Name: Student ID:

Class	CS47, Sec 01
Midterm	Fall 2015
Due Date	October 15, 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 always good practice.

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) (256*3) + (64*3) + (16*1) + (4*0) + (1*2) = 978

b) 315 / 4 => Q = 78 , R = 3 (Z)
78 / 4 => Q = 19 , R = 2 (Y)
19 / 4 => Q = 4 , R = 3 (Z)
4 / 4 => Q = 1, R = 0 (W)
1 / 4 => Q = 0, R = 1 (X)

The genX equivalent is XWZYZ

c) Direct method by grouping 2 bits and map it to xGen.
10 11 01 00 11 00 10 01 (binary) = YZXWZWYX (genX)

Alternate method: Convert to decimal and then to genX

In decimal : 46281

46281 / 4 => Q = 11570 , R = 1 (X)
11570 / 4 => Q = 2892 , R = 2 (Y)
2892 / 4 => Q = 723 , R = 0 (W)
723 / 4 => Q = 180 , R = 3 (Z)
180 / 4 => Q = 45, R = 0 (W)
45 / 4 => Q = 11, R = 1 (X)
11 / 4 => Q = 2, R = 3 (Z)
2 / 4 => Q = 0, R = 2 (Y)

Hence the number in genX system in YZXWZWYX

Name: Student ID:

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:

a) Need to save total 9 registers ($fp, $ra, $a0-$a3, $s0, $s1, $s4) or 36 bytes. Therefore stack pointer must be moved (36 + 8 – 4) = 40 bytes. The following is the RTE saving code

[The sw sequence may differ]

```
addi    $sp, $sp, -40
sw      $fp, 40($sp)
sw      $ra, 36($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, 40
```

b) The RTE restore code would be as following

```
lw      $fp, 40($sp)
lw      $ra, 36($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, 40
```

c) The return to caller code would be 'jr $ra'

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:

Name: Student ID:

[Cut paste the answer in MARS and run it]

```
#<------------------ MACRO DEFINITIONS ---------------------->#
        # Macro : print_str
        # Usage: print_str(<address of the string>)
        .macro print_str($arg)
      li    $v0, 4     # System call code for print_str
      la    $a0, $arg   # Address of the string to print
      syscall            # Print the string
      .end_macro
      # Macro : exit
        # Usage: exit
        .macro exit
      li    $v0, 10
      syscall
      .end_macro
      # Macro: read_int
      # Usage: read_int(<reg>)
      .macro read_int($arg)
      li    $v0,5 # Read intger
      syscall
      move  $arg, $v0 # move the data to target reg
      .end_macro
      # Macro: print_reg_int
      # Usage: print_reg_int(<reg>)
      .macro print_reg_int ($arg)
      li    $v0, 1            # print_int call
      move  $a0, $arg   # move the source reg value to $a0
      syscall
      .end_macro
#<------------------ Data layout ---------------------------->#
.data
msg1: .asciiz "Enter a +ve integer ? "
msg2: .asciiz "Multiplication of them is : "
msg3: .asciiz "Division quotient of them is : "
msg4: .asciiz "Division remainder of them is : "
charCR: .asciiz "\n"
#<------------------ Code layout ---------------------------->#
.text
multiply:
      # store RTE $fp, $ra, $a0, $a1, $s0
      # => 5*4 + 8 - 4 = 24 bytes for frame size
      addi  $sp, $sp, -24
      sw    $fp, 24($sp)
      sw    $ra, 20($sp)
      sw    $a0, 16($sp)
      sw    $a1, 12($sp)
      sw    $s0,  8($sp)
      addi  $fp, $sp, 24
      # Body
      add   $s0, $zero, $zero
multLoop:
      beqz  $a1, multRet
      add   $s0, $s0, $a0
      addi  $a1, $a1, -1
      j     multLoop
multRet:
      add   $v0, $s0, $zero
      # restore RTE & return
      lw    $fp, 24($sp)
      lw    $ra, 20($sp)
```

Name: Student ID:

```
      lw    $a0, 16($sp)
      lw    $a1, 12($sp)
      lw    $s0,  8($sp)
      addi  $sp, $sp, 24

      jr $ra

divide:
      # Store RTE code $fp, $ra, $a0, $a1, $s0, $s1
      # => 6*4 + 8 - 4 = 28 bytes for frame size
      addi  $sp,$sp, -28
      sw    $fp,28($sp)
      sw    $ra,24($sp)
      sw    $a0,20($sp)
      sw    $a1,16($sp)
      sw    $s0,12($sp)
      sw    $s1, 8($sp)
      addi  $fp, $sp, 28
      # Body
      add   $s0, $zero, $zero       # Quotient
      add   $s1, $a1, $zero         # Remainder
divLoop:
      blt   $s1, $a0, divRet
      sub   $s1, $s1, $a0
      addi  $s0, $s0, 1
      j     divLoop
divRet:
      move  $v0, $s0
      move  $v1, $s1
      # Restore code and return
      lw    $fp,28($sp)
      lw    $ra,24($sp)
      lw    $a0,20($sp)
      lw    $a1,16($sp)
      lw    $s0,12($sp)
      lw    $s1, 8($sp)
      addi  $sp, $sp, 28
      jr $ra

.globl main
main:
      # ask the number
      print_str(msg1)
      read_int($a1)
      print_str(msg1)
      read_int($a0)

      # save the numbers in s0, s1
      move $s1, $a1
      move $s0, $a0

      # multiply and print result
      jal multiply
      move $t0, $v0
      print_str(msg2)
      print_reg_int($t0)
      print_str(charCR)

      # divide and print result
      move $a1, $s1
      move $a0, $s0
```

Name: Student ID:

```
jal divide
move $t0, $v0 # Quotient
move $t1, $v1 # Remainder
print_str(msg3)
print_reg_int($t0)
print_str(charCR)
print_str(msg4)
print_reg_int($t1)
print_str(charCR)

exit
```