contact

- Email：wangw6@sustc.edu.cn
- Office time：

  10：00~12：00 am on Wednesday

  @teaching building 2 #205

Assignments submission rules

- All assignments must **be submitted to Sakai** site, any other forms of submission are not accepted.
- the first three weeks of homework should be **submit after the 4th week**, the late-selected students must also submit homework at the same time.
- If the submission is delayed by one day, there will be a 20% discount on the total score. If delayed for a week, NOT accept any form of supplementary submission. This assignment is 0 points.
- In the case of plagiarism, at the 1st time, the assignment was 0 for all concerned and at the 2nd time ,all concerned one was 0 for this experimental course.
- Reminder:

Assignment scoring and the score publication should be completed within two weeks after the publication of assignment. If you have any objection to the scoring, please email an application for homework scoring review within one week after the publication of homework score.

Experimental Suite
Qtspim /Mars，vivado & minisys

- Learn and practice MIPS (Qtspim / Mars)

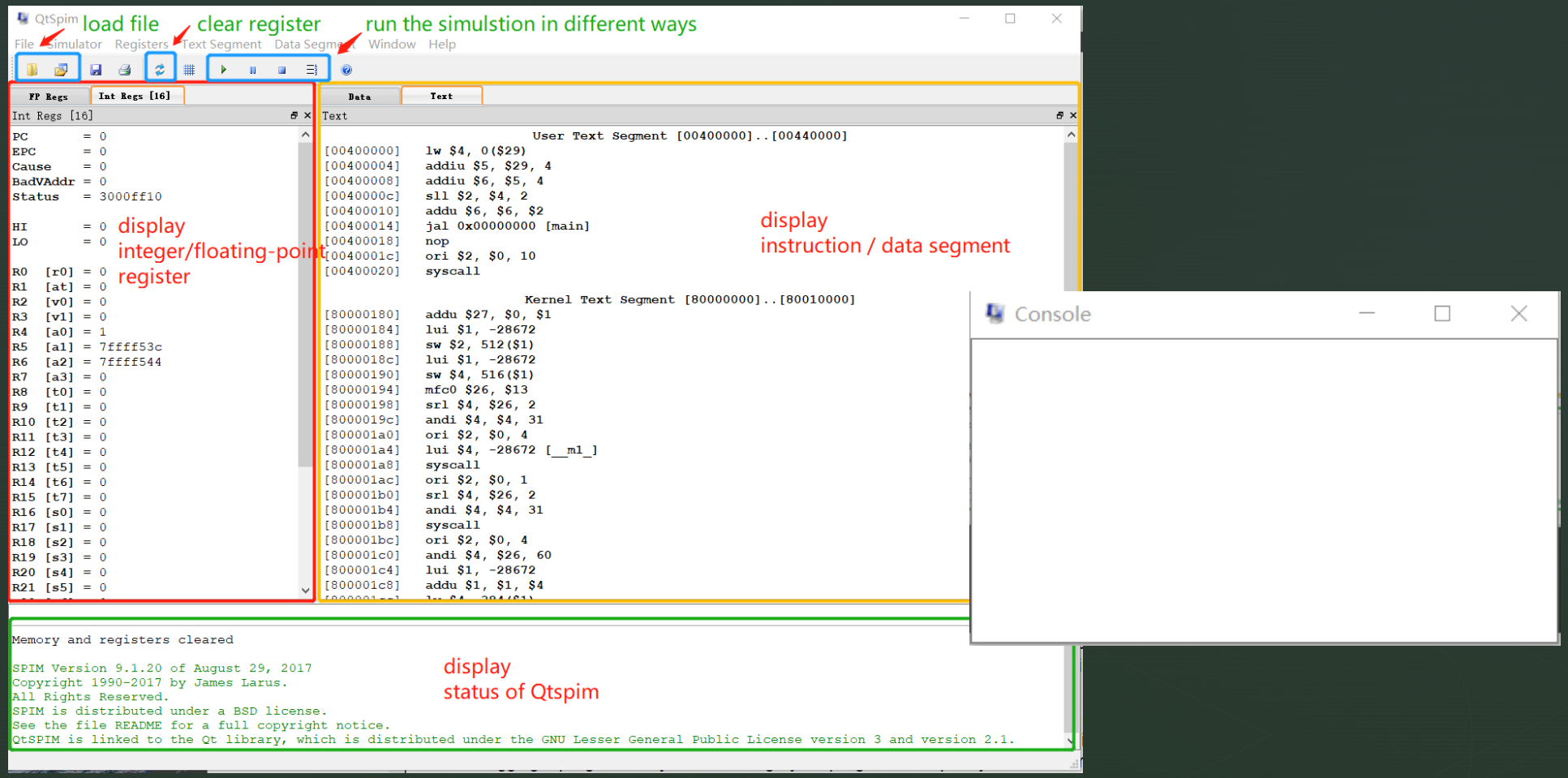
- Design and implement an CPU

QtSpim

- QtSpim
  - Spim , A self-contained simulator that runs MIPS32 program developed by Prof. James Larus from University of Wisconsin-Madison
  - QtSpim is the newest version of *Spim* that currently being actively maintained, runs on Microsoft Windows, Mac OS X, and Linux.
  - *QtSpim* reads and executes programs written in assembly language for a MIPS computer. *QtSpim* does not execute binary (compiled) programs. To simplify programming, *QtSpim* provides a simple debugger and small set of operating system services
  - *QtSpim* implements most of the MIPS32 assembler-extended instruction set. (It omits the floating point comparisons and rounding modes and the memory system page tables.) which means that *QtSpim* will not run programs for all MIPS processors.
  - url : http://spimsimulator.sourceforge.net/

Name	Modified	Size
qtspim_9.1.20_linux64.deb	2017-08-29	19.8 MB
QtSpim_9.1.20_mac.mpkg.zip	2017-08-29	12.4 MB
QtSpim_9.1.20_Windows.msi	2017-08-29	13.8 MB

The windows of QtSpim

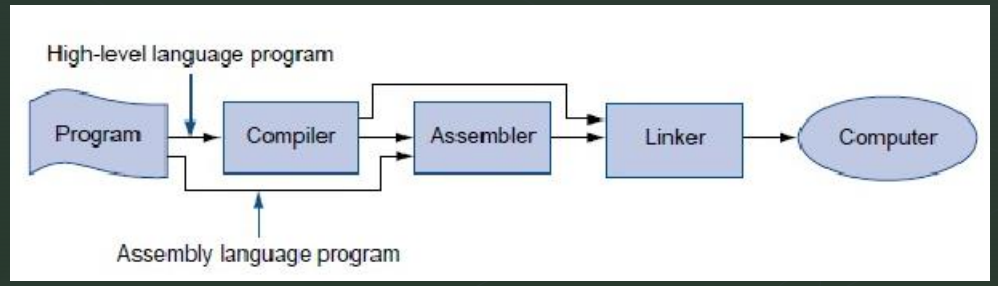


Load file and Do the simulation

- Step1: loading a file (assembly code file usually have the extension “.s”)
  - Open file : **File->Load File** or **File ->Reinitialize and Load File**
  - Tips:
    - Reinitialize and Load File : Clears all the changes made by a grogram, deleting all of its instructions, then reload the last file
    - Before doing the simulation, make sure clear registers ,you can do it by : Simulator->Clear Registers or tools
- Step2:
  - Run file : **Simulator->Run / Continue**
  - The value of register in left window will be refreshed, the output your program writes will appear in the Console window.

Assembly Language Overview

- Assembly language
  - The main body of assembly language is assembly instruction. The difference between assembly instructions and machine instructions lies in the way in which instructions are expressed. Assembly instruction is a writing format of machine instruction for human memory. Assembly language is the mnemonic of machine language.
    - For example, machine instruction 1000100111011000 means that the contents of register BX are sent to AX. The assembly instructions are written as mov ax, bx. Such writing is similar to human language and easy to read and remember.
  - Feature：programs written in assembly language are inherently machine-specific and must be totally rewritten to run on another computer architecture
  - CISC (intel x86) vs RISC (ARM , MIPS)



Assembly Program Structure

- Program Structure
  - data declarations, program code
  - data declaration section followed by program code section
- Data Declarations
  - placed in section of program identified with assembler directive .data
  - declares variable names used in program; storage allocated in main memory (RAM)
- Code
  - placed in section of text identified with assembler directive .text
  - contains program code (instructions)
  - starting point for code e.g .ecution given label main:
  - ending point of main code should use exit system call (see below under System Calls)
- Comments
  - anything following # on a line
    # This stuff would be considred a comment

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

            .data       # variable declarations follow this line
                        # ...

            .text       # instructions follow this line

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

# End of program, leave a blank line afterwards to make SPIM happy
```

http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm

Assembly Program system calls

- **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 code into register $v0 and arguments into registers $a0$a3 (or $f12 for floating-point values).
  - System calls that return values put their results in register $v0 (or $f0 for floating-point results).

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 $v0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $v0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $v0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $v0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

practice

- 1. Install QtSpim on your PC, practise it
  - Using QtCreator open qch file in the ‘help’ directory of QtSpim to get help
- 2. Make a assembly code file , load and run it by QtSpim, what happens ?
- 3. To find that is there an object file or executable file generated while doing the simulation,why?
- 5. Another ***MIPS Assembler and Runtime Simulator(Mars) ,****practise it*
  - http://courses.missouristate.edu/kenvollmar/mars/

A demo

```
#    text segment
     .text
     .globl main
main:                   # execution starts here
     la $a0,str         # put string address into a0
     li $v0,4           # system call to print a string
    syscall
     li $v0,10
    syscall             # system call to exit
#data segment
    .data

str:    .asciiz "hello Internet\n"
```