Lab 1 – Tool Installation and an Introduction to QtSpim
CECS440 – Spring 2021

Part I

Overview

In this first part you will download, install and start to become familiar with two tools that you will be using throughout the semester. The tools are 1) Vivado CAD design tools and 2) QtSPIM simulator. You can download these tools on your laptop and/or your home computer for your use in home. They are very useful and fun to use, but like any fairly complex set of tools, will require a little study and practice on your part to master their capabilities.

QtSPIM

QtSPIM is the latest version of the popular SPIM simulator. SPIM (which is MIPS spelled backwards) is a program that can read and execute assembly language programs written for the MIPS32 processor used in our textbook. The simulator was written and is still maintained by Jim Larus.

Your task for this part

Follow the QtSPIM link to download and install QtSPIM on your personal computer. Make sure that it opens and executes on your computer. We will use this in a few weeks, so nothing needs to be turned in for this lab. However those interested in finding out a little more about QtSPIM at this point can jump ahead and read SPIM documentation and assembly directives (PDF) for more information on SPIM.

Xilinx Vivado Webpack

We will be using the "WebPack" version of Xilinx's Vivado 2015.4 CAD tools. The Webpack version of the tools are free and can be downloaded from Xilinx [Vivado 2015.4]. The tools run on both Windows and Linux but not natively on a Mac. If you have a Mac, you will have to run the tools in a virtual machine. Oracle's Virtualbox VM is a free VM that works quite well and is the recommended VM. The TA's will help with the installation during your lab.

Note* The Webpack you will use on your desktop may be slightly different version from the CAD tool you will use in the lab. The TA's will explain the differences when using the version in the lab. Both versions operate in the same fashion, but the menu's you will use to enter a design and run the simulator (ISim) may be slightly different.

Your task for this part

You task here is to downloaded the Xilinx [Vivado 2015.4] and install the tools. The file is approx. 2.7GB so may take a little time to download. Once downloaded you will then need to run the installer. This will also take some time. Once installed you will then need to obtain a free license from Xilinx to run the tools. The first time you use the tools they will guide you through process of obtaining the free license.

Part II

General Information.

1- Register conventions. There are 32 general-purpose registers. Registers are always preceded by a $. There are two ways to refer to a register:

- using the actual register number e.g. $0 through $31
- or using the registers equivalent name e.g. $t1, $sp

Register Number	Alternative Name	Description
0	zero	the value 0
1	$at	(assembler temporary) reserved by the assembler
2-3	$v0 - $v1	(values) from expression evaluation and function results
4-7	$a0 - $a3	(arguments) First four parameters for subroutine. Not preserved across procedure calls
8-15	$t0 - $t7	(temporaries) Caller saved if needed. Subroutines can use w/out saving. Not preserved across procedure calls
16-23	$s0 - $s7	(saved values) - Callee saved. A subroutine using one of these must save original and restore it before exiting. Preserved across procedure calls
24-25	$t8 - $t9	(temporaries) Caller saved if needed. Subroutines can use w/out saving. These are in addition to $t0 - $t7 above. Not preserved across procedure calls.
26-27	$k0 - $k1	reserved for use by the interrupt/trap handler
28	$gp	global pointer. Points to the middle of the 64K block of memory in the static data segment.
29	$sp	stack pointer Points to last location on the stack.
30	$s8/$fp	saved value / frame pointer Preserved across procedure calls
31	$ra	return address

Each program has two sections: .data and .text.

- The .data section is where variables are declared.
- The .text is for your code. Your code needs to begin with a "main".

```
# Comment giving name of program and description of function
# Template.s
# Bare-bones outline of MIPS assembly language program

    #variable declarations follow this line
    #Format would be  like label:   storage_type    value(s)
    .data

        #create a single integer variable with initial value 3
        var1:       .word   3
        #create 2-element character array with elements initialized to a & b
        array1:     .byte   'a','b'
        # allocate 40 consecutive bytes, with storage uninitialized
        array2:     .space  40

    .text   # instructions follow this line

        # indicates start of code (first instruction to execute)
        main:


        Func1:


        Func2:
```

System Calls

SPIM provides a small set of operating-system-like services through the system call "syscall" instruction. To request a service, a program:

- loads the system call # into register $v0
- loads arguments into registers $a0–$a3
- System calls that return values put results in register $v0.

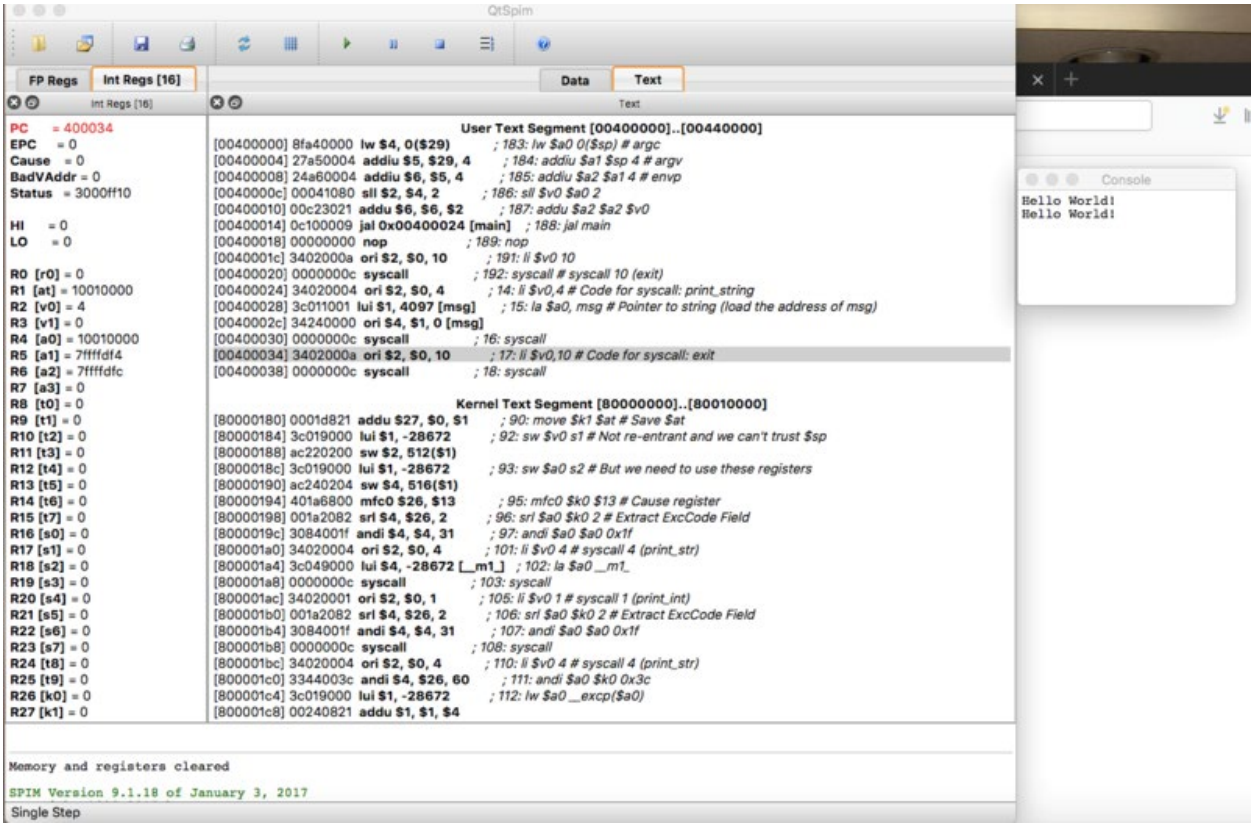
Here is an example. the following code uses syscalls to print “the answer = 5"

```
    .data

        str: .asciiz  "the answer = "

    .text

    main:
        li       $v0, 4    # system call code for print_str
        la       $a0, str  # address of string to print
        syscall            # print the string
        li       $v0, 1    # system call code for print_int
        li       $a0, 5    # integer to print
        syscall            # print it
```

The list of system calls is given below.

Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

QtSpim Simulator



The screen shot above shows the command line, register and text panes from QtSPIM.

- Command Line: Icon links to commands for the simulator are provided across the top of the window. The first icon on the left is "load file". Clicking this icon will open up a window in which you can select a file to be loaded into the simulator. Icons are also provided for running and resetting the simulator.
- Registers: The register pane shows the values of contained within the registers. You can see how the values change as the program executes.
- Text: The text pane shows three columns of information. The first column lists the memory addresses of where each instruction in your program resides in memory. The second column shows the actual machine code version of each instruction. The third column lists the assembly code for each instruction. The instruction that is executing will be highlighted as you single step through the program.
- Console: The little window on the right shows the console. You can type input for the program in the console and the output for the program will be shown in the console.
- Data: The data pane is hidden in this screen shot. Clicking on the data tab will show the operands and their values in data memory.

Assignment

- 1. Download and run the Test Program to get started and test QtSPIM. Experiment with this program to understand how to run and reset the simulator. Explore what types of information you can get from the register, text, and data panes of the simulator.
- Use the simulator to answer the following questions.
  - Q1.1:Single step through the program and try to figure out what each instruction is doing.
  - Q1.2:Which instructions change values in the registers?
  - Q1.3:Which instructions change values in the data memory?
  - Q1.4:Which instructions produce output to the console?
- 2. Download and run the hello world program.
  - Q2.1:Change the program to print out your name in place of "hello world" .
- 3. Download and run simple add
  - Q3.1:Modify the code to additionally compute "A or B=" and "A and B=" and output the results.

Lab Write-up/Video

Your lab writeup should include the following:

- 1. For the Test Program:
  - Show a screen shot of QtSpim with the program loaded.
  - How many instructions in total are executed?
  - What registers are used?
  - What addresses in Memory are changed?
  - List and explain the syscalls that were used.
- 2. For the hello world program:
  - Show a screen shot with your name output instead of hello world
- 3. For the simple add program:
  - Show a screen shot for the two outputs ("or", "and")