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

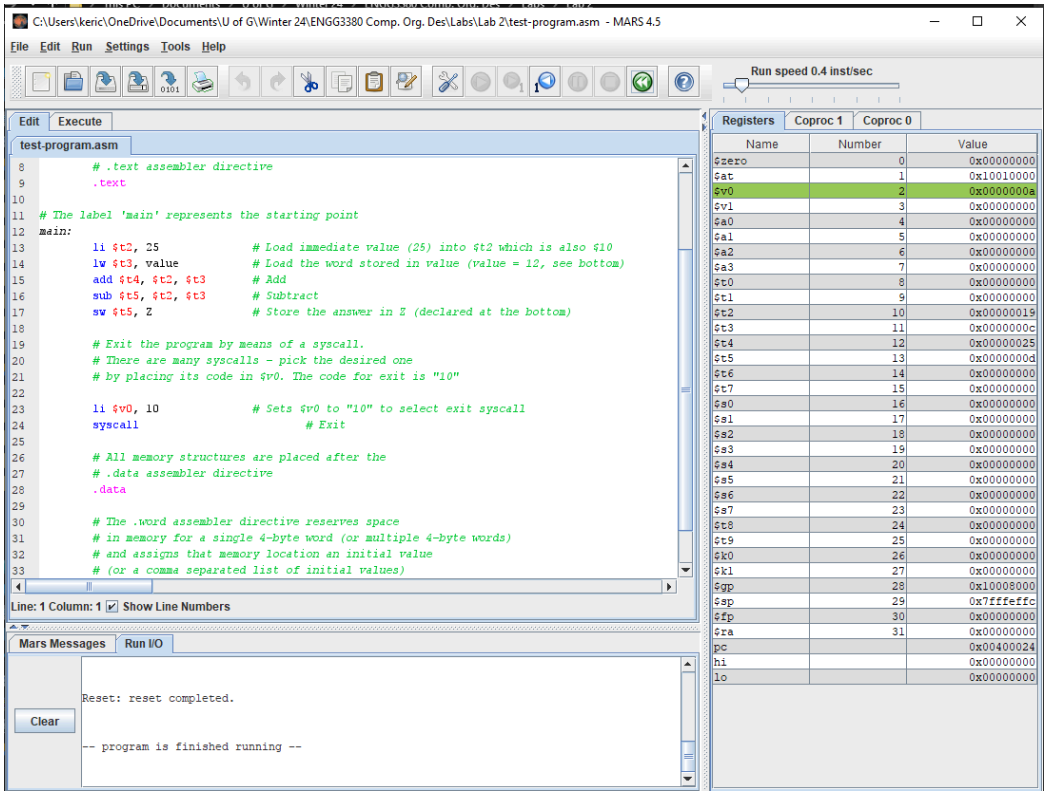
Lab # 2 - MIPS Simulators (MARS and QtSpim)

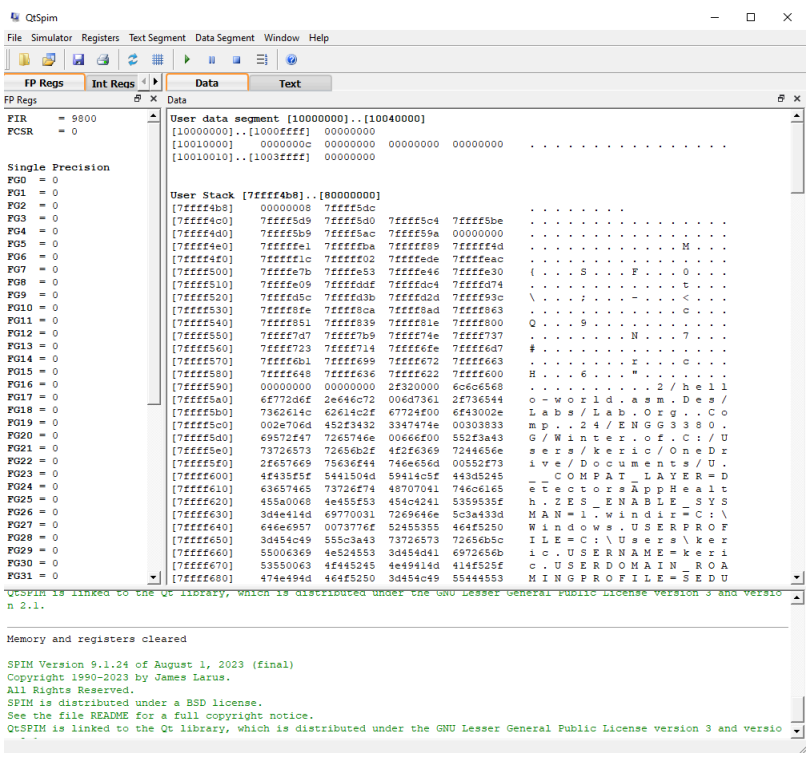
Lab Instructor: Haleh Vahedi

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, $at, $t2, $t3, $t4, $t5,

d) What addresses in Memory are changed?

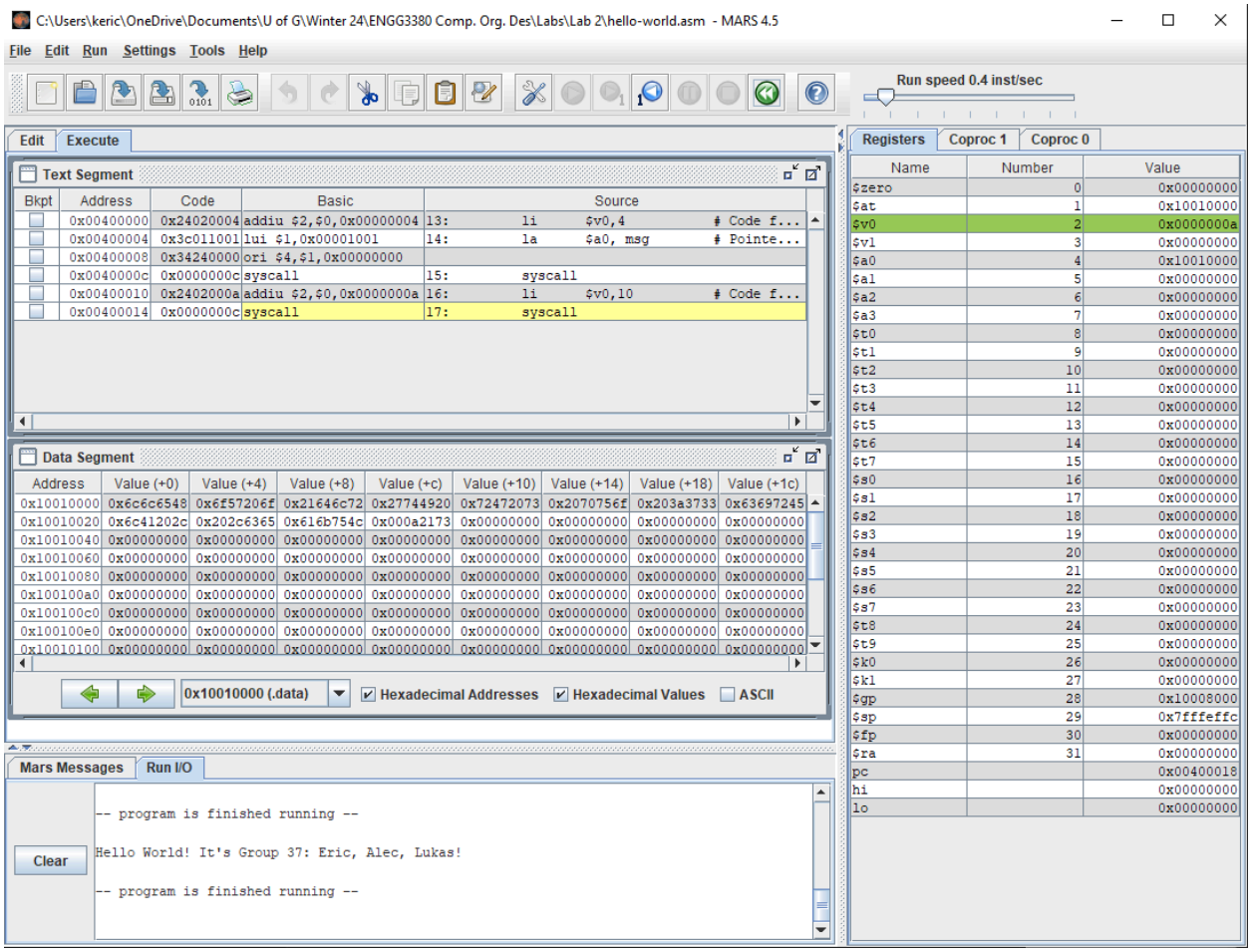
The address that is changed is: 0x10010004 also called Z. The memory address 0x10010004 received the result of the subtraction on line 16 of the assembly. This stores a word value of 13 into address 0x10010004.

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.



3. Simple Add

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

Edit | Execute

test-program.asm | simple-add.asm

```
# Simple input/output in MIPS assembly


        # Start .text segment (program code)
        .text

        .globl  main
main:
        # Print string msg1
        li      $v0,4           # print_string syscall code = 4
        la      $a0, msg1       # load the address of msg
        syscall

        # Get input A from user and save
        li      $v0,5           # read_int syscall code = 5
        syscall
        move    $t0,$v0         # syscall results returned in $v0

        # Print string msg2
        li      $v0,4           # print_string syscall code = 4
        la      $a0, msg2       # load the address of msg2
        syscall

        # Get input B from user and save
        li      $v0,5           # read_int syscall code = 5
        syscall
        move    $t1,$v0         # syscall results returned in $v0

        # Math!
        and     $t3, $t0, $t1   # A = A and B
        or      $t2, $t0, $t1   # A = A or B
        add     $t4, $t0, $t1   # A = A + B

        # Print string msg5
        li      $v0, 4
        la      $a0, msg5
        syscall

        # Print add
        li      $v0,1           # print_int syscall code = 1
        move    $a0, $t4        # int to print must be loaded into $a0
        syscall

        # Print \n
        li      $v0,4           # print_string syscall code = 4
```

```
        la      $a0, newline
        syscall

        # Print string msg3
        li      $v0, 4
        la      $a0, msg3
        syscall

        # Print and
        li      $v0,1           # print_int syscall code = 1
        move    $a0, $t3        # int to print must be loaded into $a0
        syscall

        # Print \n
        li      $v0,4           # print_string syscall code = 4
        la      $a0, newline
        syscall

        # Print string msg4
        li      $v0, 4
        la      $a0, msg4
        syscall

        # Print or
        li      $v0,1           # print_int syscall code = 1
        move    $a0, $t2        # int to print must be loaded into $a0
        syscall

        # Print \n
        li      $v0,4           # print_string syscall code = 4
        la      $a0, newline
        syscall

        li      $v0,10          # exit
        syscall

        # Start .data segment (data!)
        .data
msg1:   .asciiz "Enter A:    "
msg2:   .asciiz "Enter B:    "
msg3:   .asciiz "A and B = "
msg4:   .asciiz "A or B = "
msg5:   .asciiz "A + B = "
newline:    .asciiz     "\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 --
```

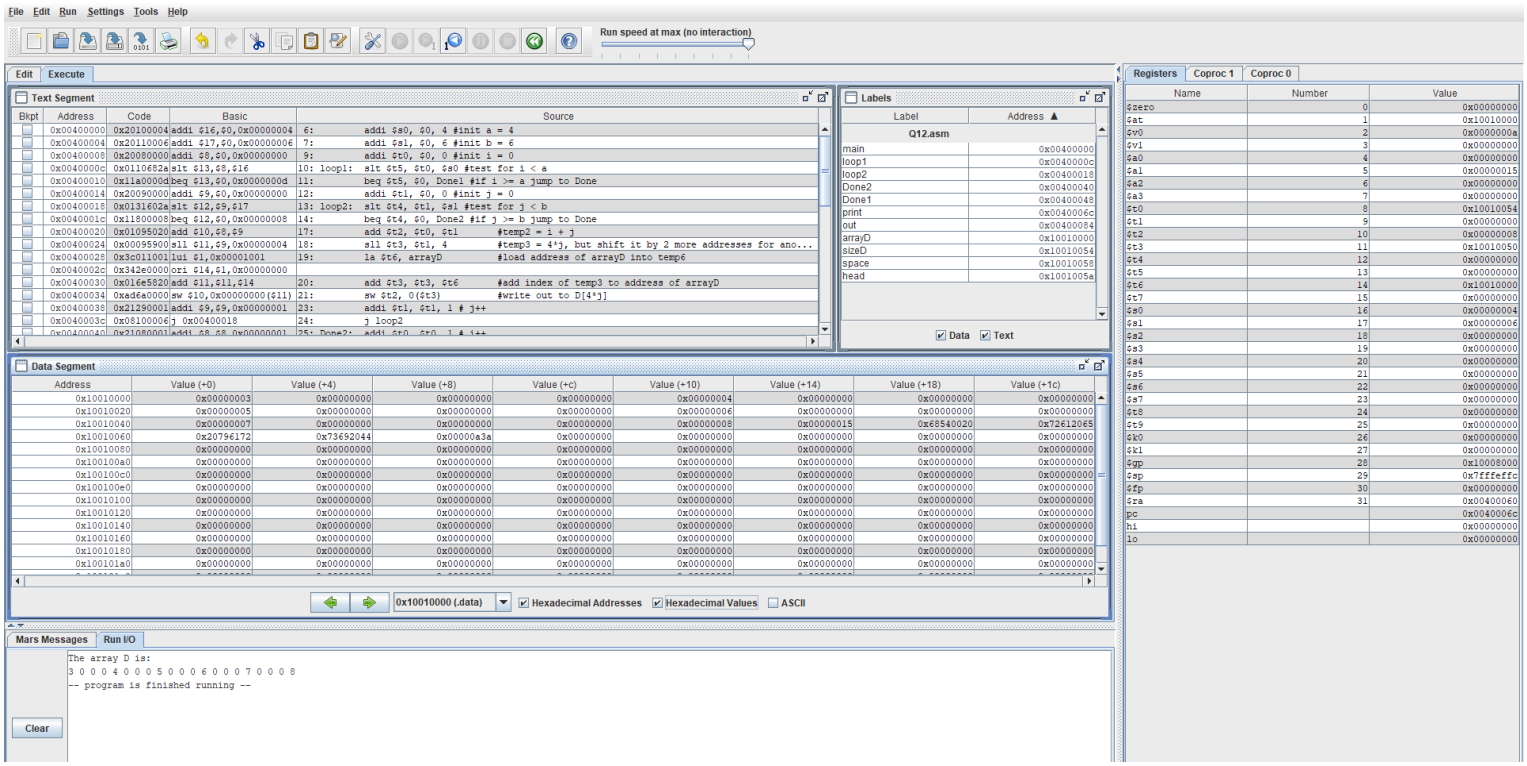
Contributions

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



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 syscal 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
##############################################################
```