**Q3**

*# Compute several Fibonacci numbers and put in array, then print*

*.data*

fibs:*.word*   0 : 19         *# "array" of words to contain fib values*

size: *.word*  19             *# size of "array" (agrees with array declaration)*

prompt: *.asciiz* "How many Fibonacci numbers to generate? (2 <= x <= 19)"

*.text*

      la   $s0, fibs        *# load address of array*

      la   $s5, size        *# load address of size variable*

      lw   $s5, 0($s5)      *# load array size*

*# Optional: user inputs the number of Fibonacci numbers to generate*

pr:   la   $a0, prompt      *# load address of prompt for syscall*

      li   $v0, 4           *# specify Print String service*

      syscall               *# print the prompt string*

      li   $v0, 5           *# specify Read Integer service*

      syscall               *# Read the number. After this instruction, the number read is in $v0.*

      bgt  $v0, $s5, pr     *# Check boundary on user input -- if invalid, restart*

      blt  $v0, $zero, pr   *# Check boundary on user input -- if invalid, restart*

      add  $s5, $v0, $zero  *# transfer the number to the desired register*

      li   $s2, 1           *# 1 is the known value of first and second Fib. number*

      sw   $s2, 0($s0)      *# F[0] = 1*

      sw   $s2, 4($s0)      *# F[1] = F[0] = 1*

      addi $s1, $s5, -2     *# Counter for loop, will execute (size-2) times*

*# Loop to compute each Fibonacci number using the previous two Fib. numbers.*

loop: lw   $s3, 0($s0)      *# Get value from array F[n-2]*

      lw   $s4, 4($s0)      *# Get value from array F[n-1]*

      add  $s2, $s3, $s4    *# F[n] = F[n-1] + F[n-2]*

      sw   $s2, 8($s0)      *# Store newly computed F[n] in array*

      addi $s0, $s0, 4      *# increment address to now-known Fib. number storage*

      addi $s1, $s1, -1     *# decrement loop counter*

      bgtz $s1, loop        *# repeat while not finished*

*# Fibonacci numbers are computed and stored in array. Print them.*

      la   $a0, fibs        *# first argument for print (array)*

      add  $a1, $zero, $s5  *# second argument for print (size)*

      jal  print            *# call print routine.*

*# The program is finished. Exit.*

      li   $v0, 10          *# system call for exit*

      syscall               *# Exit!*

*###############################################################*

*# Subroutine to print the numbers on one line.*

*.data*

space:*.asciiz*  " "          *# space to insert between numbers*

head: *.asciiz*  "The Fibonacci numbers are:\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*

*###############################################################*

**Q4**

*# Sample Data Memory Initialization*

*# Lab 6.4*

*# Data Memory Selection*

*.data*

value1: *.word* 12

value2: *.word* -45

value3: *.word* 00

*.align* 2

*# Program Memory Selection*

*.text*

main:

    lw  $t0, value1

    lw  $t1, value2

    add $t2, $t0, $t1

    bgt $t2, $zero, skip

    sub $t2, $zero, $t2

skip:   sw  $t2, value3

    li $v0, 10  *# Exit*

    syscall

**Q5**

*# Sample Data Memory Initialization*

*# Lab 6.5*

*# Data Memory Section*

*.data*

value1: *.word* 5

value2: *.word* 89

*.align* 2

*# Program Memory Section*

*.text*

main:

    lw $t0, value1

    lw $t1, value2

    sub $s0, $t0, $t1

    li $v0, 10

    syscall

**Q6**

*# Simple User Input Program*

*# Lab 6.7*

*# Data Memory Section*

*.data*

pr: *.asciiz* "Enter Number 1: "

pr1:    *.asciiz* "Enter Number 2: "

ans:    *.asciiz* "Answer: "

ad: *.asciiz* " + "

eq: *.asciiz* " = "

*# Program Memory Section*

*.text*

main:

    la $a0, pr *# Print out pr*

    li $v0, 4

    syscall

    li $v0, 5 *# Take input and put into $t0*

    syscall

    move $t0, $v0

    la $a0, pr1 *# Print out pr1*

    li $v0, 4

    syscall

    li $v0, 5 *#Take input and put into $t1*

    syscall

    move $t1, $v0

    la $a0, ans *# Print out "Answer: "*

    li $v0, 4

    syscall

    la $a0, ($t0) *# Print first integer*

    li $v0, 1

    syscall

    la $a0, ad *# Print out ad*

    li $v0, 4

    syscall

    la $a0, ($t1) *# Print second integer*

    li $v0, 1

    syscall

    la $a0, eq *# Print out eq*

    li $v0, 4

    syscall

    add $a0, $t0, $t1 *#Add integers and print out*

    li $v0, 1

    syscall

    li $a0, 10 *# Print newline (10 in ASCII is newline)*

    li $v0, 11

    syscall

    li $a0, 10 *# Print newline (10 in ASCII is newline)*

    li $v0, 11

    syscall

    j main

    li $v0, 10 *# Exit*

    syscall