 **Northwestern Polytechnic University**

**EE488 - Computer Architecture**

**Homework Assignment #5**

**Due day: 11/24/2021**

**Instruction:**

1. **Push the answer sheet to GitHub in word file**
2. **Overdue homework submission could not be accepted.**
3. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**
4. Implement a subprogram that prompt the user for *3* numbers, finds the median (middle value) of the *3*, and returns that value to the calling program.

# Program: Aditivaidyautils.asm

# Purpose: To define utilities which will be used in MIPS programs.

# Author: Aditi Vaidya

# Subprograms Index:

# Exit -Call syscall with a server 10 to exit the program

# NewLine -Print a new line character (\n) to the console

# PrintInt - Print a string with an integer to the console

# PromptInt - Prompt the user to enter an integer, and return it to the calling program.

# Subprogram: PrintNewLine

# Author: Aditi Vaidya

# Purpose: To output a new line to the user console

# Input: None

# Output: None

# Side Effects: A new line character is printed to the user's console

.text

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.data

\_\_PNL\_newline: .asciiz "\n"

# subprogram: PrintInt

# author: Aditi Vaidya

# purpose: To print a string to the console

# input: $a0 - The address of the string to print.

# $a1 - The value of the int to print

# returns: None

# side effects: The String is printed followed by the integer value.

.text

PrintInt:

# Print string. The string address is already in $a0

li $v0, 4

syscall

# Print integer. The integer value is in $a1, and must

move $a0, $a1

li $v0, 1

syscall

jr $ra

# subprogram: PromptInt

# author: Aditi Vaidya

# purpose: To print the user for an integer input, and

# to return that input value to the caller.

# input: $a0 - The address of the string to print.

# returns: $v0 - The value the user entered

# side effects: The String is printed followed by the integer value.

.text

PromptInt:

# Print the prompt, which is already in $a0

addi $v0, $zero, 4

syscall

# Read the integer value. Note that at the end of the

# syscall the value is already in $v0, so there is no

move $a0, $a1

addi $v0, $zero, 5

syscall

jr $ra

# Subprogram: Exit

# Author: Aditi Vaidya

# Purpose: To exit the program

# Input: None

# Output: None

# Side Effects: Exits the MIPS program

.text

Exit:

li $v0, 10

syscall

# Subprogram: PrintString

# Author: Aditi Vaidya

# Purpose: To print a string to the console

# Input: $a0 - The address of the string to be printed

# Returns: None

# Side Effects: The string is printed to the console

.text

PrintString:

addi $v0, $zero, 4

syscall

jr $ra

***#Subprogram for finding median***

.data

prompt1: .asciiz "Please enter first number to find the median= "

prompt2: .asciiz "Please enter second number to find the median= "

prompt3: .asciiz "Please enter third number to find the median= "

result: .asciiz "Median = "

.text

main:

la $a0, prompt1

jal PromptInt

move $s0, $v0

jal PrintNewLine

la $a0, prompt2

jal PromptInt

move $s1, $v0

jal PrintNewLine

la $a0, prompt3

jal PromptInt

move $s2, $v0

jal PrintNewLine

jal Median

move $s0, $v0

# Print out the median value returned

la $a0, result

la $a1, ($s0)

jal PrintInt

jal PrintNewLine

jal Exit

Median:

la $v0, ($s1)

jr $ra

.include "Aditivaidyautils.asm"

1. Implement a recursive program that takes in a number and finds the square of that number through addition. For example if the number *3* is entered, you would add *3+3+3=9*. If *4* is entered you would add *4+4+4+4=16*. This program must be implemented using recursion to add the numbers together.

# Program: Aditivaidyautils.asm

# Purpose: To define utilities which will be used in MIPS programs.

# Author: Aditi Vaidya

# Subprograms Index:

# Exit -Call syscall with a server 10 to exit the program

# NewLine -Print a new line character (\n) to the console

# PrintInt - Print a string with an integer to the console

# PromptInt - Prompt the user to enter an integer, and return it to the calling program.

# Subprogram: PrintNewLine

# Author: Aditi Vaidya

# Purpose: To output a new line to the user console

# Input: None

# Output: None

# Side Effects: A new line character is printed to the user's console

.text

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.data

\_\_PNL\_newline: .asciiz "\n"

# subprogram: PrintInt

# author: Aditi Vaidya

# purpose: To print a string to the console

# input: $a0 - The address of the string to print.

# $a1 - The value of the int to print

# returns: None

# side effects: The String is printed followed by the integer value.

.text

PrintInt:

# Print string. The string address is already in $a0

li $v0, 4

syscall

# Print integer. The integer value is in $a1, and must

move $a0, $a1

li $v0, 1

syscall

jr $ra

# subprogram: PromptInt

# author: Aditi Vaidya

# purpose: To print the user for an integer input, and

# to return that input value to the caller.

# input: $a0 - The address of the string to print.

# returns: $v0 - The value the user entered

# side effects: The String is printed followed by the integer value.

.text

PromptInt:

# Print the prompt, which is already in $a0

addi $v0, $zero, 4

syscall

# Read the integer value. Note that at the end of the

# syscall the value is already in $v0, so there is no

move $a0, $a1

addi $v0, $zero, 5

syscall

jr $ra

# Subprogram: Exit

# Author: Aditi Vaidya

# Purpose: To exit the program

# Input: None

# Output: None

# Side Effects: Exits the MIPS program

.text

Exit:

li $v0, 10

syscall

# Subprogram: PrintString

# Author: Aditi Vaidya

# Purpose: To print a string to the console

# Input: $a0 - The address of the string to be printed

# Returns: None

# Side Effects: The string is printed to the console

.text

PrintString:

addi $v0, $zero, 4

syscall

jr $ra

***#Subprogram for finding square by adding***

.data

prompt: .asciiz "Enter an integer to square = "

present: .asciiz "Square = "

.text

main:

# Read in the input from the user

la $a0, prompt

jal PromptInt

# Send in the input to the recursive function

move $a0, $v0

move $a1, $v0

jal RecurSquare

move $s0, $v0

# Print out the returned value

la $a0, present

la $a1, ($s0)

jal PrintInt

jal PrintNewLine

jal Exit

RecurSquare:

# If the value inputted by the user is the stopping value (-1)

beq $a0, 1, return

addi $a0, $a0, -1

addi $sp, $sp, -4

sw $ra, 0($sp)

jal RecurSquare

return:

add $v0, $v0, $a1

lw $ra, 0($sp)

addi $sp, $sp, 4

jr $ra

.include "Aditivaidyautils.asm"

1. Write a recursive program to calculate factorial numbers. Use the definition of factorial as *F(n) = n \* F(n-1)*

#Use Aditivaidyautils.asm , same as above .

***#subprogram to find recursive factorial.***

.data

prompt:.asciiz "Enter a number to find factorial = "

present:.asciiz "Factorial = "

.text

main:

# Read in the number

la $a0, prompt

jal PromptInt

move $s0, $v0

# Calculate the factorial

la $a0, ($s0)

jal Fact

la $a0, present

la $a1, ($v0)

jal PrintInt

jal PrintNewLine

jal Exit

Fact:

addi $sp, $sp, -8

sw $ra, 0($sp)

sw $a0, 4($sp)

addi $a0, $a0, -1

beqz $a0, helper

jal Fact

b return

helper: # set base case to avoid multiplying $v0 and $a0

lw $v0, 4($sp)

lw $ra, 0($sp)

addi $sp, $sp, 8

jr $ra

return:

lw $a0, 4($sp)

mult $v0, $a0

mflo $v0

lw $ra, 0($sp)

addi $sp, $sp, 8

jr $ra

.include "Aditivaidyautils.asm"

1. The following pseudo code converts an input value of a single decimal number from

*1 ≤ n ≥15* into a single hexadecimal digit. Translate this pseudo code into MIPS assembly.

*main{*

*String a[16]*

*a[0] = "0x0"*

*a[1] = "0x1"*

*a[2] = "0x2"*

*a[3] = "0x3"*

*a[4] = "0x4"*

*a[5] = "0x5"*

*a[6] = "0x6"*

*a[7] = "0x7"*

*a[8] = "0x8"*

*a[9] = "0x9"*

*a[10] = "0xa"*

*a[11] = "0xb"*

*a[12] = "0xc"*

*a[13] = "0xd"*

*a[14] = "0xe"*

*a[15] = "0xf"*

*int i = prompt("Enter a number from 0 to 15 ")*

*print("your number is " + a[i])*

*}*

#Use Aditivaidyautils.asm , same as above .

***#Subprogram to convert decimal numbers upto 15 to hexadecimal number.***

.data

prompt:.asciiz "Enter a number from 0 to 15 = "

present: .asciiz "your number is = "

error : .asciiz "\nEntered number is either less than 0 or greater than 15"

hexnum:

.asciiz "0x0"

.asciiz "0x1"

.asciiz "0x2"

.asciiz "0x3"

.asciiz "0x4"

.asciiz "0x5"

.asciiz "0x6"

.asciiz "0x7"

.asciiz "0x8"

.asciiz "0x9"

.asciiz "0xa"

.asciiz "0xb"

.asciiz "0xc"

.asciiz "0xd"

.asciiz "0xe"

.asciiz "0xf"

.text

main:

la $a0, prompt #TODO

jal PromptInt

move $s0, $v0

blt $s0,0,wrong

bge $s0,16,wrong

# Multiply the input by 4 and print out the correct hex

mul $s0, $s0, 4

la $a0, hexnum

add $a0, $a0, $s0

jal PrintString

jal Exit

wrong:

li $v0,4

la $a0,error

syscall

jal Exit

.include "Aditivaidyautils.asm"

1. The following pseudo code program calculates the Fibonacci numbers from *1…n*, and stores them in an array. Translate this pseudo code into MIPS assembly, and use the PrintIntArray subprogram to print the results.

*main{*

*int size = PromptInt(“Enter a max Fibonacci number to calc: “)*

*int Fibonacci[size]*

*Fibonacci[0] = 0*

*Fibonacci[1] = 1*

*for (int i = 2; i < size; i++){*

*Fibonacci[i] = Fibonacci[i-1] + Fibonacci[i-2]*

*}*

*PrintIntArray(Fibonacci, size)*

*}*

# Program: AVarrayfibautils.asm

# Purpose: To define utilities which will be used in MIPS programs.

# Author: Aditi Vaidya

# Subprograms Index:

# Exit -Call syscall with a server 10 to exit the program

# NewLine -Print a new line character (\n) to the console

# PrintInt - Print a string with an integer to the console

# PromptInt - Prompt the user to enter an integer, and return it to the calling program.

# PrintIntArray-print an array of integers to console

# Subprogram: AllocateArray

# Purpose: To allocate an array of $a0 items, each of size $a1.

# Author: Aditi Vaidya

# Input: $a0 - the number of items in the array

# $a1 - the size of each item

# Output:$v0 - Address of the array allocated

AllocateArray:

addi $sp, $sp, -4

sw $ra, 0($sp)

mul $a0, $a0, 4 # Question 3b ( I do not understand what is mean't in question 3a )

li $v0, 9

syscall

lw $ra, 0($sp)

addi $sp, $sp, 4

jr $ra

# Subprogram: PrintNewLine

# Author: Aditi Vaidya

# Purpose: To output a new line to the user console

# Input: None

# Output: None

# Side Effects: A new line character is printed to the user's console

.text

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.data

\_\_PNL\_newline: .asciiz "\n"

# subprogram: PrintInt

# author: Aditi Vaidya

# purpose: To print a string to the console

# input: $a0 - The address of the string to print.

# $a1 - The value of the int to print

# returns: None

# side effects: The String is printed followed by the integer value.

.text

PrintInt:

# Print string. The string address is already in $a0

li $v0, 4

syscall

# Print integer. The integer value is in $a1, and must

move $a0, $a1

li $v0, 1

syscall

jr $ra

# subprogram: PromptInt

# author: Aditi Vaidya

# purpose: To print the user for an integer input, and

# to return that input value to the caller.

# input: $a0 - The address of the string to print.

# returns: $v0 - The value the user entered

# side effects: The String is printed followed by the integer value.

.text

PromptInt:

# Print the prompt, which is already in $a0

addi $v0, $zero, 4

syscall

# Read the integer value. Note that at the end of the

# syscall the value is already in $v0, so there is no

move $a0, $a1

addi $v0, $zero, 5

syscall

jr $ra

# Subprogram: Exit

# Author: Aditi Vaidya

# Purpose: To exit the program

# Input: None

# Output: None

# Side Effects: Exits the MIPS program

.text

Exit:

li $v0, 10

syscall

# Subprogram: PrintString

# Author: Aditi Vaidya

# Purpose: To print a string to the console

# Input: $a0 - The address of the string to be printed

# Returns: None

# Side Effects: The string is printed to the console

.text

PrintString:

addi $v0, $zero, 4

syscall

jr $ra

.text

PrintIntArray:

addi $sp, $sp, -16

sw $ra, 0($sp)

sw $s0, 4($sp)

sw $s1, 8($sp)

sw $s2, 12($sp)

# save array to $s0

move $s0, $a0

# initialization for counter loop

# $s2 is the loop counter

move $s1, $a1

move $s2, $zero

la $a0 open\_bracket

jal PrintString

loop:

sge $t0, $s2, $s1 # check condition

bnez $t0, end\_loop

sll $t0, $s2, 2

add $t0, $t0, $s0

lw $a1, 0($t0)

la $a0, separator

jal PrintInt

addi $s2, $s2, 1

b loop

end\_loop:

li $v0, 4

la $a0, close\_bracket

syscall

lw $ra, 0($sp)

lw $s0, 4($sp)

lw $s1, 8($sp)

lw $s2, 12($sp)

addi $sp, $sp, 16

jr $ra

.data

open\_bracket: .asciiz "["

close\_bracket: .asciiz "]"

separator: .asciiz ","

# Purpose: To prompt for a string, allocate the string

# and return the string to the calling subprogram. $a0 - The prompt

# Input: $a1 - The maximum size of the string

# Output:$v0 - The address of the user entered string

PromptString:

addi $sp, $sp, -12 # Push stack

sw $ra, 0($sp)

sw $a1, 4($sp)

sw $s0, 8($sp)

li $v0, 4

syscall

li $v0, 9

lw $a0, 4($sp)

syscall

move $s0, $v0

move $a0, $v0

li $v0, 8

lw $a1, 4($sp)

syscall

move $v0, $a0

lw $ra, 0($sp) # Pop stack

lw $s0, 8($sp)

addi $sp, $sp, 12

jr $ra

***#Subprogram for calculating the Fibonacci numbers and storing in array***

.data

prompt:.asciiz "Enter a max Fibonacci number to calc: "

sqbracketo:.asciiz "["

sqbracketc:.asciiz "]"

.text

main:

# Read in the number

la $a0, prompt

jal PromptInt

move $s0, $v0

bgt $s0, 1, continue

la $a0,sqbracketo

la $a1, 0

jal PrintInt

blt $s0, 1, short

la $a0, separator

la $a1, 1

jal PrintInt

la $a0, sqbracketc

jal PrintString

jal Exit

short:

la $a0,sqbracketc

jal PrintString

jal Exit

# Calculate the fibonacci numbers up to and store them in an array

# $s0 - size of array

# $s1 - base of the array

# $t0 - Tracking how many bytes from the base

# $t1 - First lagging number, $t2 - second lagging number

continue:

mul $a0, $s0, 4

addi $v0, $zero, 9

syscall

move $s1, $v0

sw $t1, 0($s1)

addi $t1, $t1, 1

sw $t1, 4($s1)

addi $t0, $t0, 8

# Fibonacci loop

Fibloop:

beq $t3, $s0, printingtime

subi $t1, $t0, 4

subi $t2, $t0, 8

add $t1, $s1, $t1

add $t2, $s1, $t2

lw $t1, ($t1)

lw $t2, ($t2)

add $t4, $t1, $t2

add $t5, $t0, $s1

sw $t4, ($t5)

addi $t0, $t0, 4

addi $t3, $t3, 1

b Fibloop

printingtime:

la $a0, ($s1)

la $a1, ($s0)

jal PrintIntArray

jal Exit

.include "AVarrayfibautils.asm"