be used in your program.

For the convenience in reading the register contents you may prefer to have hexadecimal readings in the display windows.

In the second part, you will use SPIM in the more elaborated mode with the pseudo-code and trap-features loaded.

All of these settings can be set on the settings form that is accessed starting from the drop-down menu by simulator>settings.

## Organization of a MIPS assembly program

The MIPS assembly programs are text files with the extension **".s"** or **".asm"**. SPIM has no built-in editor-program for writing the assembly source. You have to use your favorite, or an available text editor such as the NOTEPAD.EXE. The Notepad program of Windows 95/98 or NT is located in:

**Start→Programs→Accessories→Notepad**

Type your MIPS assembly program (you have to leave an empty line at the end of the program) and save it by specifying a filename for your program. Note that the extension of the filename must be “.s” or ".asm".

You should first click on the PCSpim for Windows icon to start the PCSpim main window. Then load your program by using the PCSpim’s menu File > Open. Use the opened browser to choose the path and the assembly source file that you want to open.

If there is any syntax or structure error in your file, SPIM will give you a message indicating the line number and the reason of the rejected line. You have to clean your program from the syntax bugs and load it to SPIM.

After loading your assembly source, you are ready to run or trace it. Use Simulator > Go (F5-key) or Simulator > single step (F10-key) of the main menu. The starting address is automatically defined by the compiler according to the options you set-up on the settings window.

You can watch the contents of the registers using window > Registers of the main menu. For an easy to observe page organization try the window > tile option.

# 2. Experimental Work

## Part-1

Following program multiplies two unsigned integers in the registers R8 by R9 and writes the 32-bit product to register R10. In order to understand the operation of your simulator program, type to a file named "exp1a.asm" and execute the following MIPS assembly program in non-pseudo-instruction mode.

```
  .data 0x10000000
  .text 0x00400000
  .globl main
main:
  addi $8,$0,6
  addi $9,$0,12
# multiplication of $8 * $9 -> $10
  add  $2,$0,$8
  add  $10,$0,$0
mulloop:
  beq  $2,$0,mulexit # if zero exit
  addi $2,$2,-1
  add  $10,$10,$9
  j    mulloop
mulexit:
# multiplication loop is over,
#    is the result in $10 correct?
  sll  $0,$0,0
  syscall
```

① You can put comments to the end of a line after a sharp sign (#).

② You can start the single step execution applying the following items.

- First set the PC (prog.counter) to the starting address of the program

  If SPIM is set correctly the starting address is 0x00400000.

  To set the value use the key-sequence alt-s,v (or menu simulator>set value) to

  open the register-value assignment dialog box. Enter PC and the starting address in hexadecimal format.

- Next, use the fn10 key to execute one instruction at each key-press. You can also use the fn5 key to execute the complete progam at once. Correct the starting address to 0x00400000 before clicking the OK button.

③ After **`syscall`** stops the execution save the log file with the filename "exp1a.log". Open the log file by dragging it into the textpad and inspect the text segment. Fill in the following machine code table according to hexadecimal machine codes assigned by SPIM.

## Part-2

In this part, you will use the SPIM in pseudo-code allowing mode.

① Clear the bare machine setting, and check only the allow pseudo-code option in the settings dialog-box (key-sequence alt-s,s).

② Write the following text to a file named "exp1b.asm".

```
  .data
  .text
  .globl main
main:
  li   $8,0x3210
  li   $9,0x76543210
  sge  $11,$8,$9
  mul  $12,$11,$10
infloop:
  bge  $11,$0,infloop
  syscall
```

③ Load the file to SPIM, and watch the corresponding machine codes of each line. Use the log file to fill in the following binary-machine-code table to understand the fields of each instruction in a better manner.

Note that the given text is not a program, it is not traceable. It contains a sample of some commonly used MIPS pseudo-instructions.

# Reporting

Before the Lab-time is over, fill in the following report page as soon as you complete the laboratory work, and submit it to your assistant. Your report is important for your grading.

Name: _____ Student Number:_____

Submitted to (Asst.): _____ Date:dd/mm/yy ___/___/___


**EASTERN MEDITERRANEAN UNIVERSITY**

**COMPUTER ENGINEERING DEPARTMENT**

**Fall 2007-08**

# CMPE 325 - Computer Architecture II

## EXPERIMENT 1 - Reporting Sheet

Part One: The observed binary machine codes of the instructions are:

| Instruction | opc | rs | rt | rd | sa | fn |
|---|---|---|---|---|---|---|
| addi $8,$0,6 | | | | | | |
| addi $9,$0,12 | | | | | | |
| add $2,$0,$8 | | | | | | |
| add $10,$0,$0 | | | | | | |
| beq $2,$0,mulexit | | | | | | |
| addi $2,$2,-1 | | | | | | |
| add $10,$10,$9 | | | | | | |
| j mulloop | | | | | | |
| sll $0,$0,0 | | | | | | |

Part 2: The observed binary machine codes of the instructions are:

| Instruction | opc | rs | rt | rd | sa | fn |
|---|---|---|---|---|---|---|
| li $8,0x3210 | | | | | | |
| | | | | | | |
| li $9,0x76543210 | | | | | | |
| | | | | | | |
| | | | | | | |
| sge $11,$8,$9 | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| mul $12,$11,$10 | | | | | | |
| | | | | | | |
| | | | | | | |
| bge $11,$0,infloop | | | | | | |
| | | | | | | |
| | | | | | | |

Grading: Lab Performance :

Asst. Observations :