

ENGG*3380: Computer Organization and Design
January 28th, 2024

Lab # 2 - MIPS Simulators (MARS and QtSpim)

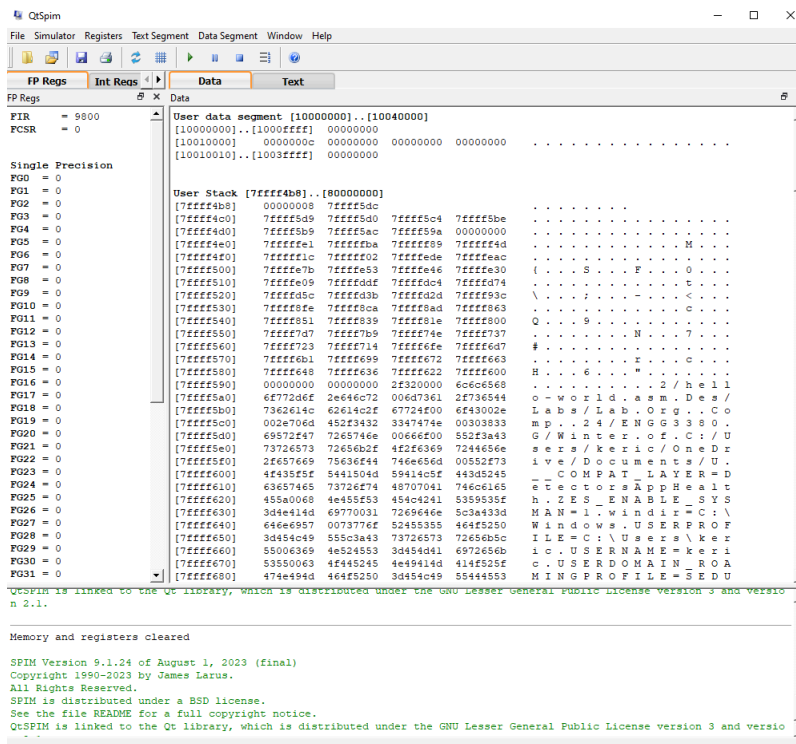
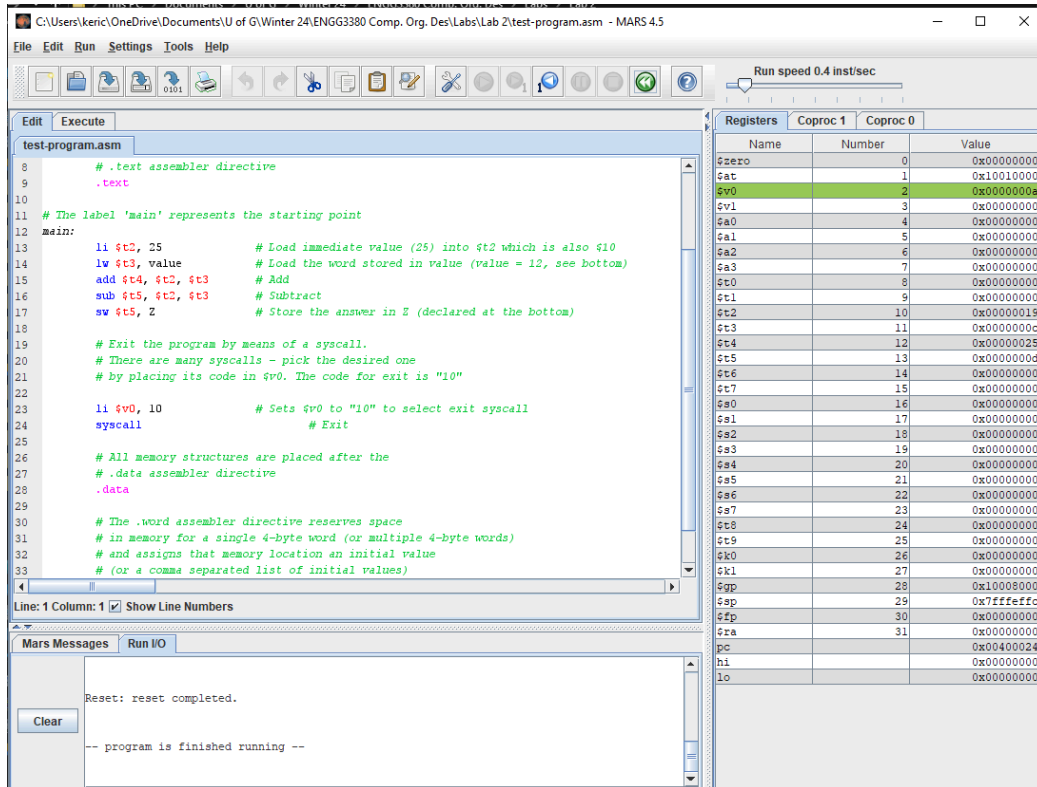
Lab Instructor: Haleh Vahedi

Group #37:

Members: Eric Yates, Lukas Krampitz, Alec McBurney

1. Test Program

a) Screenshot of QtSpim and MARS with test-program.asm loaded.



b) How many instructions are executed?

There are 9 instructions that are executed in total.

c) What registers are used?

The registers that are used are: \$0, \$v0, \$a0, \$t2, \$t3, \$t4, \$t5,

d) What addresses in Memory are changed?

The address that is changed is: 0x10010004 also called Z. The memory address 0x10010004 received the result of the subtraction on line 16 of the assembly. This stores a word value of 13 into address 0x10010004.

e) List and explain the syscalls that were used.

The syscall used is “10” to exit the program, and safely return to the Operating System.

2. Hello World

a) Show a screen-shot with our names.

C:\Users\keric\OneDrive\Documents\U of G\Winter 24\ENGG3380 Comp. Org. Des\Lab 2\hello-world.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed 0.4 inst/sec

Edit Execute

Text Segment

| Bkpt | Address | Code | Basic | Source |
|------|------------|------------|--------------------------|-----------------------------|
| | 0x00400000 | 0x24020004 | addiu \$2,\$0,0x00000004 | 13: li \$v0,4 # Code f... |
| | 0x00400004 | 0x3c011001 | lui \$1,0x00001001 | 14: la \$a0,msg # Pointe... |
| | 0x00400008 | 0x34240000 | ori \$4,\$1,0x00000000 | |
| | 0x0040000c | 0x0000000c | syscall | 15: syscall |
| | 0x00400010 | 0x2402000a | addiu \$2,\$0,0x0000000a | 16: li \$v0,10 # Code f... |
| | 0x00400014 | 0x0000000c | syscall | 17: syscall |

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x10010000 | 0x6c6c6548 | 0x6f57206f | 0x21646c72 | 0x27744920 | 0x72472073 | 0x2070756f | 0x203a3733 | 0x63697245 |
| 0x10010020 | 0x6c41202c | 0x202c6365 | 0x616b754c | 0x000a2173 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars Messages Run I/O

-- program is finished running --

Clear

Hello World! It's Group 37: Eric, Alec, Lukas!

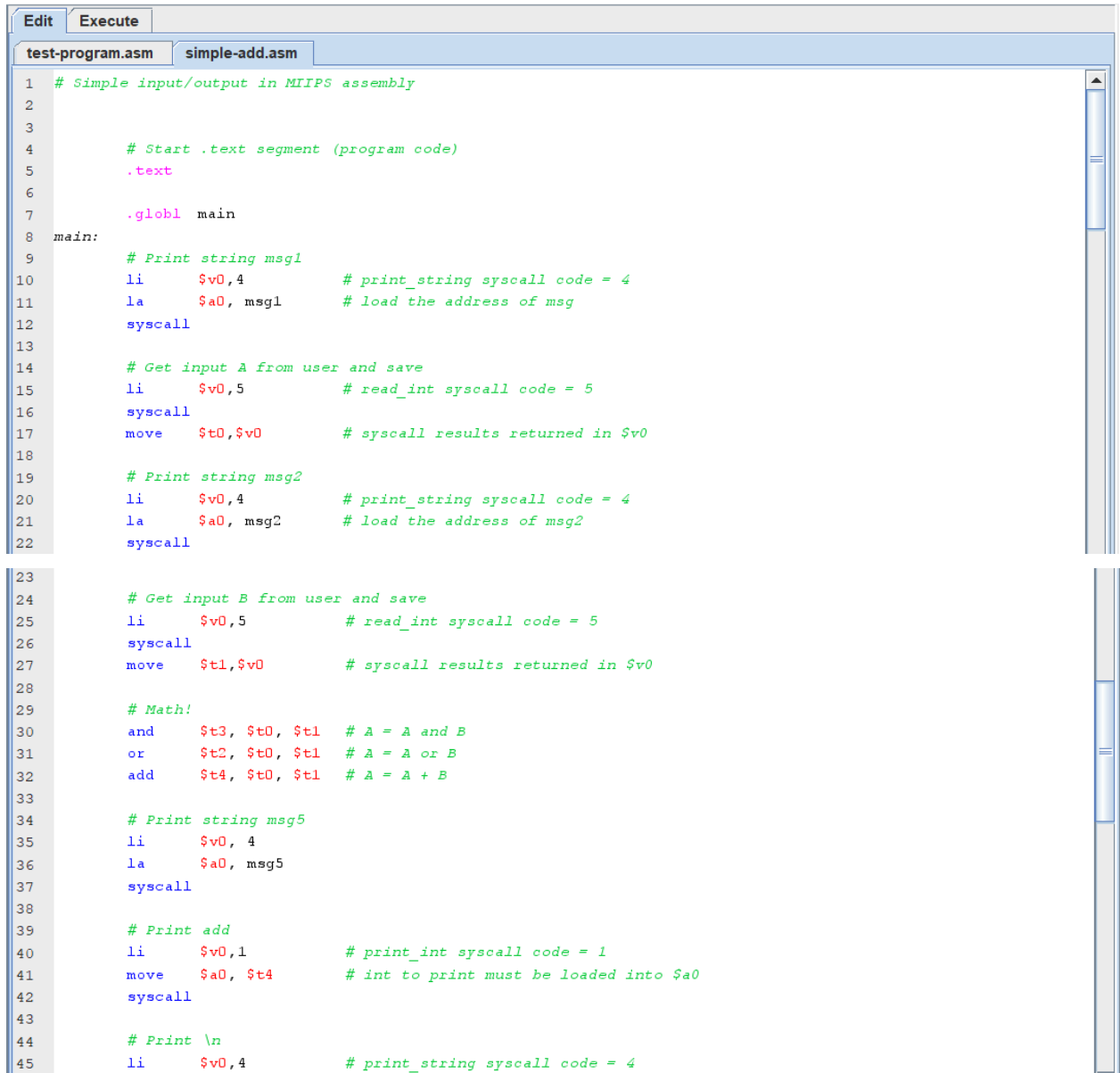
-- program is finished running --

Registers Coproc 1 Coproc 0

| Name | Number | Value |
|--------|--------|------------|
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x10010000 |
| \$v0 | 2 | 0x0000000a |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x10010000 |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0x00000000 |
| \$t1 | 9 | 0x00000000 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000000 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$s0 | 16 | 0x00000000 |
| \$s1 | 17 | 0x00000000 |
| \$s2 | 18 | 0x00000000 |
| \$s3 | 19 | 0x00000000 |
| \$s4 | 20 | 0x00000000 |
| \$s5 | 21 | 0x00000000 |
| \$s6 | 22 | 0x00000000 |
| \$s7 | 23 | 0x00000000 |
| \$s8 | 24 | 0x00000000 |
| \$s9 | 25 | 0x00000000 |
| \$k0 | 26 | 0x00000000 |
| \$k1 | 27 | 0x00000000 |
| \$gp | 28 | 0x10008000 |
| \$sp | 29 | 0x7fffffc |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00000000 |
| pc | | 0x00400018 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

3. Simple Add

a) Show a screen-shot for the two outputs (“or”, “and”)



```
1  # Simple input/output in MIPS assembly
2
3
4      # Start .text segment (program code)
5      .text
6
7      .globl main
8  main:
9      # Print string msg1
10     li    $v0,4          # print_string syscall code = 4
11     la    $a0, msg1      # load the address of msg
12     syscall
13
14     # Get input A from user and save
15     li    $v0,5          # read_int syscall code = 5
16     syscall
17     move  $t0,$v0        # syscall results returned in $v0
18
19     # Print string msg2
20     li    $v0,4          # print_string syscall code = 4
21     la    $a0, msg2      # load the address of msg2
22     syscall
23
24     # Get input B from user and save
25     li    $v0,5          # read_int syscall code = 5
26     syscall
27     move  $t1,$v0        # syscall results returned in $v0
28
29     # Math!
30     and   $t3, $t0, $t1  # A = A and B
31     or    $t2, $t0, $t1  # A = A or B
32     add   $t4, $t0, $t1  # A = A + B
33
34     # Print string msg5
35     li    $v0, 4
36     la    $a0, msg5
37     syscall
38
39     # Print add
40     li    $v0,1          # print_int syscall code = 1
41     move  $a0, $t4        # int to print must be loaded into $a0
42     syscall
43
44     # Print \n
45     li    $v0,4          # print_string syscall code = 4
```

```

46      la      $a0, newline
47      syscall
48
49      # Print string msg3
50      li      $v0, 4
51      la      $a0, msg3
52      syscall
53
54      # Print and
55      li      $v0, 1          # print_int syscall code = 1
56      move    $a0, $t3       # int to print must be loaded into $a0
57      syscall
58
59      # Print \n
60      li      $v0, 4          # print_string syscall code = 4
61      la      $a0, newline
62      syscall
63
64      # Print string msg4
65      li      $v0, 4
66      la      $a0, msg4
67      syscall
68
69
70      # Print or
71      li      $v0, 1          # print_int syscall code = 1
72      move    $a0, $t2       # int to print must be loaded into $a0
73      syscall
74
75      # Print \n
76      li      $v0, 4          # print_string syscall code = 4
77      la      $a0, newline
78      syscall
79
80      li      $v0, 10         # exit
81      syscall
82
83      # Start .data segment (data!)
84      .data
85      msg1: .ascii "Enter A:  "
86      msg2: .ascii "Enter B:  "
87      msg3: .ascii "A and B = "
88      msg4: .ascii "A or B = "
89      msg5: .ascii "A + B = "
90      newline: .ascii "\n"

```

Line: 36 Column: 14 ☒ Show Line Numbers

Mars Messages

Run I/O

Clear

```

Enter A:  6
Enter B:  4
A + B = 10
A and B = 4
A or B = 6

-- program is finished running --

```

Contributions

The group work was very evenly split up and each group member did approximately 33% of the work.

4. Exercise (i.e. Q12 from PS1)

The MIPS assembly source code can be found below in Appendix A.

When running the code with $a = 4$ and $b = 6$ the following output is produced

The screenshot displays the Mars MIPS simulator interface. The main window is divided into several panes:

- Text Segment:** Shows the assembly code for the program. The code includes initialization of registers $s0$ and $s1$ to 4 and 6 respectively, followed by nested loops to calculate the sum of elements in an array D .
- Data Segment:** Shows the memory layout. The array D is located at address $0x10010000$ and contains the values $3, 0, 0, 4, 0, 0, 0, 5, 0, 0, 0, 6, 0, 0, 0, 7, 0, 0, 0, 8$.
- Registers:** Shows the state of registers. The $s0$ register contains the value $0x00000004$ and the $s1$ register contains the value $0x00000006$.
- Console:** Displays the output of the program. It shows the array D and a confirmation that the program finished running.

The console output is as follows:

```
The array D is:
3 0 0 4 0 0 0 5 0 0 0 6 0 0 0 7 0 0 0 8
-- program is finished running --
```

Appendix A - Source Code for Exercise (i.e. Q12 from PS1)

```
.data
arrayD: .word 0 : 21
sizeD: .word 21
.text
main:
    addi $s0, $0, 4 #init a = 4
    addi $s1, $0, 6 #init b = 6

    addi $t0, $0, 0 #init i = 0
loop1:    slt $t5, $t0, $s0 #test for i < a
    beq $t5, $0, Done1 #if i >= a jump to Done
    addi $t1, $0, 0 #init j = 0
loop2:    slt $t4, $t1, $s1 #test for j < b
    beq $t4, $0, Done2 #if j >= b jump to Done

    #body of loop 2 here
    add $t2, $t0, $t1    #temp2 = i + j
    sll $t3, $t1, 4      #temp3 = 4*j, but shift it by 2 more addresses for another
*4 to account for byte addressable memory
    la $t6, arrayD      #load address of arrayD into temp6
    add $t3, $t3, $t6    #add index of temp3 to address of arrayD
    sw $t2, 0($t3)      #write out to D[4*j]

    addi $t1, $t1, 1 # j++
    j loop2
Done2:    addi $t0, $t0, 1 # i++
    j loop1
Done1:    # The array numbers are computed and stored in array. Print them.
    la $a0, arrayD      # first argument for print (array)
    la $a1, sizeD       # second argument for print (size)
    lw $a1, 0($a1)
    jal print           # call print routine.

    li $v0, 10          #prep syscall to return from the program
    add $a0, $0, $0     #put a 0 in the return argument
    syscall             #make the syscall

#####
# Subroutine to print the numbers on one line.
.data
space:.asciiz " "      # space to insert between numbers
head:.asciiz "The array D is:\n"
.text
print:add $t0, $zero, $a0 # starting address of array of data to be printed
    add $t1, $zero, $a1 # initialize loop counter to array size
    la $a0, head        # load address of the print heading string
    li $v0, 4           # specify Print String service
```



```

        syscall                # print the heading string

out:    lw    $a0, 0($t0)      # load the integer to be printed (the current Fib. number)
        li    $v0, 1          # specify Print Integer service
        syscall                # print fibonacci number

        la    $a0, space      # load address of spacer for syscall
        li    $v0, 4          # specify Print String service
        syscall                # print the spacer string

        addi $t0, $t0, 4      # increment address of data to be printed
        addi $t1, $t1, -1     # decrement loop counter
        bgtz $t1, out         # repeat while not finished

        jr    $ra             # return from subroutine
# End of subroutine to print the numbers on one line
#####

```