Computer Architecture Lab

Lab 1

Introduction to QtSpim

Objectives

After completing this experiment, you will be able to:

- How to use QtSpim for simulation

For each of the programming exercises, demonstrate your program to the instructor, format and comment your program appropriately.

Materials Needed

- Read ***Assemblers_Linkers_and_the_SPIM_simulator.pdf***, ***Hello.asm,*** ***QtSpim-Tutorial.pdf***, and ***MIPS_ short.pdf***

Procedures

P.1. Introduction to SPIM

1. SPIM

Spim is a self-contained simulator that runs MIPS32 programs. It reads and executes assembly language programs written for this processor. Spim also provides a simple debugger and minimal set of operating system services. Spim does not execute binary (compiled) programs.

Download and install the newst version of Spim called QtSpim from: http://spimsimulator.sourceforge.net/

When you open QtSpim, A window will open as shown in Figure 1. The window is divided into different sections:

1. The Register tabs display the content of all registers.

2. Buttons across the top are used to load and run a simulation

3. The Text tab displays the MIPS instructions loaded into memory to be executed. (From left-to-right, the memory address of an instruction, the contents of the address in hex, the actual MIPS instructions – where register numbers are used, the

MIPS assembly that you wrote, and any comments you made in your code are displayed.)

4. The Data tab displays memory addresses and their values in the data and stack segments of the memory.

5. The Information Console lists the actions performed by the simulator

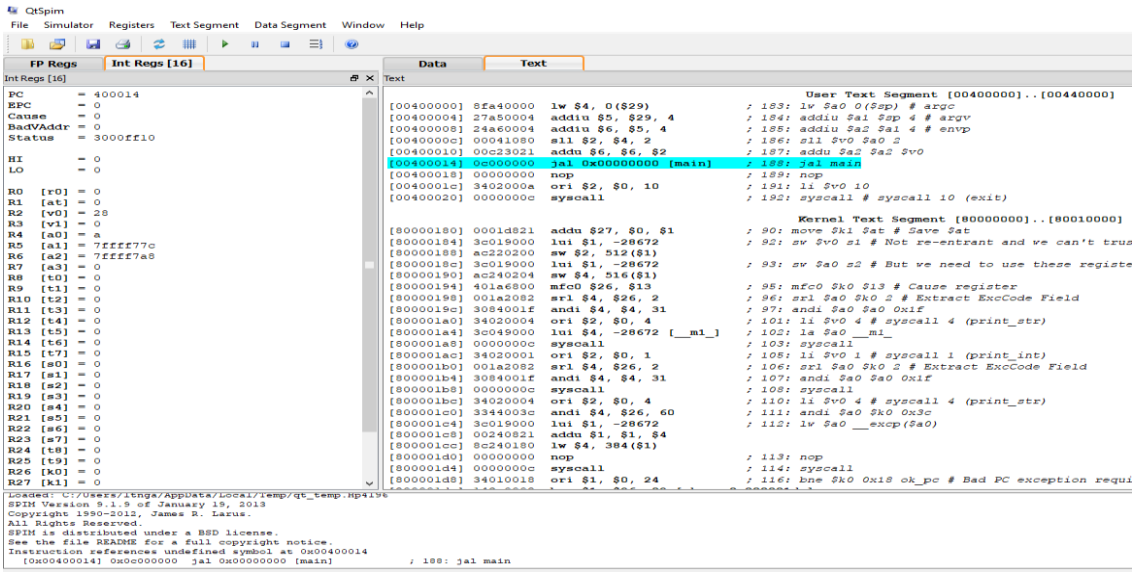


Figure 1: QtSpim interface

2. QtSpim

QtSpim has three windows:

- Left Window: Registers
- Text Window: Code
- Data Window: Data, Stack, and Kernel data

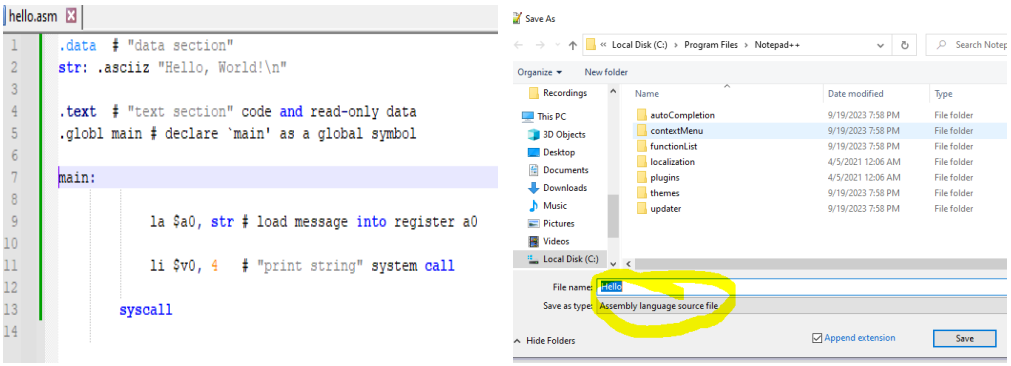
Memory organization:

0x00400000	Code
0x10000000 – 0x10040000	Data
0x7fffeffc	Stack
0x80000180	Kernel code
0x90000000	Kernel data

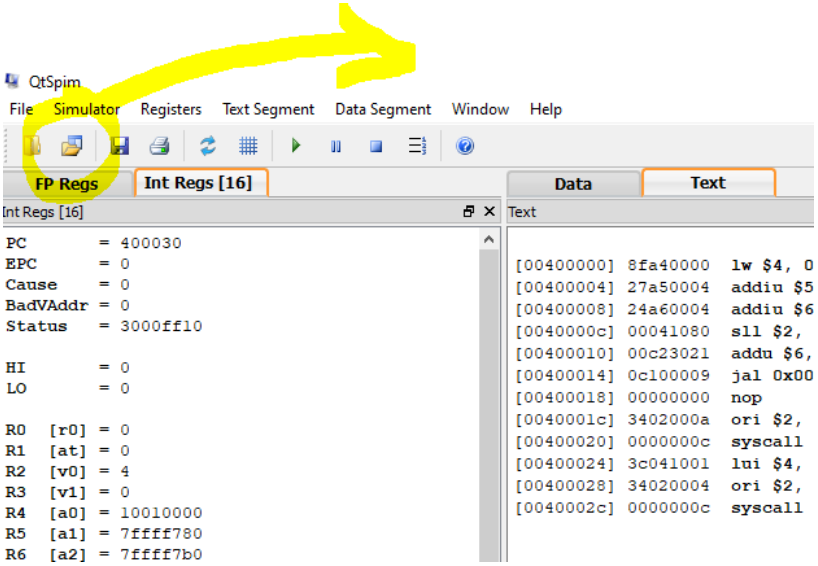
Please capture your screen in here for each row of memory organization

3. To run the program in QtSpim:

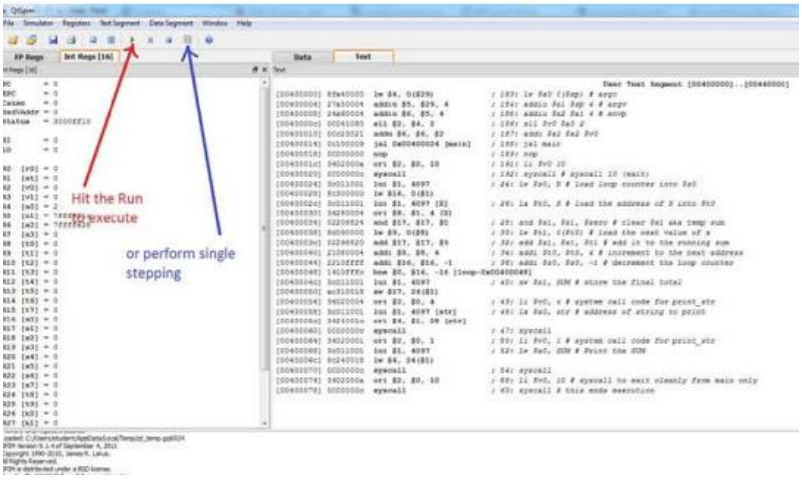
1. Use a text editor/ Notepad++/ to create your program Hello.s or Hello.asm as follows



2. Open QtSpim, **File > Reinitialize and Load File** to open Hello.s



- **Simulator > Run** or press **F5** to run the program



3. You can then run the program by simply pressing the “run” (play) button – all instructions will be executed, and the final contents of memory and the register file will be reflected in the QtSpim window. The code will print “Hello, MIPS” on Console Window

Computer Architecture Lab

```
hello.s
1           .data       # the data segment
2   msg:    .asciiz "Hello, MIPS\n"
3
4           .text       # the code segment
5           .globl main
6   main:
7       la $a0, msg     # load the argument string
8       li $v0, 4       # load the system call (print)
9       syscall         # print the string
10      jr $ra          # return to caller (__start)
11
```

P.2 Question and Tasks

Questions and Tasks:

```
                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [msg]        ; 7: la $a0, msg # load the argument string
[00400028] 34020004  ori $2, $0, 4             ; 8: li $v0, 4 # load the system call (print)
[0040002c] 0000000c  syscall                   ; 9: syscall # print the string
[00400030] 03e00008  jr $31                    ; 10: jr $ra # return to caller (__start)
```

3.1 The code was loaded into memory. Determine the memory address where the code resides.

3.2 Using **Single Step** button or **F10** to step through the code (execute the code one instruction at a time). In the Text Window, one instruction is highlighted after every step. Is this instruction the current executing instruction or the next instruction? How can you know that? Hint: there is a register indicates which instruction will be executed next.

3.3 The string “Hello, MIPS\n” was loaded into memory. Determine its location in memory. Show the content of the memory segment which stores that string (in hex). Hint: look into Data Window.

3.4 Create an assembly code to print out your full name and student ID on separate lines.

.

4. Load the program ***lab1_integer_double.s*** and run it in QtSpim. Remember to refresh register after each run.
   4.1 What is the output when you enter the integer ***343523343532***? Is the output correct? If not, explain what happened.

   4.2 What is the maximum and minimum input values that program can calculate correctly?

   4.3 Modify the program so that it prints newline after outputting the result. Save your assembly as ***lab1_integer_double_01.s***

   4.4 Modify your program so that it print out the value of the ***first byte*** of the input. Test your program with the input ***1463423***. Save your program as ***lab1_integer_double_02.s***

5. Write an assembly that reads in 2 integers and prints out the result of addition and subtraction of those 2 integers. Save the assembly as ***lab1_twointegers.s***

6. Write a MIPS program to reverse the elements of an array of 20 elements.
7. Given a 15 elements array. Let the user choose between two modes:
   a. Print the value of the element chosen by the user (the user is required to enter an index number from 0 to 14).
   b. Print a sequence of values from the elements chosen by the user (the user is required to enter two index numbers from 0 to 14, the first number must be smaller or equal to the second number.