

## Lab Assignment - 03

Q1. Write a assembly program for Hello world and print it into console.

Ans :

```
*hello_world.asm - Notepad
File Edit Format View Help

.text
main:
    li $v0,4
    la $a0, greeting
    syscall

    li $v0, 10
    syscall
.data
greeting: .asciiz "Hello World"
```

# Define the program instructions.  
# Label to define the main program.  
# Load 4 into \$v0 to indicate a print string.  
# Load the address of the greeting into \$a0.  
# Print greeting. The print is indicated by  
# \$v0 having a value of 4, and the string to  
# print is stored at the address in \$a0.  
# Load a 10 (halt) into \$v0.  
# The program ends.  
# Define the program data.  
#The string to print.

The screenshot displays the QtSpim MIPS simulator interface. On the left, the 'Int Regs [16]' panel shows the current state of registers: PC is 400038, EPC is 0, Cause is 0, BadVAddr is 0, Status is 3000fff10, HI is 0, LO is 0, R0 [r0] is 0, R1 [at] is 10010000, R2 [v0] is a, R3 [v1] is 0, R4 [a0] is 10010000, R5 [a1] is 7ffff3d4, R6 [a2] is 7ffff3e0, and R7 [a3] through R21 [s5] are all 0. The main window shows the 'Text' segment of the assembly code, which includes instructions like 'lw \$4, 0(\$29)', 'addiu \$5, \$29, 4', 'addiu \$6, \$5, 4', 'sll \$2, \$4, 2', 'addu \$6, \$6, \$2', 'jal 0x00400024 [main]', 'nop', 'ori \$2, \$0, 10', 'syscall', 'ori \$2, \$0, 4', 'lui \$1, 4097 [greeting]', 'ori \$4, \$1, 0 [greeting]', 'syscall', 'ori \$2, \$0, 10', and 'syscall'. A 'Console' window in the foreground shows the output 'Hello World'.

Q2. Write a program to read an integer number from user and print to the console.

ANS :

```
*input_int.asm - Notepad
File Edit Format View Help
# Program to read an integer number from a user, and
# print that number back to the console.

.text
main:
# Prompt for the integer to enter
li $v0, 4
la $a0, prompt
syscall

li $v0, 5                                # Read the integer and save it in $s0
syscall
move $s0, $v0
li $v0, 4                                # Output the text
la $a0, output
syscall

li $v0, 1                                # Output the number
move $a0, $s0
syscall

li $v0, 10                               # Exit the program
syscall

.data
prompt: .asciiz "Please enter an integer: "
output: .asciiz "\nYou typed the number "
```

The screenshot displays the QtSPIM simulator interface. On the left, the 'Int Regs [16]' panel shows register values, with R16 (\$s0) highlighted in red and set to 17. The 'Text' panel on the right shows the assembly code being executed, with the final instruction 'syscall' at address 00400060 highlighted in blue. A 'Console' window on the right shows the program's output: 'Please enter an integer: 23' and 'You typed the number 23'. The bottom status bar indicates the simulator is running on a 32-bit MIPS architecture.

### Q3. Program to prompt and read a string from a user.

ANS :

```
*input_str.asm - Notepad
File Edit Format View Help

# Program to read a string from a user, and # print that string back to the console.
.text
main:
    # Prompt for the string to enter
    li $v0, 4
    la $a0, prompt
    syscall

    # Read the string.
    li $v0, 8
    la $a0, input
    lw $a1, inputSize
    syscall

    # Output the text
    li $v0, 4
    la $a0, output
    syscall

    # Output the number
    li $v0, 4
    la $a0, input
    syscall

    # Exit the program
    li $v0, 10
    syscall

.data
input:      .space 81
inputSize:  .word 80
prompt:     .asciiz "Please enter an string: "
output:     .asciiz "\nYou typed the string: "
```

Ln 1, Col 1

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

Reg	Value
PC	400070
EPC	0
Cause	0
BadVAddr	0
Status	3000ff10
HI	0
LO	0
R0 [r0]	0
R1 [at]	10010000
R2 [v0]	a
R3 [v1]	0
R4 [a0]	10010000
R5 [a1]	50
R6 [a2]	7ffff598
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 [a0]	0
R17 [a1]	0
R18 [a2]	0
R19 [a3]	0
R20 [a4]	0
R21 [a5]	0
R22 [a6]	0
R23 [a7]	0
R24 [t8]	0
R25 [t9]	0

Text

User Text Segment [00400000]..[00400000]

Address	Instruction	Comment
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
00400024	34020004 ori \$2, \$0, 4	; 7: li \$v0, 4
00400028	3c011001 lui \$1, 4097 [prompt]	; 8: la \$a0, prompt
0040002c	34240058 ori \$4, \$1, 88 [prompt]	
00400030	0000000c syscall	; 9: syscall
00400034	34020008 ori \$2, \$0, 8	; 12: li \$v0, 8
00400038	3c011001 lui \$1, 4097 [input]	; 13: la \$a0, input
0040003c	34240000 ori \$4, \$1, 0 [input]	
00400040	3c011001 lui \$1, 4097 [inputSize]	; 14: lw \$a1, inputSize
00400044	8c250054 lw \$5, 84(\$1) [inputSize]	
00400048	0000000c syscall	; 15: syscall
0040004c	34020004 ori \$2, \$0, 4	; 18: li \$v0, 4
00400050	3c011001 lui \$1, 4097 [output]	; 19: la \$a0, output
00400054	34240071 ori \$4, \$1, 113 [output]	
00400058	0000000c syscall	; 20: syscall
0040005c	34020004 ori \$2, \$0, 4	; 23: li \$v0, 4
00400060	3c011001 lui \$1, 4097 [input]	; 24: la \$a0, input
00400064	34240000 ori \$4, \$1, 0 [input]	
00400068	0000000c syscall	; 25: syscall
0040006c	3402000a ori \$2, \$0, 10	; 28: li \$v0, 10
00400070	0000000c syscall	; 29: syscall

Kernel Text Segment [80000000]..[80010000]

Address	Instruction	Comment
80000180	0001d821 addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at

spim: (parser) syntax error on line 14 of file D:/college/coa/assignment lab 3/input\_str.asm

Console

Please enter an string: Mohan Manjhi

You typed the string: Mohan Manjhi

Q4. Write a program to print out a random number from 1..100.

```
*Random_number.asm - Notepad
File Edit Format View Help

.data
random_min: .word 1      # Minimum value for the random number (1)
random_max: .word 100    # Maximum value for the random number (100)
format_string: .asciiz "Random number between 1 to 100 : "

.text
.globl main

main:
    # Initialize the random number generator
    li $v0, 43            # syscall code for srandom
    li $a0, 42            # Seed for random number generation (you can use any value)
    syscall

    li $v0, 43            # syscall code for random
    lw $a0, random_min    # Minimum value
    lw $a1, random_max    # Maximum value
    syscall
    move $t0, $v0         # Store the random number in $t0

    # Ensure the random number is at least 1
    bge $t0, $zero, random_done
    addi $t0, $t0, 1

random_done:

    # Print a newline character
    li $v0, 4            # syscall code for print string
    la $a0, format_string
    syscall

    # Print the random number
    li $v0, 1            # syscall code for print integer
    move $a0, $t0
    syscall

    # Exit the program
    li $v0, 10           # syscall code for exit
    syscall
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 400078  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000ff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 10010000  
R2 [v0] = a  
R3 [v1] = 0  
R4 [a0] = 2b  
R5 [a1] = 6  
R6 [a2] = 7ffff598  
R7 [a3] = 0  
R8 [t0] = 2b  
R9 [t1] = 0  
R10 [t2] = 0  
R11 [t3] = 0  
R12 [t4] = 0  
R13 [t5] = 0  
R14 [t6] = 0  
R15 [t7] = 0  
R16 [a0] = 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

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 $v0
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $v0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 1
[00400020] 0000000c syscall ; 192: syscall
[00400024] 3402002b ori $2, $0, 43 ; 193: li $v0, 43
[00400028] 3404002a ori $4, $0, 42 ; 194: li $v0, 42
[0040002c] 0000000c syscall ; 195: syscall
[00400030] 3402002b ori $2, $0, 43 ; 198: li $v0, 43
[00400034] 3c011001 lui $1, 4097 ; 199: lw $a0, r
[00400038] 8c240000 lw $4, 0($1) ; 20: lw $a1, r
[0040003c] 3c011001 lui $1, 4097 ; 21: syscall
[00400040] 8c250004 lw $5, 4($1) ; 22: move $t0,
[00400044] 0000000c syscall ; 25: bge $t0,
[00400048] 00024021 addu $8, $0, $2 ; 26: addi $t0,
[0040004c] 0100082a slt $1, $8, $0 ; 32: li $v0, 4
[00400050] 10200002 beq $1, $0, 8 [random_done-0x00400050] ; 33: la $a0
[00400054] 21080001 addi $8, $8, 1 ; 34: syscall
[00400058] 3402000a ori $2, $0, 4 ; 37: li $v0, 1 # syscall code for print integer
[0040005c] 3c011001 lui $1, 4097 [format_string] ; 38: move $a0, $t0
[00400060] 34240008 ori $4, $1, 8 [format_string] ; 39: syscall
[00400064] 0000000c syscall ; 42: li $v0, 10 # syscall code for exit
[00400068] 34020001 ori $2, $0, 1 ; 43: syscall
[0040006c] 00082021 addu $4, $0, $8 ; 43: syscall
```

Console

Random number between 1 to 100 : 43

spim: (parser) syntax error on line 14 of file D:/college/coa/assignment lab 3/input\_str.asm



Q5. Write a MIPS assembly program called Problem2.asm. This program should have two global variables, which stores numeric values. The program itself should sum those two values and then print the result to the console.

```

Problem2.asm - Notepad
File Edit Format View Help

.data
    # Global variables to store numeric values
    num1: .word 50
    num2: .word 50
    result_message: .asciiz "The sum is: "

.text
.globl main

main:
    # Load values from global variables into registers
    lw $t0, num1
    lw $t1, num2

    # Add the values
    add $t2, $t0, $t1

    # Print the result message
    li $v0, 4
    la $a0, result_message
    syscall

    # Print the result (the sum)
    li $v0, 1
    move $a0, $t2
    syscall

    # Exit the program
    li $v0, 10
    syscall

```

The screenshot shows the QtSPIM MIPS simulator interface. The main window is titled 'QtSPIM' and contains a menu bar (File, Simulator, Registers, Text Segment, Data Segment, Window, Help) and a toolbar. Below the toolbar, there are tabs for 'FP Regs', 'Int Regs [16]', 'Data', and 'Text'. The 'Int Regs [16]' tab is selected, showing the register file with the following values:

Register	Value
PC	400058
EPC	0
Cause	0
BadVAddr	0
Status	3000ff10
HI	0
LO	0
R0 [r0]	0
R1 [at]	10010000
R2 [v0]	a
R3 [v1]	0
R4 [a0]	64
R5 [a1]	7ffff3d4
R6 [a2]	7ffff3e0
R7 [a3]	0
R8 [t0]	2a
R9 [t1]	3a
R10 [t2]	64
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

The 'Text' tab is also selected, showing the assembly code for the 'User Text Segment [00400000] .. [00440000]'. The code is as follows:

```

[00400000] lw $4, 0($29)
[00400004] addiu $5, $29, 4
[00400008] addiu $6, $5, 4
[0040000c] sll $2, $4, 2
[00400010] addu $6, $6, $2
[00400014] jal 0x00400024 [main]
[00400018] nop
[0040001c] ori $2, $0, 10
[00400020] syscall
[00400024] lui $1, 4097
[00400028] lw $8, 0($1)
[0040002c] lui $1, 4097
[00400030] lw $9, 4($1)
[00400034] add $10, $8, $9
[00400038] ori $2, $0, 4
[0040003c] lui $1, 4097 [result_msg]
[00400040] ori $4, $1, 8 [result_msg]
[00400044] syscall
[00400048] ori $2, $0, 1
[0040004c] addu $4, $0, $10
[00400050] syscall
[00400054] ori $2, $0, 10
[00400058] syscall

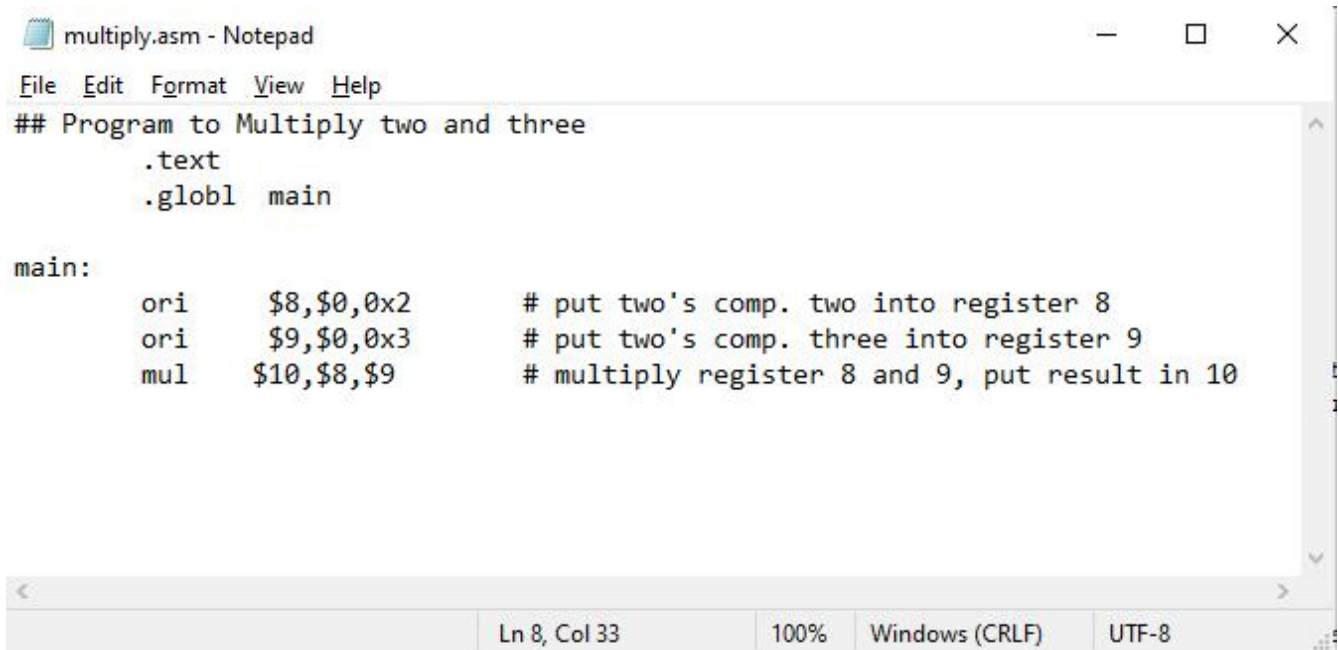
```

A console window is open in the foreground, displaying the output: 'The sum is: 100'.

At the bottom of the QtSPIM window, there is a copyright notice: 'Copyright 1990-2021 by James 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.'

Q6. The program should have two temporary registers , which stores numeric values. The program must have multiply these two numbers and the result is store in the saved value registers.

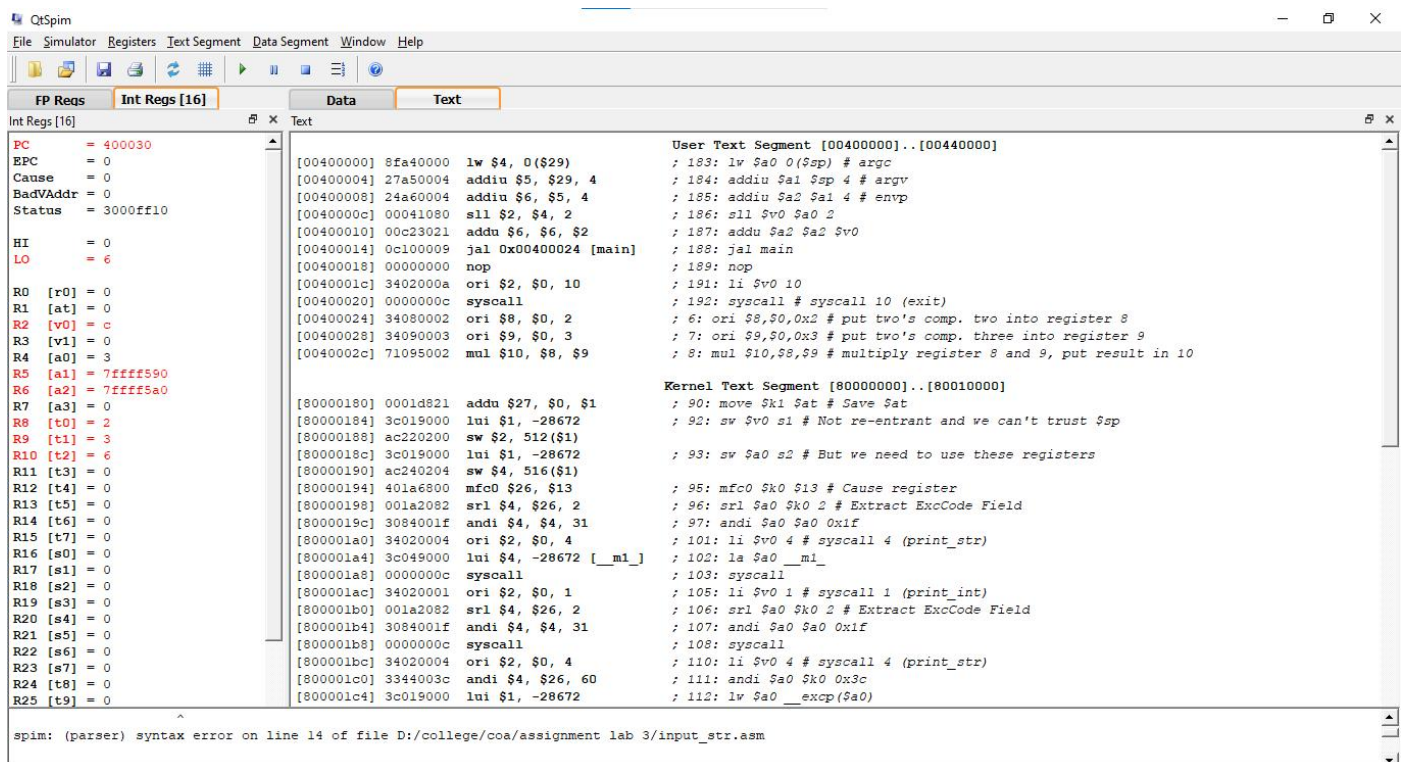
ANS :



```
File Edit Format View Help
## Program to Multiply two and three
.text
.globl main

main:
    ori    $8,$0,0x2    # put two's comp. two into register 8
    ori    $9,$0,0x3    # put two's comp. three into register 9
    mul    $10,$8,$9     # multiply register 8 and 9, put result in 10
```

Ln 8, Col 33    100%    Windows (CRLF)    UTF-8



QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Reqs Int Regs [16] Data Text

Int Regs [16]

PC	= 400030
EPC	= 0
Cause	= 0
BadVAddr	= 0
Status	= 3000ff10
HI	= 0
LO	= 6
R0 [r0]	= 0
R1 [at]	= 0
R2 [v0]	= c
R3 [v1]	= 0
R4 [a0]	= 3
R5 [a1]	= 7ffff590
R6 [a2]	= 7ffff5a0
R7 [a3]	= 0
R8 [t0]	= 2
R9 [t1]	= 3
R10 [t2]	= 6
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

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] 34080002 ori $8, $0, 2 ; 6: ori $8,$0,0x2 # put two's comp. two into register 8
[00400028] 34090003 ori $9, $0, 3 ; 7: ori $9,$0,0x3 # put two's comp. three into register 9
[0040002c] 71095002 mul $10, $8, $9 ; 8: mul $10,$8,$9 # multiply register 8 and 9, put result in 10
```

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 $1 # Not re-entrant and we can't trust $sp
[80000188] ac220200 sw $2, 512($1) ; 93: sw $a0 $2 # But we need to use these registers
[8000018c] 3c019000 lui $1, -28672 ; 95: mfc0 $k0 $13 # Cause register
[80000190] ac240204 sw $4, 516($1) ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[80000194] 401a6800 mfc0 $26, $13 ; 97: andi $a0 $a0 0x1f
[80000198] 001a2082 srl $4, $26, 2 ; 101: li $v0 4 # syscall 4 (print_str)
[8000019c] 3084001f andi $4, $4, 31 ; 102: la $a0 __ml_
[800001a0] 34020004 ori $2, $0, 4 ; 103: syscall
[800001a4] 3c049000 lui $4, -28672 [__ml_] ; 105: li $v0 1 # syscall 1 (print_int)
[800001a8] 0000000c syscall ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001ac] 34020001 ori $2, $0, 1 ; 107: andi $a0 $a0 0x1f
[800001b0] 001a2082 srl $4, $26, 2 ; 108: syscall
[800001b4] 3084001f andi $4, $4, 31 ; 110: li $v0 4 # syscall 4 (print_str)
[800001b8] 0000000c syscall ; 111: andi $a0 $k0 0x3c
[800001bc] 34020004 ori $2, $0, 4 ; 112: lw $a0 __excp($a0)
[800001c0] 3344003c andi $4, $26, 60
[800001c4] 3c019000 lui $1, -28672
```

spim: (parser) syntax error on line 14 of file D:/college/coa/assignment lab 3/input\_str.asm

Q7. Write a program to find out the square of an number and print the result to the console.

ANS :

```
square.asm - Notepad
File Edit Format View Help
.data
    prompt: .asciiz "Enter a number: "
    result_message: .asciiz "The square is: "

.text
.globl main

main:
    # Print a prompt to enter a number
    li $v0, 4
    la $a0, prompt
    syscall

    # Read an integer from the user
    li $v0, 5
    syscall
    move $t0, $v0 # Store the input in $t0

    # Calculate the square of the number
    mul $t1, $t0, $t0 # $t1 = $t0 * $t0

    # Print the result message
    li $v0, 4
    la $a0, result_message
    syscall

    # Print the square (the result)
    li $v0, 1
    move $a0, $t1
    syscall

    # Exit the program
    li $v0, 10
    syscall
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

Register	Value
PC	400060
EPC	0
Cause	0
BadVAddr	0
Status	3000ff10
HI	0
LO	e1
R0 [r0]	0
R1 [at]	10010000
R2 [v0]	a
R3 [v1]	0
R4 [a0]	e1
R5 [a1]	7ffff590
R6 [a2]	7ffff5a0
R7 [a3]	0
R8 [t0]	f
R9 [t1]	e1
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

Text

```
[00400000] 8fa40000 lw $4, 0($29)
[00400004] 27a50004 addiu $5, $29, 4
[00400008] 24a60004 addiu $6, $5, 4
[0040000c] 00041080 sll $2, $4, 2
[00400010] 00c23021 addu $6, $6, $2
[00400014] 0c100009 jal 0x00400024 [main]
[00400018] 00000000 nop
[0040001c] 3402000a ori $2, $0, 10
[00400020] 0000000c syscall
[00400024] 34020004 ori $2, $0, 4
[00400028] 3c041001 lui $4, 4097 [prompt]
[0040002c] 0000000c syscall
[00400030] 34020005 ori $2, $0, 5
[00400034] 0000000c syscall
[00400038] 00024021 addu $8, $0, $2
[0040003c] 71084802 mul $9, $8, $8
[00400040] 34020004 ori $2, $0, 4
[00400044] 3c011001 lui $1, 4097 [result_message]
[00400048] 34240011 ori $4, $1, 17 [result_message]
[0040004c] 0000000c syscall
[00400050] 34020001 ori $2, $0, 1
[00400054] 00092021 addu $4, $0, $9
[00400058] 0000000c syscall
[0040005c] 3402000a ori $2, $0, 10
[00400060] 0000000c syscall
```

User Text Segment [00400000]..[00440000]

```
: 183: lw $a0 0($sp) # argc
: 184: addiu $a1 $sp 4 # argv
: 185: addiu $a2 $a1 4 # envp
: 186: sll $v0 $a0 2
: 187: addu $a2 $a2 $v0
: 188:
: 189:
: 190:
: 191:
: 192:
: 193:
: 194:
: 195:
: 196:
: 197:
: 198:
: 199:
: 200:
: 201:
: 202:
: 203:
: 204:
: 205:
: 206:
: 207:
: 208:
: 209:
: 210:
: 211:
: 212:
: 213:
: 214:
: 215:
: 216:
: 217:
: 218:
: 219:
: 220:
: 221:
: 222:
: 223:
: 224:
: 225:
: 226:
: 227:
: 228:
: 229:
: 230:
: 231:
: 232:
: 233:
: 234:
: 235:
: 236:
: 237:
: 238:
: 239:
: 240:
: 241:
: 242:
: 243:
: 244:
: 245:
: 246:
: 247:
: 248:
: 249:
: 250:
: 251:
: 252:
: 253:
: 254:
: 255:
: 256:
: 257:
: 258:
: 259:
: 260:
: 261:
: 262:
: 263:
: 264:
: 265:
: 266:
: 267:
: 268:
: 269:
: 270:
: 271:
: 272:
: 273:
: 274:
: 275:
: 276:
: 277:
: 278:
: 279:
: 280:
: 281:
: 282:
: 283:
: 284:
: 285:
: 286:
: 287:
: 288:
: 289:
: 290:
: 291:
: 292:
: 293:
: 294:
: 295:
: 296:
: 297:
: 298:
: 299:
: 300:
: 301:
: 302:
: 303:
: 304:
: 305:
: 306:
: 307:
: 308:
: 309:
: 310:
: 311:
: 312:
: 313:
: 314:
: 315:
: 316:
: 317:
: 318:
: 319:
: 320:
: 321:
: 322:
: 323:
: 324:
: 325:
: 326:
: 327:
: 328:
: 329:
: 330:
: 331:
: 332:
: 333:
: 334:
: 335:
: 336:
: 337:
: 338:
: 339:
: 340:
: 341:
: 342:
: 343:
: 344:
: 345:
: 346:
: 347:
: 348:
: 349:
: 350:
: 351:
: 352:
: 353:
: 354:
: 355:
: 356:
: 357:
: 358:
: 359:
: 360:
: 361:
: 362:
: 363:
: 364:
: 365:
: 366:
: 367:
: 368:
: 369:
: 370:
: 371:
: 372:
: 373:
: 374:
: 375:
: 376:
: 377:
: 378:
: 379:
: 380:
: 381:
: 382:
: 383:
: 384:
: 385:
: 386:
: 387:
: 388:
: 389:
: 390:
: 391:
: 392:
: 393:
: 394:
: 395:
: 396:
: 397:
: 398:
: 399:
: 400:
: 401:
: 402:
: 403:
: 404:
: 405:
: 406:
: 407:
: 408:
: 409:
: 410:
: 411:
: 412:
: 413:
: 414:
: 415:
: 416:
: 417:
: 418:
: 419:
: 420:
: 421:
: 422:
: 423:
: 424:
: 425:
: 426:
: 427:
: 428:
: 429:
: 430:
: 431:
: 432:
: 433:
: 434:
: 435:
: 436:
: 437:
: 438:
: 439:
: 440:
: 441:
: 442:
: 443:
: 444:
: 445:
: 446:
: 447:
: 448:
: 449:
: 450:
: 451:
: 452:
: 453:
: 454:
: 455:
: 456:
: 457:
: 458:
: 459:
: 460:
: 461:
: 462:
: 463:
: 464:
: 465:
: 466:
: 467:
: 468:
: 469:
: 470:
: 471:
: 472:
: 473:
: 474:
: 475:
: 476:
: 477:
: 478:
: 479:
: 480:
: 481:
: 482:
: 483:
: 484:
: 485:
: 486:
: 487:
: 488:
: 489:
: 490:
: 491:
: 492:
: 493:
: 494:
: 495:
: 496:
: 497:
: 498:
: 499:
: 500:
: 501:
: 502:
: 503:
: 504:
: 505:
: 506:
: 507:
: 508:
: 509:
: 510:
: 511:
: 512:
: 513:
: 514:
: 515:
: 516:
: 517:
: 518:
: 519:
: 520:
: 521:
: 522:
: 523:
: 524:
: 525:
: 526:
: 527:
: 528:
: 529:
: 530:
: 531:
: 532:
: 533:
: 534:
: 535:
: 536:
: 537:
: 538:
: 539:
: 540:
: 541:
: 542:
: 543:
: 544:
: 545:
: 546:
: 547:
: 548:
: 549:
: 550:
: 551:
: 552:
: 553:
: 554:
: 555:
: 556:
: 557:
: 558:
: 559:
: 560:
: 561:
: 562:
: 563:
: 564:
: 565:
: 566:
: 567:
: 568:
: 569:
: 570:
: 571:
: 572:
: 573:
: 574:
: 575:
: 576:
: 577:
: 578:
: 579:
: 580:
: 581:
: 582:
: 583:
: 584:
: 585:
: 586:
: 587:
: 588:
: 589:
: 590:
: 591:
: 592:
: 593:
: 594:
: 595:
: 596:
: 597:
: 598:
: 599:
: 600:
: 601:
: 602:
: 603:
: 604:
: 605:
: 606:
: 607:
: 608:
: 609:
: 610:
: 611:
: 612:
: 613:
: 614:
: 615:
: 616:
: 617:
: 618:
: 619:
: 620:
: 621:
: 622:
: 623:
: 624:
: 625:
: 626:
: 627:
: 628:
: 629:
: 630:
: 631:
: 632:
: 633:
: 634:
: 635:
: 636:
: 637:
: 638:
: 639:
: 640:
: 641:
: 642:
: 643:
: 644:
: 645:
: 646:
: 647:
: 648:
: 649:
: 650:
: 651:
: 652:
: 653:
: 654:
: 655:
: 656:
: 657:
: 658:
: 659:
: 660:
: 661:
: 662:
: 663:
: 664:
: 665:
: 666:
: 667:
: 668:
: 669:
: 670:
: 671:
: 672:
: 673:
: 674:
: 675:
: 676:
: 677:
: 678:
: 679:
: 680:
: 681:
: 682:
: 683:
: 684:
: 685:
: 686:
: 687:
: 688:
: 689:
: 690:
: 691:
: 692:
: 693:
: 694:
: 695:
: 696:
: 697:
: 698:
: 699:
: 700:
: 701:
: 702:
: 703:
: 704:
: 705:
: 706:
: 707:
: 708:
: 709:
: 710:
: 711:
: 712:
: 713:
: 714:
: 715:
: 716:
: 717:
: 718:
: 719:
: 720:
: 721:
: 722:
: 723:
: 724:
: 725:
: 726:
: 727:
: 728:
: 729:
: 730:
: 731:
: 732:
: 733:
: 734:
: 735:
: 736:
: 737:
: 738:
: 739:
: 740:
: 741:
: 742:
: 743:
: 744:
: 745:
: 746:
: 747:
: 748:
: 749:
: 750:
: 751:
: 752:
: 753:
: 754:
: 755:
: 756:
: 757:
: 758:
: 759:
: 760:
: 761:
: 762:
: 763:
: 764:
: 765:
: 766:
: 767:
: 768:
: 769:
: 770:
: 771:
: 772:
: 773:
: 774:
: 775:
: 776:
: 777:
: 778:
: 779:
: 780:
: 781:
: 782:
: 783:
: 784:
: 785:
: 786:
: 787:
: 788:
: 789:
: 790:
: 791:
: 792:
: 793:
: 794:
: 795:
: 796:
: 797:
: 798:
: 799:
: 800:
: 801:
: 802:
: 803:
: 804:
: 805:
: 806:
: 807:
: 808:
: 809:
: 810:
: 811:
: 812:
: 813:
: 814:
: 815:
: 816:
: 817:
: 818:
: 819:
: 820:
: 821:
: 822:
: 823:
: 824:
: 825:
: 826:
: 827:
: 828:
: 829:
: 830:
: 831:
: 832:
: 833:
: 834:
: 835:
: 836:
: 837:
: 838:
: 839:
: 840:
: 841:
: 842:
: 843:
: 844:
: 845:
: 846:
: 847:
: 848:
: 849:
: 850:
: 851:
: 852:
: 853:
: 854:
: 855:
: 856:
: 857:
: 858:
: 859:
: 860:
: 861:
: 862:
: 863:
: 864:
: 865:
: 866:
: 867:
: 868:
: 869:
: 870:
: 871:
: 872:
: 873:
: 874:
: 875:
: 876:
: 877:
: 878:
: 879:
: 880:
: 881:
: 882:
: 883:
: 884:
: 885:
: 886:
: 887:
: 888:
: 889:
: 890:
: 891:
: 892:
: 893:
: 894:
: 895:
: 896:
: 897:
: 898:
: 899:
: 900:
: 901:
: 902:
: 903:
: 904:
: 905:
: 906:
: 907:
: 908:
: 909:
: 910:
: 911:
: 912:
: 913:
: 914:
: 915:
: 916:
: 917:
: 918:
: 919:
: 920:
: 921:
: 922:
: 923:
: 924:
: 925:
: 926:
: 927:
: 928:
: 929:
: 930:
: 931:
: 932:
: 933:
: 934:
: 935:
: 936:
: 937:
: 938:
: 939:
: 940:
: 941:
: 942:
: 943:
: 944:
: 945:
: 946:
: 947:
: 948:
: 949:
: 950:
: 951:
: 952:
: 953:
: 954:
: 955:
: 956:
: 957:
: 958:
: 959:
: 960:
: 961:
: 962:
: 963:
: 964:
: 965:
: 966:
: 967:
: 968:
: 969:
: 970:
: 971:
: 972:
: 973:
: 974:
: 975:
: 976:
: 977:
: 978:
: 979:
: 980:
: 981:
: 982:
: 983:
: 984:
: 985:
: 986:
: 987:
: 988:
: 989:
: 990:
: 991:
: 992:
: 993:
: 994:
: 995:
: 996:
: 997:
: 998:
: 999:
: 1000:
: 1001:
: 1002:
: 1003:
: 1004:
: 1005:
: 1006:
: 1007:
: 1008:
: 1009:
: 1010:
: 1011:
: 1012:
: 1013:
: 1014:
: 1015:
: 1016:
: 1017:
: 1018:
: 1019:
: 1020:
: 1021:
: 1022:
: 1023:
: 1024:
: 1025:
: 1026:
: 1027:
: 1028:
: 1029:
: 1030:
: 1031:
: 1032:
: 1033:
: 1034:
: 1035:
: 1036:
: 1037:
: 1038:
: 1039:
: 1040:
: 1041:
: 1042:
: 1043:
: 1044:
: 1045:
: 1046:
: 1047:
: 1048:
: 1049:
: 1050:
: 1051:
: 1052:
: 1053:
: 1054:
: 1055:
: 1056:
: 1057:
: 1058:
: 1059:
: 1060:
: 1061:
: 1062:
: 1063:
: 1064:
: 1065:
: 1066:
: 1067:
: 1068:
: 1069:
: 1070:
: 1071:
: 1072:
: 1073:
: 1074:
: 1075:
: 1076:
: 1077:
: 1078:
: 1079:
: 1080:
: 1081:
: 1082:
: 1083:
: 1084:
: 1085:
: 1086:
: 1087:
: 1088:
: 1089:
: 1090:
: 1091:
: 1092:
: 1093:
: 1094:
: 1095:
: 1096:
: 1097:
: 1098:
: 1099:
: 1100:
: 1101:
: 1102:
: 1103:
: 1104:
: 1105:
: 1106:
: 1107:
: 1108:
: 1109:
: 1110:
: 1111:
: 1112:
: 1113:
: 1114:
: 1115:
: 1116:
: 1117:
: 1118:
: 1119:
: 1120:
: 1121:
: 1122:
: 1123:
: 1124:
: 1125:
: 1126:
: 1127:
: 1128:
: 1129:
: 1130:
: 1131:
: 1132:
: 1133:
: 1134:
: 1135:
: 1136:
: 1137:
: 1138:
: 1139:
: 1140:
: 1141:
: 1142:
: 1143:
: 1144:
: 1145:
: 1146:
: 1147:
: 1148:
: 1149:
: 1150:
: 1151:
: 1152:
: 1153:
: 1154:
: 1155:
: 1156:
: 1157:
: 1158:
: 1159:
: 1160:
: 1161:
: 1162:
: 1163:
: 1164:
: 1165:
: 1166:
: 1167:
: 1168:
: 1169:
: 1170:
: 1171:
: 1172:
: 1173:
: 1174:
: 1175:
: 1176:
: 1177:
: 1178:
: 1179:
: 1180:
: 1181:
: 1182:
: 1183:
: 1184:
: 1185:
: 1186:
: 1187:
: 1188:
: 1189:
: 1190:
: 1191:
: 1192:
: 1193:
: 1194:
: 1195:
: 1196:
: 1197:
: 1198:
: 1199:
: 1200:
: 1201:
: 1202:
: 1203:
: 1204:
: 1205:
: 1206:
: 1207:
: 1208:
: 1209:
: 1210:
: 1211:
: 1212:
: 1213:
: 1214:
: 1215:
: 1216:
: 1217:
: 1218:
: 1219:
: 1220:
: 1221:
: 1222:
: 1223:
: 1224:
: 1225:
: 1226:
: 1227:
: 1228:
: 1229:
: 1230:
: 1231:
: 1232:
: 1233:
: 1234:
: 1235:
: 1236:
: 1237:
: 1238:
: 1239:
: 1240:
: 1241:
: 1242:
: 1243:
: 1244:
: 1245:
: 1246:
: 1247:
: 1248:
: 1249:
: 1250:
: 1251:
: 1252:
: 1253:
: 1254:
: 1255:
: 1256:
: 1257:
: 1258:
: 1259:
: 1260:
: 1261:
: 1262:
: 1263:
: 1264:
: 1265:
: 1266:
: 1267:
: 1268:
: 1269:
: 1270:
: 1271:
: 1272:
: 1273:
: 1274:
: 1275:
: 1276:
: 1277:
: 1278:
: 1279:
: 1280:
: 1281:
: 1282:
: 1283:
: 1284:
: 1285:
: 1286:
: 1287:
: 1288:
: 1289:
: 1290:
: 1291:
: 1292:
: 1293:
: 1294:
: 1295:
: 1296:
: 1297:
: 1298:
: 1299:
: 1300:
: 1301:
: 1302:
: 1303:
: 1304:
: 1305:
: 1306:
: 1307:
: 1308:
: 1309:
: 1310:
: 1311:
: 1312:
: 1313:
: 1314:
: 1315:
: 1316:
: 1317:
: 1318:
: 1319:
: 1320:
: 1321:
: 1322:
: 1323:
: 1324:
: 1325:
: 1326:
: 1327:
: 1328:
: 1329:
: 1330:
: 1331:
: 1332:
: 1333:
: 1334:
: 1335:
: 1336:
: 1337:
: 1338:
: 1339:
: 1340:
: 1341:
: 1342:
: 1343:
: 1344:
: 1345:
: 1346:
: 1347:
: 1348:
: 1349:
: 1350:
: 1351:
: 1352:
: 1353:
: 1354:
: 1355:
: 1356:
: 1357:
: 1358:
: 1359:
: 1360:
: 1361:
: 1362:
: 1363:
: 1364:
: 1365:
: 1366:
: 1367:
: 1368:
: 1369:
: 1370:
: 1371:
: 1372:
: 1373:
: 1374:
: 1375:
: 1376:
: 1377:
: 1378:
: 1379:
: 1380:
: 1381:
: 1382:
: 1383:
: 1384:
: 1385:
: 1386:
: 1387:
: 1388:
: 1389:
: 1390:
: 1391:
: 1392:
: 1393:
: 1394:
: 1395:
: 1396:
: 1397:
: 1398:
: 1399:
: 1400:
: 1401:
: 1402:
: 1403:
: 1404:
: 1405:
: 1406:
: 1407:
: 1408:
: 1409:
: 1410:
: 1411:
: 1412:
: 1413:
: 1414:
: 1415:
: 1416:
: 1417:
: 1418:
: 1419:
: 1420:
: 1421:
: 1422:
: 1423:
: 1424:
: 1425:
: 1426:
: 1427:
: 1428:
: 1429:
: 1430:
: 1431:
: 1432:
: 1433:
: 1434:
: 1435:
: 1436:
: 1437:
: 1438:
: 1439:
: 1440:
: 1441:
: 1442:
: 1443:
: 1444:
: 1445:
: 1446:
: 1447:
: 1448:
: 1449:
: 1450:
: 1451:
: 1452:
: 1453:
: 1454:
: 1455:
: 1456:
: 1457:
: 1458:
: 1459:
: 1460:
: 1461:
: 1462:
: 1463:
: 1464:
: 1465:
: 1466:
: 1467:
: 1468:
: 1469:
: 1470:
: 1471:
: 1472:
: 1473:
: 1474:
: 1475:
: 1476:
: 1477:
: 1478:
: 1479:
: 1480:
: 1481:
: 1482:
: 1483:
: 1484:
: 1485:
: 1486:
: 1487:
: 1488:
: 1489:
: 1490:
: 1491:
: 1492:
: 1493:
: 1494:
: 1495:
: 1496:
: 1497:
: 1498:
: 1499:
: 1500:
: 1501:
: 1502:
: 1503:
: 1504:
: 1505:
: 1506:
: 1507:
: 1508:
: 1509:
: 1510:
: 1511:
: 1512:
: 1513:
: 1514:
: 1515:
: 1516:
: 1517:
: 1518:
: 1519:
: 1520:
: 1521:
: 1522:
: 1523:
: 1524:
: 1525:
: 1526:
: 1527:
: 1528:
: 1529:
: 1530:
: 1531:
: 1532:
: 1533:
: 1534:
: 1535:
: 1536:
: 1537:
: 1538:
: 1539:
: 1540:
: 1541:
: 1542:
: 1543:
: 1544:
: 1545:
: 1546:
: 1547:
: 1548:
: 1549:
: 1550:
: 1551:
: 1552:
: 1553:
: 1554:
: 1555:
: 1556:
: 1557:
: 1558:
: 1559:
: 1560:
: 1561:
: 1562:
: 1563:
: 1564:
: 1565:
: 1566:
: 1567:
: 1568:
: 1569:
: 1570:
: 1571:
: 1572:
: 1573:
: 1574:
: 1575:
: 1576:
: 1577:
: 1578:
: 1579:
: 1580:
: 1581:
: 1582:
: 1583:
: 1584:
: 1585:
: 1586:
: 1587:
: 1588:
: 1589:
: 1590:
: 1591:
: 1592:
: 1593:
: 1594:
: 1595:
: 
```



Q8. Convert the following program

ANS :

```
Q8.asm - Notepad
File Edit Format View Help

.text
main:
    li $v0, 4
    la $a0, result1
    syscall

    li $v0, 1
    li $a0, 5
    syscall

    li $v0, 4
    la $a0, result2
    syscall

    li $v0, 1
    li $a0, 8
    syscall

    addi $v0, $zero, 10      #Exit program
    syscall

.data
result1: .asciiz "\nfirst value = "
result2: .asciiz "\nsecond value = "
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC	=	400060
EPC	=	0
Cause	=	0
BadVAddr	=	0
Status	=	3000fff10
HI	=	0
LO	=	0
R0 [r0]	=	0
R1 [at]	=	10010000
R2 [v0]	=	a
R3 [v1]	=	0
R4 [a0]	=	8
R5 [a1]	=	7ffff590
R6 [a2]	=	7ffff5a0
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

Text

```
[00400000] 8fa40000 lw $4, 0($29)
[00400004] 27a50004 addiu $5, $29, 4
[00400008] 24a60004 addiu $6, $5, 4
[0040000c] 00041080 sll $2, $4, 2
[00400010] 00c23021 addn $6, $6, $2
[00400014] 0c100009 jal 0x00400024 [main]
[00400018] 00000000 nop
[0040001c] 3402000a ori $2, $0, 10
[00400020] 0000000c syscall
[00400024] 34020004 ori $2, $0, 4
[00400028] 3c011001 lui $1, 4097 [result1]
[0040002c] 34240000 ori $4, $1, 0 [result1]
[00400030] 0000000c syscall
[00400034] 34020001 ori $2, $0, 1
[00400038] 34040005 ori $4, $0, 5
[0040003c] 0000000c syscall
[00400040] 34020004 ori $2, $0, 4
[00400044] 3c011001 lui $1, 4097 [result2]
[00400048] 34240010 ori $4, $1, 16 [result2]
[0040004c] 0000000c syscall
[00400050] 34020001 ori $2, $0, 1
[00400054] 34040008 ori $4, $0, 8
[00400058] 0000000c syscall
[0040005c] 2002000a addi $2, $0, 10
[00400060] 0000000c syscall
```

User Text Segment [00400000]..[00440000]

```
; 183: lw $a0 0($sp) # argc
; 184: addiu $a1 $sp 4 # argv
; 185: addiu $a2 $a1 4 # envp
; 186: sll $v0 $a0 2
; 187: addu $a2 $a2 $v0
; 188: jal main
; 189: nop
; 191: li $v0 10
; 192: syscall # syscall 10 (exit)
; 3: li $v0, 4
; 4: la $a0, result1
; 5: syscall
; 7: li $v0, 1
; 8: li $a0, 5
; 9: syscall
; 11: li $v0, 4
; 12: la $a0, result2
; 13: syscall
; 15: li $v0, 1
; 16: li $a0, 8
; 17: syscall
; 19: addi $v0, $zero, 10 #Exit program
; 20: syscall
```

Kernel Text Segment [80000000]..[80010000]

```
; 90: move $k1 $at # Save $at
; 92: sv $v0 $1 # Not re-entrant and we can't trust $sp
; 93: sv $a0 $2 # But we need to use these registers
```

Console

```
first value = 5
second value = 8
```

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)



Q9. Write a program to retrieve two numbers from a user, and swap those number using only the XOR operation. You should not use a temporary variable to store the numbers while swapping them. Your program should include a proper and useful prompt for input, and print the results in a meaningful manner.

ANS :

```
*swap_value.asm - Notepad
File Edit Format View Help
.data
prompt1: .ascii "Enter the first number: "
prompt2: .ascii "Enter the second number: "
result_message: .ascii "After swapping:\nFirst number: "
result_message2: .ascii "\nSecond number: "
.text
.globl main
main:
    li $v0, 4          # Prompt for the first number
    la $a0, prompt1
    syscall

    li $v0, 5          # Read the first number from the user
    syscall
    move $t0, $v0

    li $v0, 4          # Prompt for the second number
    la $a0, prompt2
    syscall

    li $v0, 5          # Read the second number from the user
    syscall
    move $t1, $v0

    xor $t0, $t0, $t1  # Swap the numbers using XOR without a temporary variable
    xor $t1, $t0, $t1
    xor $t0, $t0, $t1

    li $v0, 4          # Display the results
    la $a0, result_message
    syscall

    li $v0, 1          # Print the first number
    move $a0, $t0
    syscall

    li $v0, 4          # Display the results
    la $a0, result_message2
    syscall

    li $v0, 1          # Print the second number
    move $a0, $t1
    syscall

    li $v0, 10         # Exit the program
    syscall
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs

Int Regs [16]

Data

Text

Int Regs [16]

PC = 4000a0

EPC = 0

Cause = 0

BadVAddr = 0

Status = 3000ff10

HI = 0

LO = 0

R0 [r0] = 0

R1 [at] = 10010000

R2 [v0] = a

R3 [v1] = 0

R4 [a0] = 5

R5 [a1] = 7ffff588

R6 [a2] = 7ffff598

R7 [a3] = 0

R8 [t0] = 2

R9 [t1] = 5

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

Text

[00400018] 00000000 nop ; 189: nop

[0040001c] 3402000a ori \$2, \$0, 10 ; 191: li \$v0 10

[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)

[00400024] 34020004 ori \$2, \$0, 4 ; 12: li \$v0, 4

[00400028] 3c041001 lui \$4, 4097 [prompt1] ; 13: la \$a0, prompt1

[0040002c] 0000000c syscall ; 14: syscall

[00400030] 34020005 ori \$2, \$0, 5 ; 17: li \$v0, 5

[00400034] 0000000c syscall ; 18: syscall

[00400038] 00024021 addu \$8, \$0, \$2 ; 19: move \$t0, \$v0

[0040003c] 34020004 ori \$2, \$0, 4 ; 22: li \$v0, 4

[00400040] 3c011001 lui \$1, 4097 [prompt2] ; 23: la \$a0, prompt2

[00400044] 34240019 ori \$4, \$1, 25 [prompt2] ; 24: syscall

[00400048] 0000000c syscall ; 27: li \$v0, 5

[0040004c] 34020005 ori \$2, \$0, 5 ; 28: syscall

[00400050] 0000000c syscall ; 29: move \$t1, \$v0

[00400054] 00024821 addu \$9, \$0, \$2 ; 32: xor \$t0, \$t0, \$t1

[00400058] 01094026 xor \$8, \$8, \$9 ; 33: xor \$t1, \$t0, \$t1

[0040005c] 01094826 xor \$9, \$8, \$9 ; 34: xor \$t0, \$t0, \$t1

[00400060] 01094026 xor \$8, \$8, \$9 ; 37: li \$v0, 4

[00400064] 34020004 ori \$2, \$0, 4 ; 38: la \$a0, result\_mes

[00400068] 3c011001 lui \$1, 4097 [result\_message]; 39: syscall

[0040006c] 34240033 ori \$4, \$1, 51 [result\_message] ; 42: li \$v0, 1

[00400070] 0000000c syscall ; 43: move \$a0, \$t0

[00400074] 34020001 ori \$2, \$0, 1 ; 44: syscall

[00400078] 00082021 addu \$4, \$0, \$8 ; 46: li \$v0, 4

[0040007c] 0000000c syscall ; 47: la \$a0, result\_message2

[00400080] 34020004 ori \$2, \$0, 4 ; 48: syscall

[00400084] 3c011001 lui \$1, 4097 [result\_message2]; 51: li \$v0, 1

[00400088] 34240052 ori \$4, \$1, 82 [result\_message2] ; 52: move \$a0, \$t1

[0040008c] 0000000c syscall ; 53: syscall

[00400090] 34020001 ori \$2, \$0, 1

[00400094] 00092021 addu \$4, \$0, \$9

[00400098] 0000000c syscall

Memory and registers cleared

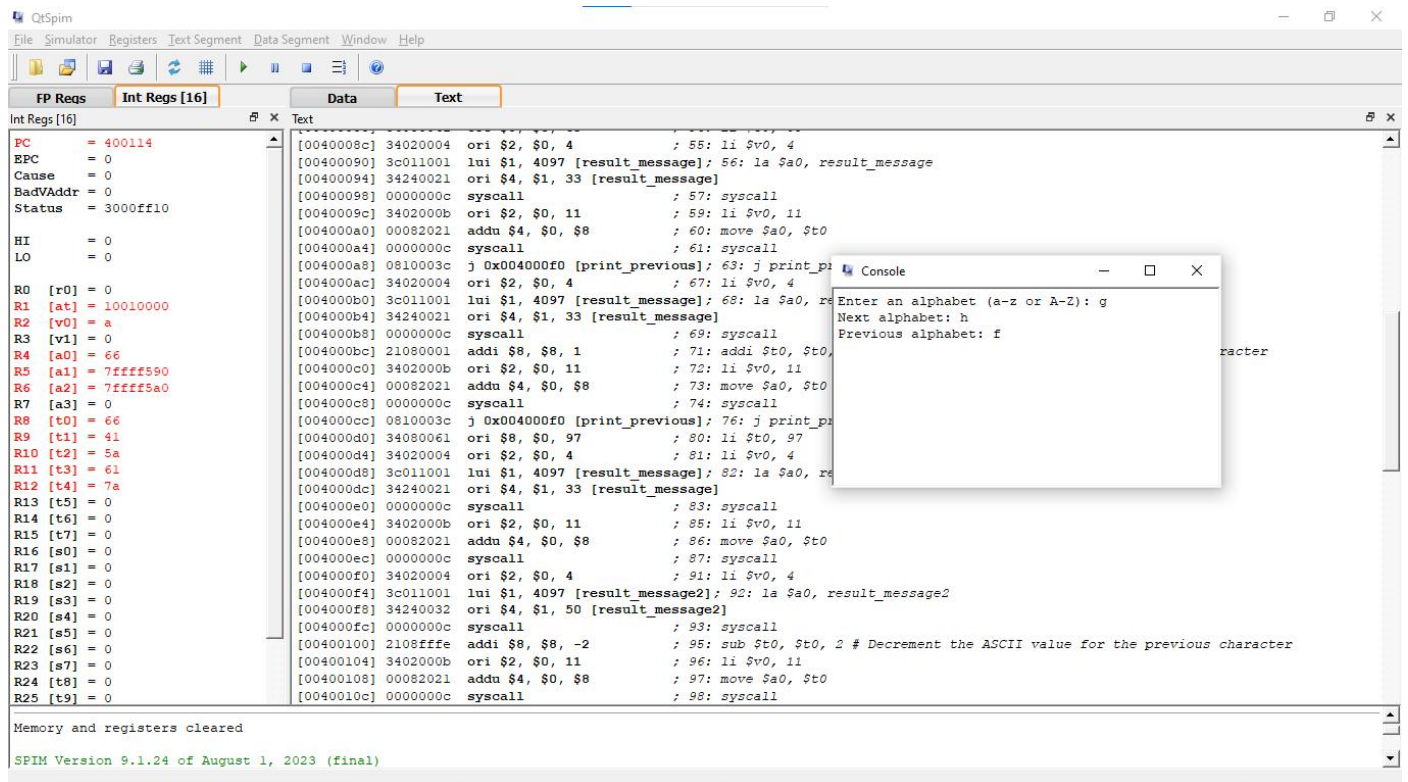
SPIM Version 9.1.24 of August 1, 2023 (final)

Console

Enter the first number: 5  
Enter the second number: 2  
After swapping:  
First number: 2  
Second number: 5

Q10. Write an Assembly code which take an alphabet from user and print the next and previous alphabets on the screen.

ANS :



The screenshot displays the QtSpim MIPS simulator interface. The 'Text' window shows the assembly code for a program that takes an input character and prints its next and previous alphabets. The 'Registers' window shows the state of the registers, with \$a0 containing the input character 'g'. The 'Console' window shows the program's output: 'Enter an alphabet (a-z or A-Z): g', 'Next alphabet: h', and 'Previous alphabet: f'. The status bar at the bottom indicates 'Memory and registers cleared' and 'SPIM Version 9.1.24 of August 1, 2023 (final)'.

```
PC = 400114
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 66
R5 [a1] = 7ffff90
R6 [a2] = 7ffff9a0
R7 [a3] = 0
R8 [t0] = 66
R9 [t1] = 41
R10 [t2] = 5a
R11 [t3] = 61
R12 [t4] = 7a
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

[0040008c] 34020004 ori $2, $0, 4 ; 55: li $v0, 4
[00400090] 3c011001 lui $1, 4097 [result_message]; 56: la $a0, result_message
[00400094] 34240021 ori $4, $1, 33 [result_message]
[00400098] 0000000c syscall ; 57: syscall
[0040009c] 3402000b ori $2, $0, 11 ; 59: li $v0, 11
[004000a0] 00082021 addu $4, $0, $8 ; 60: move $a0, $t0
[004000a4] 0000000c syscall ; 61: syscall
[004000a8] 0810003c j 0x004000f0 [print_previous]; 63: j print_previous
[004000ac] 34020004 ori $2, $0, 4 ; 67: li $v0, 4
[004000b0] 3c011001 lui $1, 4097 [result_message]; 68: la $a0, result_message
[004000b4] 34240021 ori $4, $1, 33 [result_message]
[004000b8] 0000000c syscall ; 69: syscall
[004000bc] 21080001 addi $8, $8, 1 ; 71: addi $t0, $t0, 1
[004000c0] 3402000b ori $2, $0, 11 ; 72: li $v0, 11
[004000c4] 00082021 addu $4, $0, $8 ; 73: move $a0, $t0
[004000c8] 0000000c syscall ; 74: syscall
[004000cc] 0810003c j 0x004000f0 [print_previous]; 76: j print_previous
[004000d0] 34080061 ori $8, $0, 97 ; 80: li $t0, 97
[004000d4] 34020004 ori $2, $0, 4 ; 81: li $v0, 4
[004000d8] 3c011001 lui $1, 4097 [result_message]; 82: la $a0, result_message
[004000dc] 34240021 ori $4, $1, 33 [result_message]
[004000e0] 0000000c syscall ; 83: syscall
[004000e4] 3402000b ori $2, $0, 11 ; 85: li $v0, 11
[004000e8] 00082021 addu $4, $0, $8 ; 86: move $a0, $t0
[004000ec] 0000000c syscall ; 87: syscall
[004000f0] 34020004 ori $2, $0, 4 ; 91: li $v0, 4
[004000f4] 3c011001 lui $1, 4097 [result_message2]; 92: la $a0, result_message2
[004000f8] 34240032 ori $4, $1, 50 [result_message2]
[004000fc] 0000000c syscall ; 93: syscall
[00400100] 2108fffe addi $8, $8, -2 ; 95: sub $t0, $t0, 2 # Decrement the ASCII value for the previous character
[00400104] 3402000b ori $2, $0, 11 ; 96: li $v0, 11
[00400108] 00082021 addu $4, $0, $8 ; 97: move $a0, $t0
[0040010c] 0000000c syscall ; 98: syscall
```

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)

## Q11. Implementation of Fibonacci series program In MIPS.

ANS :

```
Fibonacci.asm - Notepad
File Edit Format View Help
.data
    result_message: .asciiz "Fibonacci Series: |"

.text
.globl main

main:
    # Initialize variables
    li $t0, 0          # First Fibonacci number (F(0))
    li $t1, 1          # Second Fibonacci number (F(1))
    li $t2, 10         # Number of Fibonacci numbers to generate
    li $t3, 0          # Counter

    li $v0, 4          # Print the result message
    la $a0, result_message
    syscall

fib_loop:
    li $v0, 1          # Print the current Fibonacci number
    move $a0, $t0
    syscall

    # Calculate the next Fibonacci number
    add $t4, $t0, $t1 # F(n) = F(n-1) + F(n-2)

    # Swap variables to prepare for the next iteration
    move $t0, $t1 # F(n-2) = F(n-1)
    move $t1, $t4 # F(n-1) = F(n)

    addi $t3, $t3, 1    # Increment the counter

    # Check if we have generated the desired number of Fibonacci numbers
    beq $t3, $t2, exit
    j fib_loop

exit:
    li $v0, 10         # Exit the program
    syscall
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Reqs Int Regs [16] Data Text

Int Regs [16]

PC = 400068  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000ff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 0  
R2 [v0] = a  
R3 [v1] = 0  
R4 [a0] = 22  
R5 [a1] = 7ffff588  
R6 [a2] = 7ffff598  
R7 [a3] = 0  
R8 [t0] = 37  
R9 [t1] = 59  
R10 [t2] = a  
R11 [t3] = a  
R12 [t4] = 59  
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

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
[00400024] 34080000 ori $8, $0, 0 ; 9: li $t0, 0 # First Fi
[00400028] 34090001 ori $9, $0, 1 ; 10: li $t1, 1 # Second
[0040002c] 340a000a ori $10, $0, 10 ; 11: li $t2, 10 # Number
[00400030] 340b0000 ori $11, $0, 0 ; 12: li $t3, 0 # Counter
[00400034] 34020004 ori $2, $0, 4 ; 14: li $v0, 4 # Print t
[00400038] 3c041001 lui $4, 4097 [result_message]; 15: la $a0, result_
[0040003c] 0000000c syscall ; 16: syscall
[00400040] 34020001 ori $2, $0, 1 ; 19: li $v0, 1 # Print t
[00400044] 00082021 addu $4, $0, $8 ; 20: move $a0, $t0
[00400048] 0000000c syscall ; 21: syscall
[0040004c] 01096020 add $12, $8, $9 ; 24: add $t4, $t0, $t1 # F(n) = F(n-1) + F(n-2)
[00400050] 00094021 addu $8, $0, $9 ; 27: move $t0, $t1 # F(n-2) = F(n-1)
[00400054] 000c4821 addu $9, $0, $12 ; 28: move $t1, $t4 # F(n-1) = F(n)
[00400058] 216b0001 addi $11, $11, 1 ; 30: addi $t3, $t3, 1 # Increment the counter
[0040005c] 116a0002 beq $11, $10, 8 [exit-0x0040005c]
[00400060] 08100010 j 0x00400040 [fib_loop] ; 34: j fib_loop
[00400064] 3402000a ori $2, $0, 10 ; 37: li $v0, 10 # Exit the program
[00400068] 0000000c syscall ; 38: syscall

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 $1 # Not re-entrant and we can't trust $sp
[80000188] ac220200 sw $2, 512($1)
```

Console

Fibonacci Series:  
0112358132134

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)



Q12. Write a MIPS program that inputs two integer values. The program should output equal if the two integers are equal. Otherwise, it should output not equal. Use the branch instruction to check for equality.

ANS :

```
*check_equalOrNot.asm - Notepad
File Edit Format View Help
.data
prompt1: .asciiz "Enter the first integer: "
prompt2: .asciiz "Enter the second integer: "
equal_message: .asciiz "equal"
not_equal_message: .asciiz "not equal"
.text
.globl main
main:
    li $v0, 4                # Prompt for the first integer
    la $a0, prompt1
    syscall

    li $v0, 5                # Read the first integer from the user
    syscall
    move $t0, $v0

    li $v0, 4                # Prompt for the second integer
    la $a0, prompt2
    syscall

    li $v0, 5                # Read the second integer from the user
    syscall
    move $t1, $v0

    beq $t0, $t1, equal      # Compare the two integers
    j not_equal

equal:
    li $v0, 4                # Output "equal" if integers are equal
    la $a0, equal_message
    syscall
    j exit

not_equal:
    li $v0, 4                # Output "not equal" if integers are not equal
    la $a0, not_equal_message
    syscall

exit:
    li $v0, 10               # Exit the program
    syscall
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC	=	400084
EPC	=	0
Cause	=	0
BadVAddr	=	0
Status	=	3000ff10
HI	=	0
LO	=	0
R0 [r0]	=	0
R1 [at]	=	10010000
R2 [v0]	=	a
R3 [v1]	=	0
R4 [a0]	=	1001003b
R5 [a1]	=	7ffff3d4
R6 [a2]	=	7ffff3e0
R7 [a3]	=	0
R8 [t0]	=	4
R9 [t1]	=	6
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

Text

```
[0040000c] sll $2, $4, 2
[00400010] addu $6, $6, $2
[00400014] jal 0x00400024 [main]
[00400018] nop
[0040001c] ori $2, $0, 10
[00400020] syscall
[00400024] ori $2, $0, 4
[00400028] lui $4, 4097 [prompt1]
[0040002c] syscall
[00400030] ori $2, $0, 5
[00400034] syscall
[00400038] addu $8, $0, $2
[0040003c] ori $2, $0, 4
[00400040] lui $1, 4097 [prompt2]
[00400044] ori $4, $1, 26 [prompt2]
[00400048] syscall
[0040004c] ori $2, $0, 5
[00400050] syscall
[00400054] addu $9, $0, $2
[00400058] beq $8, $9, 24 [equal-0x00400058]
[0040005c] ori $2, $0, 4
[00400060] lui $1, 4097 [not_equal_msg]
[00400064] ori $4, $1, 59 [not_equal_msg]
[00400068] syscall
[0040006c] j 0x00400080 [exit_program]
[00400070] ori $2, $0, 4
[00400074] lui $1, 4097 [equal_msg]
[00400078] ori $4, $1, 53 [equal_msg]
[0040007c] syscall
[00400080] ori $2, $0, 10
[00400084] syscall
```

Console

```
Enter the first integer: 4
Enter the second integer: 6
not equal
```

Copyright 1990-2021 By James 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.

Q13. Variables are of type unsigned integers and word sized. Also the variables b, c, d and e are initialized to values 10, 20, 30 and 40 respectively.

$$a = (b + c) - (d + e)$$

ANS :

```
Q13.asm - Notepad
File Edit Format View Help
.data
# Define the variables
a: .word 0
B: .word 10
c: .word 20
d: .word 30
e: .word 40
result_message: .asciiz "Result: "

.text
.globl main

main:
# Load the values of b, c, d, and e into registers
lw $t0, B
lw $t1, c
lw $t2, d
lw $t3, e

# Perform the calculation: a = (B + c) - (d + e)
add $t4, $t0, $t1 # $t4 = B + c
add $t5, $t2, $t3 # $t5 = d + e
sub $t6, $t4, $t5 # $t6 = (B + c) - (d + e)

# Store the result in variable a
sw $t6, a

# Print the result
li $v0, 4
la $a0, result_message
syscall

li $v0, 1
lw $a0, a
syscall

# Exit the program
li $v0, 10
syscall
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC	= 40007c
EPC	= 0
Cause	= 0
BadVAddr	= 0
Status	= 3000fff10
HI	= 0
LO	= 0
R0 [r0]	= 0
R1 [at]	= 10010000
R2 [v0]	= a
R3 [v1]	= 0
R4 [a0]	= ffffffff3
R5 [a1]	= 7ffff590
R6 [a2]	= 7ffff5a0
R7 [a3]	= 0
R8 [t0]	= a
R9 [t1]	= 14
R10 [t2]	= 1e
R11 [t3]	= 23
R12 [t4]	= 1e
R13 [t5]	= 46
R14 [t6]	= ffffffff3
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

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] 3c011001 lui $1, 4097 ; 15: lw $t0, B
[00400028] 8c280004 lw $8, 4($1) ; 16: lw $t1, c
[0040002c] 3c011001 lui $1, 4097 ; 17: lw $t2, d
[00400030] 8c290008 lw $9, 8($1) ; 18: lw $t3, e
[00400034] 3c011001 lui $1, 4097
[00400038] 8c2a000c lw $10, 12($1)
[0040003c] 3c011001 lui $1, 4097
[00400040] 8c2b0010 lw $11, 16($1)
[00400044] 01096020 add $12, $8, $9 ; 21: add $t4, $t0, $t1 # $t4 = B + c
[00400048] 014b6820 add $13, $10, $11 ; 22: add $t5, $t2, $t3 # $t5 = d + e
[0040004c] 018d7022 sub $14, $12, $13 ; 23: sub $t6, $t4, $t5 # $t6 = (B + c) - (d + e)
[00400050] 3c011001 lui $1, 4097 ; 26: sw $t6, a
[00400054] ac2e0000 sw $14, 0($1)
[00400058] 3402000a ori $2, $0, 4 ; 29: li $v0, 4
[0040005c] 3c011001 lui $1, 4097 [result_message]; 30: la $a0, result_message
[00400060] 34240014 ori $4, $1, 20 [result_message]
[00400064] 0000000c syscall ; 31: syscall
[00400068] 34020001 ori $2, $0, 1 ; 33: li $v0, 1
[0040006c] 3c011001 lui $1, 4097 ; 34: lw $a0, a
[00400070] 8c240000 lw $4, 0($1)
[00400074] 0000000c syscall ; 35: syscall
[00400078] 3402000a ori $2, $0, 10 ; 38: li $v0, 10
[0040007c] 0000000c syscall ; 39: syscall
```

Console

Result: -40

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)

Q14. Write MIPS Assembly Language Programs equivalent to the following C-code fragments.

```
1. if(a<b)
{
    c = a - b;
}
```

//a and b are signed integers

```
alessb.asm - Notepad
File Edit Format View Help

.data
    a: .word 5
    B: .word 10
    c: .word 0

.text
.globl main

main:
    lw $t0, a      # Load a into $t0
    lw $t1, B      # Load B into $t1

    slt $t2, $t0, $t1 # Set $t2 to 1 if $t0 < $t1, 0 otherwise
    beq $t2, $zero, not_less

    sub $t3, $t0, $t1 # c = a - B
    sw $t3, c        # Store the result in c
    j exit

not_less:
    # Handle the case when a >= B (c not calculated)

exit:
    # Exit the program
    li $v0, 10
    syscall

Ln 2, Col 15    100%    Windows (CRLF)    UTF-8
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Reqs Int Reqs [16] Data Text

Int Reqs [16]

PC = 400050  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000ff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 10010000  
R2 [v0] = a  
R3 [v1] = 0  
R4 [a0] = 3  
R5 [a1] = 7ffff590  
R6 [a2] = 7ffff5a0  
R7 [a3] = 0  
R8 [t0] = 5  
R9 [t1] = a  
R10 [t2] = 1  
R11 [t3] = ffffffff  
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

Text

User Text Segment [00400000]..[00440000]

```
[00400000] 0fa40000 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] 3c011001 lui $1, 4097 ; 10: lw $t0, a # Load a into $t0
[00400028] 8c280000 lw $8, 0($1) ; 11: lw $t1, B # Load B into $t1
[0040002c] 3c011001 lui $1, 4097 ; 13: slt $t2, $t0, $t1 # Set $t2 to 1 if $t0
[00400030] 8c290004 lw $9, 4($1) ; 16: sub $t3, $t0, $t1 # c = a - B
[00400034] 0109502a slt $10, $8, $9 ; 17: sw $t3, c # Store the result in c
[00400038] 11400005 beq $10, $0, 20 [not_less-0x00400038] ; 18: j exit
[0040003c] 01095822 sub $11, $8, $9 ; 25: li $v0, 10
[00400040] 3c011001 lui $1, 4097 ; 26: syscall
[00400044] ac2b0008 sw $11, 8($1) ; 
[00400048] 08100013 j 0x0040004c [exit] ; 
[0040004c] 3402000a ori $2, $0, 10 ; 
[00400050] 0000000c syscall ; 

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 $1 # Not re-entrant and we can't trust $sp
[80000188] ac220200 sw $2, 512($1) ; 93: sw $a0 $2 # But we need to use these registers
[8000018c] 3c019000 lui $1, -28672 ; 
[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)
```

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)



```

2. if(a<b)
{
    // a and b are unsigned integers
    c=a-b;
}

```

```

Unsigned_aleessb.asm - Notepad
File Edit Format View Help
.data
a: .word 5
B: .word 10
c: .word 0

.text
.globl main

main:
    lw $t0, a      # Load a into $t0
    lw $t1, B      # Load b into $t1

    #sltu $t2, $t0, $t1 # Set $t2 to 1 if $t0 < $t1, 0 otherwise
    #beq $t2, $zero, not_less

    subu $t3, $t0, $t1 # c = a - B
    sw $t3, c        # Store the result in c
    j exit

not_less:
    # Handle the case when a >= b (c not calculated)

exit:
    # Exit the program
    li $v0, 10
    syscall

```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 400048  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000fff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 10010000  
R2 [v0] = a  
R3 [v1] = 0  
R4 [a0] = 3  
R5 [a1] = 7ffff588  
R6 [a2] = 7ffff598  
R7 [a3] = 0  
R8 [t0] = 5  
R9 [t1] = a  
R10 [t2] = 0  
R11 [t3] = ffffffff  
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

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] 3c011001 lui $1, 4097 ; 19: lw $t0, a # Load a into $t0
[00400028] 8c280000 lw $8, 0($1) ; 10: lw $t0, a # Load a into $t0
[0040002c] 3c011001 lui $1, 4097 ; 11: lw $t1, B # Load b into $t1
[00400030] 8c290004 lw $9, 4($1) ; 11: lw $t1, B # Load b into $t1
[00400034] 01095823 subu $11, $8, $9 ; 16: subu $t3, $t0, $t1 # c = a - B
[00400038] 3c011001 lui $1, 4097 ; 17: sw $t3, c # Store the result in c
[0040003c] ac2b0008 sw $11, 8($1) ; 17: sw $t3, c # Store the result in c
[00400040] 08100011 j 0x00400044 [exit] ; 18: j exit
[00400044] 3402000a ori $2, $0, 10 ; 25: li $v0, 10
[00400048] 0000000c syscall ; 26: syscall

```

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 $1 # Not re-entrant and we can't trust $sp
[80000188] ac220200 sw $2, 512($1) ; 92: sw $v0 $1 # Not re-entrant and we can't trust $sp
[8000018c] 3c019000 lui $1, -28672 ; 93: sv $a0 $s2 # But we need to use these registers
[80000190] ac240204 sw $4, 516($1) ; 93: sv $a0 $s2 # But we need to use these registers
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2 # Extract ExCoDe 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

```

Memory and registers cleared

SPIM Version 9.1.24 of August 1, 2023 (final)

```

3. if( a < -1234)
{
    // a is a signed integer
    a = 4 * a;
}
else
{
    a = a/4;
}

```

The image shows a Notepad window titled 'mul\_aleseb.asm - Notepad' containing MIPS assembly code. Below the code editor is the QtSpim MIPS simulator interface, which includes a menu bar, a toolbar, and a multi-pane view showing registers, memory, and disassembly.

**Assembly Code (mul\_aleseb.asm):**

```

.data
a: .word -2000

.text
.globl main

main:
    lw $t0, a      # Load a into $t0

    li $t1, -1234  # Load -1234 into $t1

    slt $t2, $t0, $t1 # Set $t2 to 1 if $t0 < $t1, 0 otherwise
    beq $t2, $zero, not_less

    sll $t0, $t0, 2  # a = 4 * a
    j exit

not_less:
    sra $t0, $t0, 2  # a = a / 4

exit:
    # Exit the program
    li $v0, 10
    syscall

```

**QtSpim Simulator Interface:**

- Registers:** The 'Int Regs [16]' pane shows the state of MIPS registers. For example, \$t0 contains 40004c, \$t1 contains ffffff10, and \$t2 contains 1.
- Disassembly:** The main pane shows the assembly code being executed, with comments explaining each instruction. For example, 'lw \$t0, a' is commented as 'Load a into \$t0'.
- Memory:** The 'Data' pane shows the memory layout, including the .data segment where variable 'a' is located at address 00400000.

## 15. Write a MIPS assembly for the following C codes :

int a, b, result

```
int main(){
```

```
    a = 0x12345;
```

```
    b = 7;
```

```
    result = a + b;
```

```
    return 1;
```

```
}
```

ANS :

```
*15_c_programme.asm - Notepad
File Edit Format View Help

.data
    a: .word 0x12345      # Initialize 'a' with 0x12345
    #b: .word 7           # Initialize 'b' with 7(Error occured in this line)
    result: .word 0       # Initialize 'result' with 0

.text
.globl main

main:
    # Load the value of 'a' into a register
    lw $t0, a

    # Load the value of 'b' into a register
    #lw $t1, b
    ori $t1,$0,0x7        #This is use for solve the error of variable b

    # Add 'a' and 'b' and store the result in 'result'
    add $t2, $t0, $t1
    sw $t2, result

    # Set the return value to 1
    li $v0, 10             # Exit syscall code
    li $a0, 1              # Exit status 1
    syscall

Ln 24, Col 12
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 400044  
EPC = 0  
Cause = 0  
BadVAddr = 0  
Status = 3000ff10  
HI = 0  
LO = 0  
R0 [r0] = 0  
R1 [at] = 10010000  
R2 [v0] = a  
R3 [v1] = 0  
R4 [a0] = 1  
R5 [a1] = 7ffff588  
R6 [a2] = 7ffff598  
R7 [a3] = 0  
R8 [t0] = 12345  
R9 [t1] = 7  
R10 [t2] = 1234c  
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

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] 3c011001 lui $1, 4097 ; 11: lw $t0, a
[00400028] 3c020000 lw $8, 0($1) ; 15: ori $t1,$0,0x7
[0040002c] 34090007 ori $9, $0, 7 ; 18: add $t2, $t0, $t1
[00400030] 01095020 add $10, $8, $9 ; 19: sw $t2, result
[00400034] 3c011001 lui $1, 4097 ; 22: li $v0, 10 # Exit syscall code
[00400038] ac2a0004 sw $10, 4($1) ; 23: li $a0, 1 # Exit status 1
[0040003c] 3402000a ori $2, $0, 10 ; 24: syscall
[00400040] 34040001 ori $4, $0, 1
[00400044] 0000000c syscall

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

spim: (parser) syntax error on line 14 of file D:/college/coa/assignment lab 3/input\_str.asm