Name: Scot Matson Student ID: 009602502

Class	CS47, Sec 01
Midterm	Fall 2015
Due Date	October 14, 2015 7:15 PM PST
Notes	1. Open book exam 2. Insert your name and student ID at header section 3. Insert your answer in this document and export it to PDF format. 4. Upload the PDF document in Canvas 'midterm' assignment. 5. If you are asked to write a program, you can write it on MARS, test it. Then copy paste the complete code into your answer here (the code must be able to be assembled and executed at examiner end). 6. Explanation of answer is required. 7. MUST GENERATE PDF FILE BY 7:15PM. The time stamp will be checked and anything beyond that point will not be accepted. 8. It is also suggested to email your copy of PDF to kaushik.patra

1. A genX number system uses symbol W, X, Y, Z with decimal weight 0, 1, 2, 3.
   a) what is decimal equivalent of ZZXWY (2pts)
   b) what is the genX equivalent of decimal number 315? (2pts)
   c) what is genX equivalent of binary number 1011010011001001? (1pts)

Ans:

a) Give the predefined decimal weight.
ZZXWY
33102

b) Since genX is a base-4 number system, we can convert 315 as follows.
315 / 4 = 78 R 3
78 / 4 = 19 R 2
19 / 4 = 4 R 3
4 / 4 = 1 R 0

In genX, (315)base10 translates to WZYZ

c) (1011 0100 1100 1001)base-2 represented as 4 8-bit segments can be represented (10 11 01 00 11 00 10 01)base-2 with 8 4-bit segments that translate easliy into our genX system.

00 = 0 = W, 01 = 1 = X, 10 = 2 = Y, 11 = 3 = Z.
Thus 10 11 01 00 11 00 10 01 = YZXWZWYX

Name: Scot Matson Student ID: 009602502

2. A procedure '**foo**' uses four argument and returns one value. Internally it uses $s0, $s1 and $s4. It also calls other procedure from inside.
   a) Write down the caller RTE saving code. (2pts)
   b) Write down the caller RTE restoring code. (2pts)
   c) Write instruction to return to caller.(1pts)

Ans:

Min stack frame = 24 bytes (1byte each: $fp, $sp + padding, $a0, $a1, $a2, $a3)
Additional registers: $s0, $s1, $s4 + req. Min = 36 bytes.

a)

```
addi    $sp, $sp, -36
sw      $fp, 36($sp)
sw      $a0, 32($sp)
sw      $a1, 28($sp)
sw      $a2, 24($sp)
sw      $a3, 20($sp)
sw      $s0, 16($sp)
sw      $s1, 12($sp)
sw      $s4,  8($sp)
addi    $fp, $sp, 36
```

b)

```
lw      $fp, 36($sp)
lw      $a0, 32($sp)
lw      $a1, 28($sp)
lw      $a2, 24($sp)
lw      $a3, 20($sp)
lw      $s0, 16($sp)
lw      $s1, 12($sp)
lw      $s4,  8($sp)
addi    $sp, $sp, 36
```

c)

```
jr      $ra
```

Name: Scot Matson Student ID: 009602502

3. Write the following program in MIPS assembly code. Assume we are handling +ve integers only. Provide complete code, without any .include, so that the program can run standalone.
   a) A 16-bit positive integer multiplication procedure that takes two arguments in $a0, $a1 and return the result in $v0. The multiplication procedure can only use addition (add) operation. (3pts)
   b) A 32-bit positive integer division procedure that takes two arguments in $a0, $a1 and return the quotient in $v0 and remainder in $v1. The division procedure can only use addition and subtraction operation. (4pts)
   c) Write a program that asks two positive integers and print result of multiplication and division using the implemented procedure in 1a and 1b. (3pts)

Ans:

a)

 # Macro: print_str

```
# Usage: print_str(<address of the string>)
.macro print_str($arg)
      li    $v0, 4
      la    $a0, $arg
      syscall
.end_macro

# Macro: read_int
# Usage: read_int(<reg>)
.macro read_int($arg)
      li    $v0, 5            #     System Call for read_int
      syscall                 #     Call OS to perform operation
      move  $arg, $v0   #     Move value into register
.end_macro

# Macro: print_reg_int
# Usage: print_reg_int(<reg>)
.macro print_reg_int($arg)
      li    $v0, 1
      move  $a0, $arg
      syscall
.end_macro

.data
      msg0:       .asciiz "Enter two integers to find the product:\n"
      msg1:       .asciiz "Product: "
      exitMsg:          .asciiz "Exiting program."


.text
.globl main
main:
      # Get user input.
      print_str(msg0)
      read_int($a0)           # Integer 1
      read_int($a1)           # Integer 2

      # Caller for multiply proc, handles arithmetic
      jal multiply

      print_str(msg1)
      # $v0 is not returning the correct product even when using
      # the mult opcode.
```

```
      print_reg_int($v0)      # The returned product

      # Exit the application
      j     exit

# Should use a loop to add 'X',  'Y' number of times.
multiply:

      # DEBUG: Show the input value
      # **These are accessible
      # print_reg_int($a0)
      # print_reg_int($a1)


      slt   $t0, $zero, $a1 # if second integer > 0
      add   $a0, $a0, $a1
      add

      # Even with Mult
      mflo  $v0         # Move the result into the return register.
      jr $ra                  # Return to exit the program.



exit:
      print_str(exitMsg)
      li    $v0, 10
      syscall
```