Assignment1 - Due: Wednesday, August 26th, 5pm

Assignment

CSE/EEE230 Assignment1

Due Date

Wednesday, August 26th, 5pm

Important: This is an individual assignment. Please do not collaborate.

It must be submitted on-line through the course website.

Go to "GradeScope" tab on Canvas -> CSE/EEE230 -> Assignment1, and upload your program file.

No late assignment will be accepted

Minimal Submitted Files

You are required to turn in the following source file:

assignment1c.s

Objectives:

This assignment is for you to become familiar with the QTSpim simulator that we will be using in this course. You can use a machine in BYENG 214 (a computer lab) or install the QTSpim into your own computer, by downloading from:

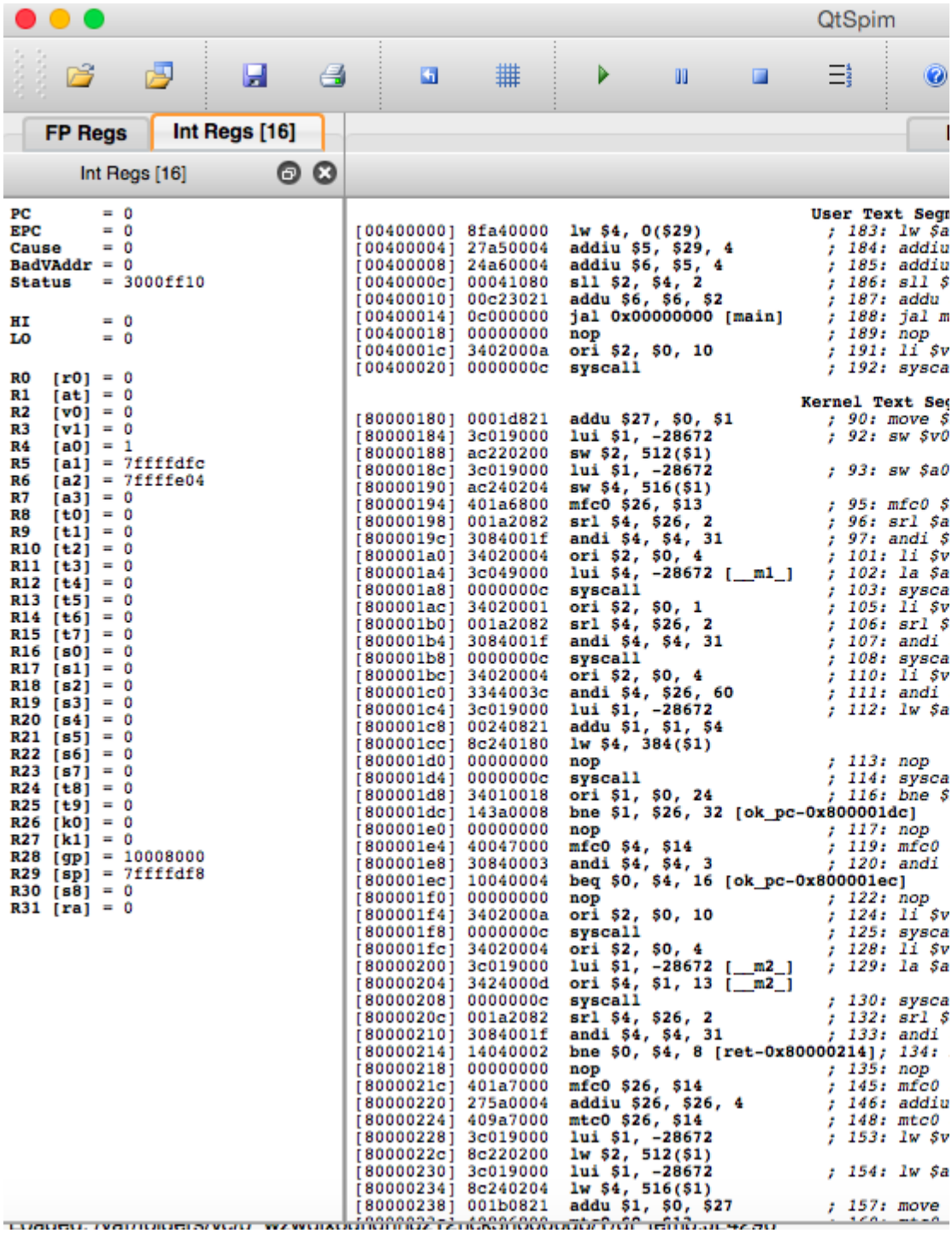
http://sourceforge.net/projects/spimsimulator/files/ (http://sourceforge.net/projects/spimsimulator/files/)

You will edit an assembly language program, load it in the QTSpim, execute it, and step through the program.

Important Note: Your submitted assignment will be tested using SPIM, therefore you should test your program using SPIM before its submission.

Assignment Description:

1. Load the QTSpim program. You can also use other SPIM and you will have a similar, but slightly different interface. You will see the windows looking as follows:


QtSpim
FP Regs
Int Regs [16]
Int Regs [16]
PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7fffffdfc
R6 [a2] = 7fffffe04
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7fffffd8
R30 [s8] = 0
R31 [ra] = 0
User Text Segm
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu
[00400014] 0c000000 jal 0x00000000 [main] ; 188: jal m
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v
[00400020] 0000000c syscall ; 192: sysca
Kernel Text Seg
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a
[800001a8] 0000000c syscall ; 103: sysca
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $
[800001b4] 3084001f andi $4, $4, 31 ; 107: andi
[800001b8] 0000000c syscall ; 108: sysca
[800001bc] 34020004 ori $2, $0, 4 ; 110: li $v
[800001c0] 3344003c andi $4, $26, 60 ; 111: andi
[800001c4] 3c019000 lui $1, -28672 ; 112: lw $a
[800001c8] 00240821 addu $1, $1, $4
[800001cc] 8c240180 lw $4, 384($1)
[800001d0] 00000000 nop ; 113: nop
[800001d4] 0000000c syscall ; 114: sysca
[800001d8] 34010018 ori $1, $0, 24 ; 116: bne $
[800001dc] 143a0008 bne $1, $26, 32 [ok_pc-0x800001dc]
[800001e0] 00000000 nop ; 117: nop
[800001e4] 40047000 mfc0 $4, $14 ; 119: mfc0
[800001e8] 30840003 andi $4, $4, 3 ; 120: andi
[800001ec] 10040004 beq $0, $4, 16 [ok_pc-0x800001ec]
[800001f0] 00000000 nop ; 122: nop
[800001f4] 3402000a ori $2, $0, 10 ; 124: li $v
[800001f8] 0000000c syscall ; 125: sysca
[800001fc] 34020004 ori $2, $0, 4 ; 128: li $v
[80000200] 3c019000 lui $1, -28672 [__m2_] ; 129: la $a
[80000204] 3424000d ori $4, $1, 13 [__m2_]
[80000208] 0000000c syscall ; 130: sysca
[8000020c] 001a2082 srl $4, $26, 2 ; 132: srl $
[80000210] 3084001f andi $4, $4, 31 ; 133: andi
[80000214] 14040002 bne $0, $4, 8 [ret-0x80000214] ; 134:
[80000218] 00000000 nop ; 135: nop
[8000021c] 401a7000 mfc0 $26, $14 ; 145: mfc0
[80000220] 275a0004 addiu $26, $26, 4 ; 146: addiu
[80000224] 409a7000 mtc0 $26, $14 ; 148: mtc0
[80000228] 3c019000 lui $1, -28672 ; 153: lw $v
[8000022c] 8c220200 lw $2, 512($1)
[80000230] 3c019000 lui $1, -28672 ; 154: lw $a
[80000234] 8c240204 lw $4, 516($1)
[80000238] 001b0821 addu $1, $0, $27 ; 157: move

If the console window is not visible, then select *Window -> Console* from the menu in the QTSpim. The console window should be empty.

The QTSpim window is divided into some sections. The left section under Int Regs is the registers section. Registers are storage areas internal to the CPU. They are identified by notations such as $r0-$r31. In QTSpim, registers are 32-bits wide, meaning that they can store a 32-bit value. (Note that there are Int Regs to store integers, and FP Regs to store floating point numbers. We will discuss floating point numbers later in the semester.)

The right section under the tab Text is the text segment. This is where the assembly/machine language code you are executing is displayed. On the left hand side you see values such as [00400000]. These are memory addresses. The simulator begins executing code at 00400000 by default. The next column is the 32-bit machine language instruction stored at that address. It is displayed in hex. The next column is the assembly language statement that corresponds to that 32-bit machine language instruction. The next column (with the ;) is an alternative representation of the statement in column three.

The right section under the tab Data is the data segment. Any data used by your program is stored here. The first column displays a memory address (or range) and the columns display the data stored there.

2. Minimize the QTSpim window. Then using your favorite text editor (such as NotePad, TextPad, note that TextEdit for Mac is not a good choice since it deals with non-ascii characters, you will need a simple text editor where you can write code), type the following program and save it within the file name assignment1.s (i.e., with a ***.s*** extension).

Anything written right side of '#' will be a comment just like "//" in Java or C++.
It is case-sensitive (upper case and lower case make differences.)

Put your name and other information at the top of the file:

```
################################################################
# Assignment #: 1
#  Name: Your name
#  ASU email: Your ASU email
#  Course: CSE/EEE230, your lecture time such as MWF 1:30pm
#  Description: This is my first assembly language program.
#                    It prints "Hello world".
################################################################

#data declarations: declare variable names used in program, storage allocated in RAM
                    .data
message1:       .asciiz "Hello world.\n" #create a string containing "Hello world.\n"

#program code is contained below under .text
                .text
                .globl    main    #define a global function main

# the program begins execution at main()
main:
        la      $a0, message1  # $a0 = address of message1
        li      $v0, 4              # $v0 = 4  --- this is to call print_string()
        syscall                     # call print_string()
        jr      $ra                  # return
```

Note:

li -- "load immediate"
la -- "load address":

```
la  $t0, var1  # copy RAM address of var1 into register $t0
```

jr -- "jump register":

```
jr  $ra # jump to the address contained in $ra
```

3. Once you have typed in the program and saved it, switch back to the QTSpim window and load the program by clicking *File -> Load File* on the menu bar, and navigating to the location where you saved the file. If your program contains syntax errors, the QTSpim will display the line with the error. If that is the case, you need to go back to the editor window and correct the error and re-save the program. Continue doing this until QTSpim load the program without errors

To execute the program, select *Simulator -> Run/Continue* (or press the *F5* key). You should see "Hello world." displayed in the console window if it is running successfully. You just wrote and executed your first assembly language program!!

4. You can also execute the program by selecting Simulator -> Single Step. This executes one line in the loaded assembly program at a time, and this can be used to debug if it is not producing a correct result.
The program that you have just written has a line using **syscall** The **syscall** instruction stands for system call, i.e., it allows the assembly language program to invoke a call to the QTSpim simulator operating system (in a real computer this would be a call the underlying operating system such as Windows or Linux). Various system calls can be used to request the QTSpim simulator to perform various functions. Here are some of them:

System Call Number	System Call Operation	System Call Description
1	print_int	$v0 = 1; $a0 = int to be printed
4	print_string	$v0 = 4; $a0 = address of beginning of ASCIIZ string
5	read_int	$v0 = 5; user types int at keyboard; value is stored in $v0
8	read_string	$v0 = 8; user types string at keybd; addr of beginning of string is stored in $a0; len in $a1

10	exit	$v0 = 10; terminates the program
11	print_char	$v0 = 11; $a0 = char to be printed
12	read_char	$v0 = 12; user types char at keyboard; value is stored in $v0

We are invoking system call number 4 (print_string). Note that the second instruction of the program, li $v0, 4 puts the value 4 in the $v0 register (i.e., the $r2 register). The second instruction, la, $a0, message1, loads the address of the beginning of our message1 string ("Hello world") into the $a0 register (i.e., the $r4 register). This is what is expected by the print_string system call. (Incidentally, you can think of the system call as a function call with the input parameters being stored in registers; if the function returns values, those values will be store in registers as well.)

5. Modify the program to display the string "Goodbye World.\n" Instead of "Hello world.\n". Save this program as *assignment1b.s*. Be sure to update the program header block. Re-run the program in QTSpim and verify the correct string is displayed.
NOTE: Once you have uploaded a program successfully before, then when you upload a program again, you will need to choose File -> Reinitialize and Load File (instead of Load File) to ignore any main function definition that you have uploaded before. (You can also use Simulator -> Reinitialize Simulator first before loading).

6. If we want to display multiple strings, we can invoke the print_string system call multiple times, once with the address of the beginning of the first string in $a0, and another time with the address of the beginning of another string in $a0 again.
Modify your program to display the following three texts in the console window on separate lines. Update the program header block, save this program as *assignment1c.s*, and run it.

Hello World.
Goodbye World.
This is my first MIPS program.

What to turn in:

-Upload your assignment1c.s file through the assignment submission link in the course website by the assignment deadline. Make sure that your name and email address are correct in the header block.

Go to "GradeScope" tab on Canvas -> CSE/EEE230 -> Assignment1, and upload your program file, assignment1c.s.

Grading Criteria:

____/ 5 Documentation (header with your name, your information/ASU email, and program description and comments within your code)

____/ 1 Indentation and spacing (easy to read)

____/ 6 Required functions and functionalities implemented -- just main for this assignment

____/ 8 Produces correct results?

Total points: 20