Amrita Vishwa Vidyapeetham

Amrita School of Engineering, Coimbatore

Department of Computer Science and Engineering

Practice Question

Sub Code: 19CS211	**Sub Title: COA**
Roll No: CB.EN.U4CSE19453	**Name: R.ABHINAV**
Practice Lab	**Date: 26-04-2021**

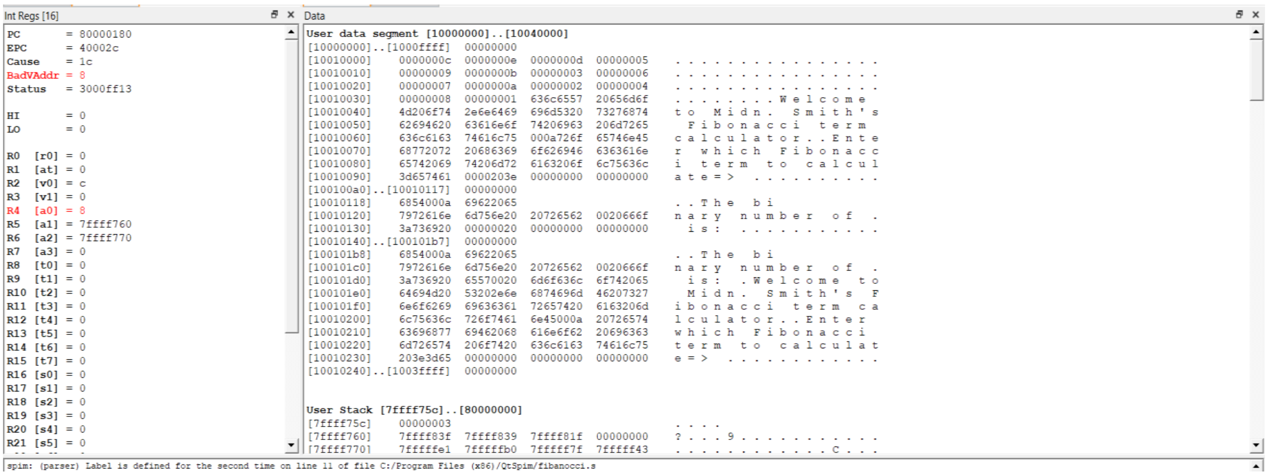
---

1) Write a program in MIPS that contains a procedure which takes one argument - an integer greater than or equal to zero which specifies which element of the Fibonacci sequence is to be returned.The procedure must be recursive and return the correct Fibonacci value. Your final Fibonacci value should be stored in the register $t1.

Code: (done factorial)

```
# Here is a recursive implementation of factorial, first in C, then in assembly:
# int factorial (int n){
#     if (n < 2) return 1;
#     return (n * factorial (n-1));  /* n! = n * (n-1)! */
# }


.text
.globl main

factorial:
   bgtz  $a0, doit
   li   $v0, 1      # base case, 0! = 1
   jr   $ra
doit:
   sub  $sp,8          # stack frame
   sw   $s0,0($sp)      # will use for argument n
   sw   $ra,4($sp)      #  return address

   move $s0, $a0       #  save argument
```

```
    sub  $a0, 1          #  n-1
    jal  factorial       #  v0 = (n-1)!
    mul  $v0,$s0,$v0     #     n*(n-1)!

    lw   $s0,($sp)       # restore registers from stack
    lw   $ra,4($sp)
    add  $sp,8
    jr   $ra
main:
    li   $a0, 7          # set the argument for the factorial function to 7
    sub  $sp, 4          # create the stack frame
    sw   $ra,0($sp)      # save the return address
    jal  factorial       # call factorial
    move $t1, $v0        # save the return value
    lw   $ra,0($sp)      # restore the orignal return address
    add  $sp,4
    jr   $ra
```



```
PC       = 80000180
EPC      = 40002c
Cause    = 1c
BadVAddr = 8
Status   = 3000ff13

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = c
R3  [v1] = 0
R4  [a0] = 8
R5  [a1] = 7ffff760
R6  [a2] = 7ffff770
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
```

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000c  0000000e  0000000d  00000005    ................
[10010010]    00000009  0000000b  00000003  00000006    ................
[10010020]    00000007  0000000a  00000002  00000004    ................
[10010030]    00000008  00000001  636c6557  20656d6f    ........Welcome
[10010040]    4d206f74  2e6e6469  696d5320  73276874    to Midn. Smith's
[10010050]    62694620  63616e6f  74206963  206d7265     Fibonacci term
[10010060]    636c6163  74616c75  000a726f  65746e45    calculator..Ente
[10010070]    68772072  20686369  6f626946  6363616e    r which Fibonacc
[10010080]    65742069  74206d72  6163206f  6c75636c    i term to calcul
[10010090]    3d657461  0000203e  00000000  00000000    ate=> ..........
[100100a0]..[10010117]  00000000
[10010118]    6854000a  69622065                        ..The bi
[10010120]    7972616e  6d756e20  20726562  0020666f    nary number of .
[10010130]    3a736920  00000020  00000000  00000000     is: ...........
[10010140]..[100101b7]  00000000
[100101b8]    6854000a  69622065                        ..The bi
[100101c0]    7972616e  6d756e20  20726562  0020666f    nary number of .
[100101d0]    3a736920  65570020  6d6f636c  6f742065     is: .Welcome to
[100101e0]    64694d20  53202e6e  6874696d  46207327     Midn. Smith's F
[100101f0]    6e6f6269  69636361  72657420  6163206d    ibonacci term ca
[10010200]    6c75636c  726f7461  6e45000a  20726574    lculator..Enter
[10010210]    63696877  69462068  616e6f62  20696363    which Fibonacci
[10010220]    6d726574  206f7420  636c6163  74616c75    term to calculat
[10010230]    203e3d65  00000000  00000000  00000000    e=> ............
[10010240]..[1003ffff]  00000000


User Stack [7ffff75c]..[80000000]
[7ffff75c]    00000003                                  ....
[7ffff760]    7ffff83f  7ffff839  7ffff81f  00000000    ?...9...........
[7ffff770]    7fffffe1  7fffffb0  7fffff7f  7fffff43    ............C...
```

```
spim: (parser) Label is defined for the second time on line 11 of file C:/Program Files (x86)/QtSpim/fibanocci.s
```

2) Write a MIPS program that given a number N and N integers can print the integers in a sorted order using Bubble Sort. Bubble Sort algorithm involves swapping of two numbers. Write a procedure for swapping two numbers separately and use it in the sort function.

Code :

```
 .text
     .globl main
 main:       la  $a0, Array
 loop:      lw  $t0, 0($a0)
              lw  $t1, 4($a0)
```

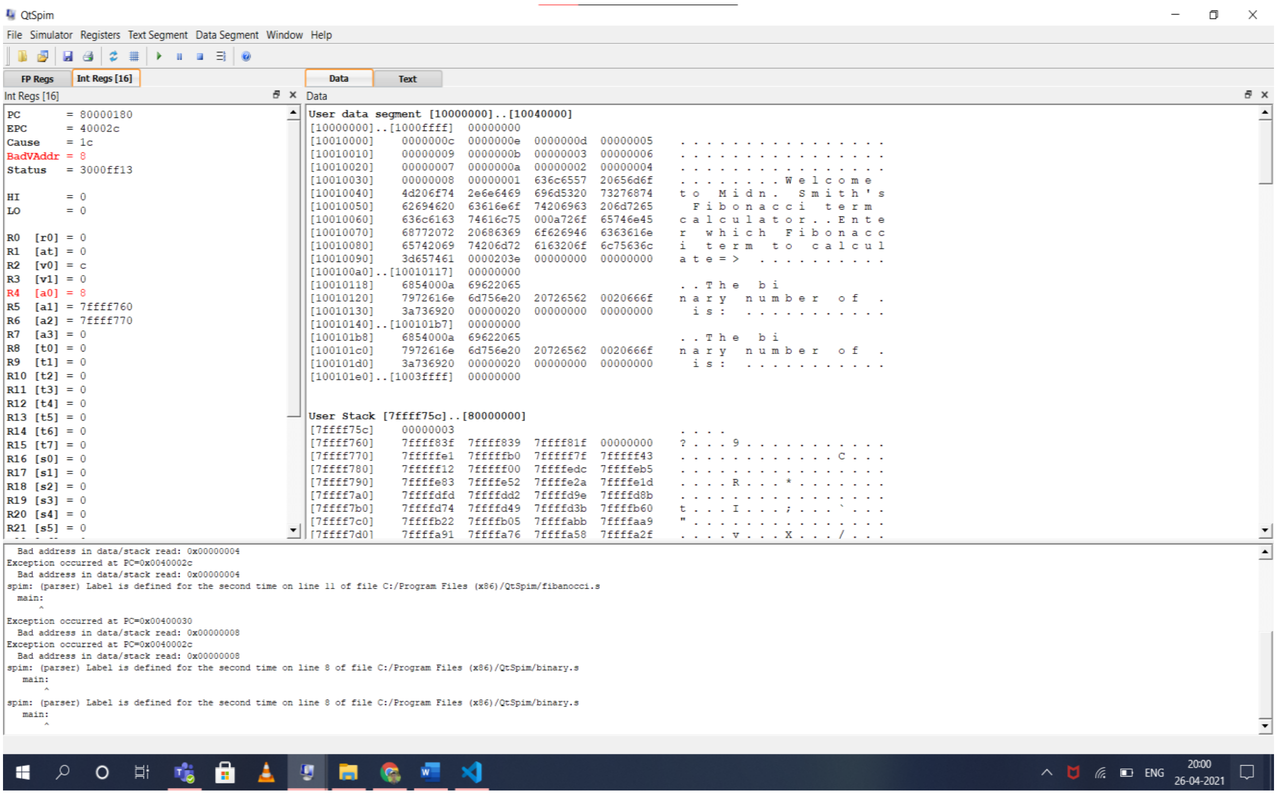
```
            blt $t1, $t0, swap
            addi    $a0, $a0, 4
            j   loop
   swap:        sw  $t0, 4($a0)
            sw  $t1, 0($a0)
            li  $a0, 0
            j   loop

            .data

   Array:       .word   14, 12, 13, 5, 9, 11, 3, 6, 7, 10, 2, 4, 8, 1
```

3) Write a MIPS program to convert a user given integer to a binary number. Consider both the positive and negative integers. In case of a negative integer, the output has to be in 2’s complement form. Print the binary number as a string.

Code:

```
.data
        myArray: .word 0:32
        newLine: .asciiz "\n"
        message: .asciiz "The binary number of "
        m_is: .asciiz " is: "
.text
        .globl main
        main:
        li $t1, 10 # Decimal
        li $t4, 4

        li $v0, 4
        la $a0, message
        syscall

        li $v0, 1
        add $a0, $t1, $zero
        syscall

        li $v0, 4
        la $a0, m_is
        syscall
```

```
# Index = $t0
li $t0, 0
li $t2, 2

while:
        blez $t1, sub_print

        divu $t1, $t2

        mfhi $a0
        mflo $t1

        sw $a0, myArray($t0)
        add $t0, $t0, $t4

        j while


sub_print:
        sub $t0, $t0, $t4

        j print_arr

print_arr:
        bltz $t0, exit

        lw $t6, myArray($t0)

        sub $t0, $t0, $t4

        li $v0, 1
        move $a0, $t6
        syscall

        j print_arr
exit:
        li $v0, 10
        syscall
```



```
PC       = 80000180
EPC      = 40002c
Cause    = 1c
BadVAddr = 8
Status   = 3000ff13

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = c
R3  [v1] = 0
R4  [a0] = 8
R5  [a1] = 7ffff760
R6  [a2] = 7ffff770
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
```

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000c  0000000e  0000000d  00000005    . . . . . . . . . . . . . . . .
[10010010]    00000009  0000000b  00000003  00000006    . . . . . . . . . . . . . . . .
[10010020]    00000007  0000000a  00000002  00000004    . . . . . . . . . . . . . . . .
[10010030]    00000008  00000001  636c6557  20656d6f    . . . . . . . . W e l c o m e
[10010040]    4d206f74  2e6e6469  696d5320  73276874    t o   M i d n .   S m i t h ' s
[10010050]    62694620  63616e6f  74206963  206d7265      F i b o n a c c i   t e r m
[10010060]    636c6163  74616c75  000a726f  65746e45    c a l c u l a t o r . . E n t e
[10010070]    68772072  20686369  6f626946  6363616e    r   w h i c h   F i b o n a c c
[10010080]    65742069  74206d72  6163206f  6c75636c    i   t e r m   t o   c a l c u l
[10010090]    3d657461  0000203e  00000000  00000000    a t e = >   . . . . . . . . . .
[100100a0]..[10010117]  00000000
[10010118]    6854000a  69622065                        . . T h e   b i
[10010120]    7972616e  6d756e20  20726562  0020666f    n a r y   n u m b e r   o f   .
[10010130]    3a736920  00000020  00000000  00000000      i s :   . . . . . . . . . . .
[10010140]..[100101b7]  00000000
[100101b8]    6854000a  69622065                        . . T h e   b i
[100101c0]    7972616e  6d756e20  20726562  0020666f    n a r y   n u m b e r   o f   .
[100101d0]    3a736920  00000020  00000000  00000000      i s :   . . . . . . . . . . .
[100101e0]..[1003ffff]  00000000


User Stack [7ffff75c]..[80000000]
[7ffff75c]    00000003                                  . . . .
[7ffff760]    7ffff83f  7ffff839  7ffff81f  00000000    ? . . . 9 . . . . . . . . . . .
[7ffff770]    7fffffe1  7fffffb0  7fffff7f  7fffff43    . . . . . . . . . . . . C . . .
[7ffff780]    7fffff12  7fffff00  7ffffedc  7ffffeb5    . . . . . . . . . . . . . . . .
[7ffff790]    7ffffe83  7ffffe52  7ffffe2a  7ffffe1d    . . . . R . . . * . . . . . . .
[7ffff7a0]    7ffffdfd  7ffffdd2  7ffffd9e  7ffffd8b    . . . . . . . . . . . . . . . .
[7ffff7b0]    7ffffd74  7ffffd49  7ffffd3b  7ffffb60    t . . . I . . . ; . . . ` . . .
[7ffff7c0]    7ffffb22  7ffffb05  7ffffabb  7ffffaa9    " . . . . . . . . . . . . . . .
[7ffff7d0]    7ffffa91  7ffffa76  7ffffa58  7ffffa2f    . . . . v . . . X . . . / . . .
```

```
  Bad address in data/stack read: 0x00000004
Exception occurred at PC=0x0040002c
  Bad address in data/stack read: 0x00000004
spim: (parser) Label is defined for the second time on line 11 of file C:/Program Files (x86)/QtSpim/fibanocci.s
  main:
      ^

Exception occurred at PC=0x00400030
  Bad address in data/stack read: 0x00000008
Exception occurred at PC=0x0040002c
  Bad address in data/stack read: 0x00000008
spim: (parser) Label is defined for the second time on line 8 of file C:/Program Files (x86)/QtSpim/binary.s
   main:
       ^

spim: (parser) Label is defined for the second time on line 8 of file C:/Program Files (x86)/QtSpim/binary.s
   main:
       ^
```

Practice problems :

1.

```
# A demonstration of some simple MIPS instructions
# used to test QtSPIM

        # Declare main as a global function
        .globl main

        # All program code is placed after the
        # .text assembler directive
        .text

# The label 'main' represents the starting point
main:
        li $t2, 25              # Load immediate value (25)
        lw $t3, value           # Load the word stored in value (see bottom)
        add $t4, $t2, $t3       # Add
        sub $t5, $t2, $t3       # Subtract
        sw $t5, Z               #Store the answer in Z (declared at the bottom)

        # Exit the program by means of a syscall.
        # There are many syscalls - pick the desired one
        # by placing its code in $v0. The code for exit is "10"
        li $v0, 10 # Sets $v0 to "10" to select exit syscall
        syscall # Exit

        # All memory structures are placed after the
        # .data assembler directive
        .data

        # The .word assembler directive reserves space
        # in memory for a single 4-byte word (or multiple 4-byte words)
        # and assigns that memory location an initial value
        # (or a comma separated list of initial values)
    value:  .word 12
Z:      .word 0
```