CENG3420

Lab 1-1: MIPS assembly language programing

Haoyu Yang

Department of Computer Science and Engineering
The Chinese University of Hong Kong

hyyang@cse.cuhk.edu.hk

Spring 2020

香 港 中 文 大 學
The Chinese University of Hong Kong

Overview

SPIM

Assembly Programing

System Service in SPIM

Lab Assignment

Overview

SPIM

Assembly Programing

System Service in SPIM

Lab Assignment

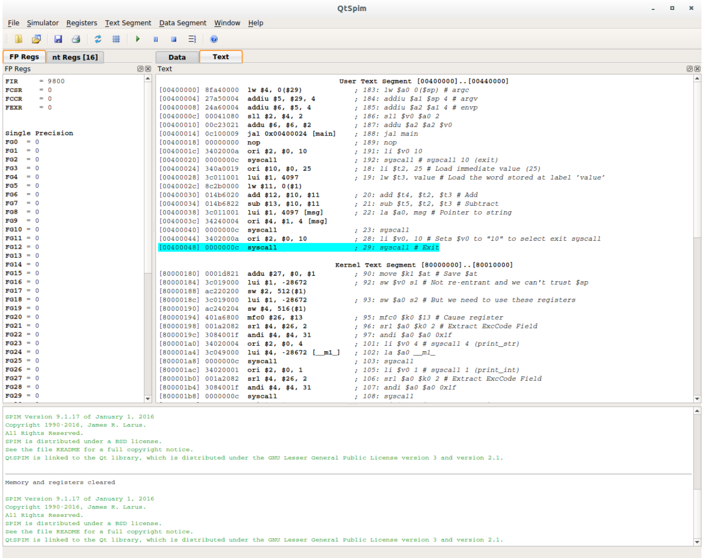
What is SPIM

- **SPIM is a MIPS32 simulator.**
- *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.

Dowload it here:

`http://sourceforge.net/projects/spimsimulator/files/`

SPIM Overview



FP Regs

```
FIR      = 9800
FCSR     = 0
FCCR     = 0
FEXR     = 0


Single Precision
FG0  = 0
FG1  = 0
FG2  = 0
FG3  = 0
FG4  = 0
FG5  = 0
FG6  = 0
FG7  = 0
FG8  = 0
FG9  = 0
FG10 = 0
FG11 = 0
FG12 = 0
FG13 = 0
FG14 = 0
FG15 = 0
FG16 = 0
FG17 = 0
FG18 = 0
FG19 = 0
FG20 = 0
FG21 = 0
FG22 = 0
FG23 = 0
FG24 = 0
FG25 = 0
FG26 = 0
FG27 = 0
FG28 = 0
FG29 = 0
```

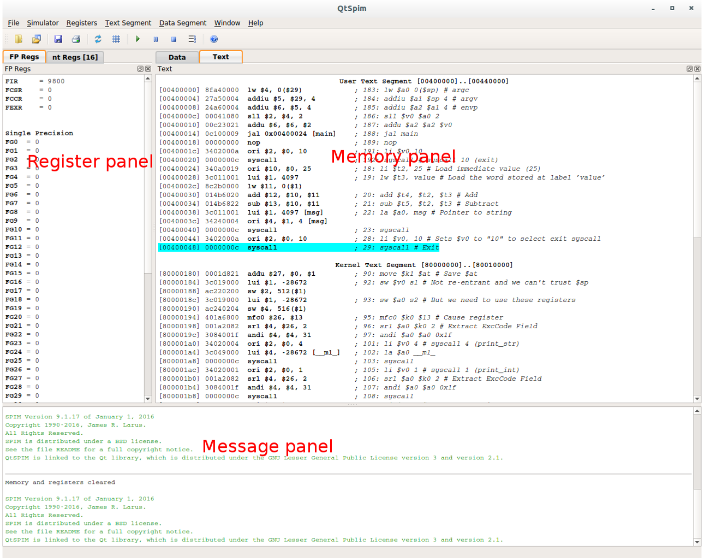
Text

```
                                      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] 340a0019  ori $10, $0, 25          ; 18: li $t2, 25 # Load immediate value (25)
[00400028] 3c011001  lui $1, 4097             ; 19: lw $t3, value # Load the word stored at label 'value'
[0040002c] 8c2b0000  lw $11, 0($1)
[00400030] 014b6020  add $12, $10, $11        ; 20: add $t4, $t2, $t3 # Add
[00400034] 014b6822  sub $13, $10, $11        ; 21: sub $t5, $t2, $t3 # Subtract
[00400038] 3c011001  lui $1, 4097 [msg]       ; 22: la $a0, msg # Pointer to string
[0040003c] 34240004  ori $4, $1, 4 [msg]
[00400040] 0000000c  syscall                  ; 23: syscall
[00400044] 3402000a  ori $2, $0, 10           ; 28: li $v0, 10 # Sets $v0 to "10" to select exit syscall
[00400048] 0000000c  syscall                  ; 29: syscall # Exit

                                     Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1         ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672           ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672           ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13            ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2           ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31          ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4            ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]   ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                  ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1            ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2           ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31          ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                  ; 108: syscall
```

```
SPIM Version 9.1.17 of January 1, 2016
Copyright 1990-2016, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.17 of January 1, 2016
Copyright 1990-2016, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

What SPIM looks like.

Register Panel and Memory Panel



Register panel

```
FIR     = 9800
FCSR    = 0
FCCR    = 0
FEXR    = 0

Single Precision
FG0  = 0
FG1  = 0
FG2  = 0
FG3  = 0
FG4  = 0
FG5  = 0
FG6  = 0
FG7  = 0
FG8  = 0
FG9  = 0
FG10 = 0
FG11 = 0
FG12 = 0
FG13 = 0
FG14 = 0
FG15 = 0
FG16 = 0
FG17 = 0
FG18 = 0
FG19 = 0
FG20 = 0
FG21 = 0
FG22 = 0
FG23 = 0
FG24 = 0
FG25 = 0
FG26 = 0
FG27 = 0
FG28 = 0
FG29 = 0
```

Memory panel

```
                          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] 340a0019  ori $10, $0, 25        ; 18: li $t2, 25 # Load immediate value (25)
[00400028] 3c011001  lui $1, 4097           ; 19: lw $t3, value # Load the word stored at label ’value’
[0040002c] 8c2b0000  lw $11, 0($1)
[00400030] 014b6020  add $12, $10, $11      ; 20: add $t4, $t2, $t3 # Add
[00400034] 014b6822  sub $13, $10, $11      ; 21: sub $t5, $t2, $t3 # Subtract
[00400038] 3c011001  lui $1, 4097 [msg]     ; 22: la $a0, msg # Pointer to string
[0040003c] 34240004  ori $4, $1, 4 [msg]
[00400040] 0000000c  syscall                ; 23: syscall
[00400044] 3402000a  ori $2, $0, 10         ; 28: li $v0, 10 # Sets $v0 to "10" to select exit syscall
[00400048] 0000000c  syscall                ; 29: syscall # Exit

                          Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1       ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672         ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672         ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13          ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2         ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31        ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4          ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1          ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2         ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31        ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                ; 108: syscall
```

Message panel

```
SPIM Version 9.1.17 of January 1, 2016
Copyright 1990-2016, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.17 of January 1, 2016
Copyright 1990-2016, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

There’s also a console window.

Operations

- Load a source file: File → Reinitialize and Load File
- Run the code: F5 or Press the green triangle button
- Single stepping: F10
- Breakpoint: in Text panel, right click on an address to set a breakpoint there.

Overview

SPIM

Assembly Programing

System Service in SPIM

Lab Assignment

Registers

- 32 general-purpose registers
- register preceded by $ in assembly language instruction
- two formats for addressing:
  - using register number e.g. `$0` through `$31`
  - using equivalent names e.g. `$t1`, `$sp`
- special registers Lo and Hi used to store result of multiplication and division
  - not directly addressable; contents accessed with special instruction `mfhi` ("move from Hi") and `mflo` ("move from Lo")

Register Names and Descriptions

Name	Register Number	Usage	Preserve on call?
$zero	0	constant 0 (hardware)	n.a.
$at	1	reserved for assembler	n.a.
$v0 - $v1	2-3	returned values	no
$a0 - $a3	4-7	arguments	yes
$t0 - $t7	8-15	temporaries	no
$s0 - $s7	16-23	saved values	yes
$t8 - $t9	24-25	temporaries	no
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return addr (hardware)	yes

Data Types and Literals

Data types:

- Instructions are all 32 bits
- byte(8 bits), halfword (2 bytes), word (4 bytes)
- a character requires 1 byte of storage
- an integer requires 1 word (4 bytes) of storage
- Data types: `.asciiz` for string, `.word` for int, ...

Literals:

- numbers entered as is. e.g. 4
- characters enclosed in single quotes. e.g. ‘b’
- strings enclosed in double quotes. e.g. “A string”

Program Structure I

- Just plain text file with data declarations, program code (name of file should end in suffix .s to be used with SPIM simulator)
- Data declaration section followed by program code section

Data Declarations

- Identified with assembler directive **.data**.
- Declares variable names used in program
- Storage allocated in main memory (RAM)
- `<name>:   .<datatype> <value>`

Program Structure II

Code

- placed in section of text identified with assembler directive **.text**
- contains program code (instructions)
- starting point for code e.g. execution given label **main:**
- ending point of main code should use exit system call

Comments

anything following # on a line

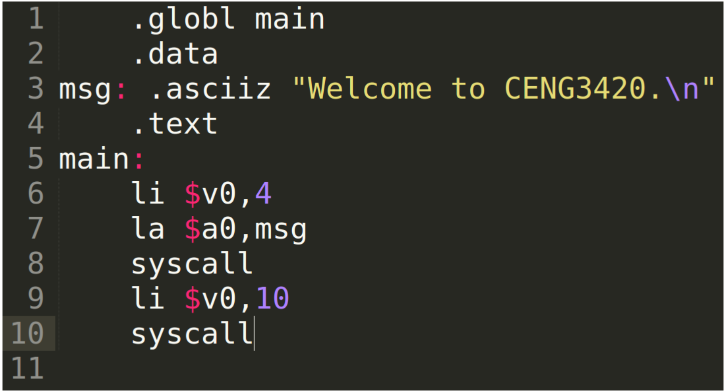
Program Structure III

The structure of an assembly program looks like this:

Program outline

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

    .globl main

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

main:       # indicates start of code
            # ...

# End of program, leave a blank line afterwards
```

An Example Program



```
    .globl main
    .data
msg: .asciiz "Welcome to CENG3420.\n"
    .text
main:
    li $v0,4
    la $a0,msg
    syscall
    li $v0,10
    syscall
```

- `li`: load immediate
- `la`: load address
- `lw`: load word from memory

More Information

For more information about MIPS instructions and assembly programing you can refer to:

1. Lecture slides and textbook.
2. `http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html`

Overview

SPIM

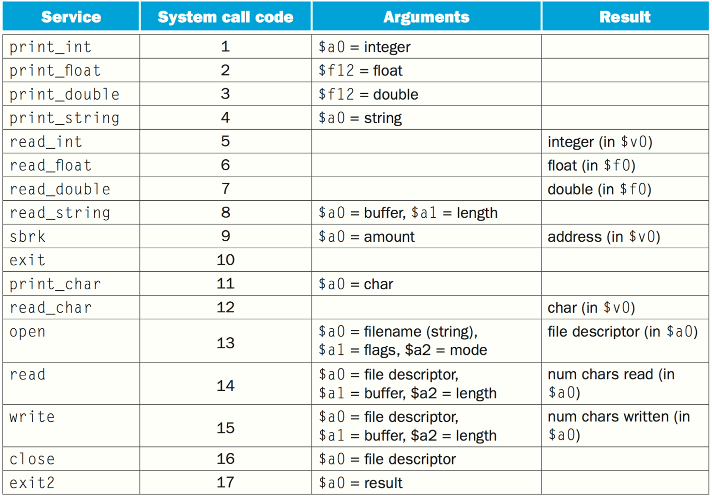
Assembly Programing

System Service in SPIM

Lab Assignment

System calls in SPIM I

SPIM provides a small set of operating system-like services through the system call (`syscall`) instruction.



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 $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	

System calls in SPIM II

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). Like this example:

Using system call

```
     .data
str: .asciiz "the␣answer␣=␣" #labels always followed by colon
     .text

      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
```

Overview

SPIM

Assembly Programing

System Service in SPIM

Lab Assignment

Lab Assignment

Write an assembly program with the following requirements:

1. Define two variables `var1` and `var2` which have initial value 15 and 19, respectively.
2. Print RAM addresses of `var1` and `var2` using syscall.
3. Increase `var1` by 1 and multiply `var2` by 4.
4. Print `var1` and `var2`.
5. Swap `var1` and `var2` and print them.

Submission Method:

Submit the source code and report after the whole Lab1, onto blackboard.

Some Tips

1. Variables should be declared following the `.data` identifier.
2. `<name>:   .<datatype> <value>`
3. Use `la` instruction to access the RAM address of declared data.
4. Use system call to print integers.
5. Do not forget exit system call.