**COE 301/ ICS 233, Term 172**

**Computer Architecture & Assembly Language**

**Programming Assignment# 2**

1. Names and IDs

* Yousef Khalid Majeed 201568070
* Bader Abed Almazmumi 201480600

1. Assignment Number: 2
2. Problem statement:

Write a MIPS assembly language program that implements the following:

1. A procedure, PrintA, that prints the content of an array of integers in a two-dimensional format (row-wise) leaving a space between elements. Assume that the procedure receives as parameters the address of the array in register $a0, the number of rows in register $a1, and the number of columns in register $a2.

# A procedure that receives in $a0 the address of an array,

# in $a1 the number of rows, and in $a2 the number of columns

# and displays the array.

printA:

move $t0, $a0

numOfRaws:

move $t1, $a2

numOfCulmns:

lb $a0, 0($t0)

li $v0, 1

syscall

li $v0, 11

la $a0, ' '

syscall

addi $t0, $t0, 1

addi $t1, $t1, -1

bnez $t1, numOfCulmns

li $v0, 4 # system call code for printing string = 4

la $a0, newLine # load address of string to be printed into $a0

syscall

addi $a1, $a1, -1

bnez $a1, numOfRaws

jr $ra

1. A procedure, RSum, that computes the sum of a given row. Assume that the procedure receives as parameters the address of the array in register $a0, the number of columns in register $a1, and the index of the row to be summed in register $a2.The procedure should return the sum of the row in register $v0.

# A procedure that receives in $a0 the address of an array, in $a1 the number of

# columns, and in $a2 the index of the row to be summed and that computes the

# sum of a given row.

RSum:

# Compute starting address of the row index

# $a1 --> the number of cols

# $a2 --> the row to be summed

# $t0 --> the offset

mul $t0, $a1, $a2

add $t0, $t0, $a0

li $v0, 0

RSumHelper:

lb $t1, 0($t0)

addu $v0, $v0, $t1

addi $t0, $t0, 1

addi $a1, $a1, -1

bnez $a1, RSumHelper

jr $ra

1. A procedure, CSum, that computes the sum of a given column. Assume that the procedure receives as parameters the address of the array in register $a0, the number of rows in register $a1, the number of columns in register $a2, and the index of the column to be summed in register $a3. The procedure should return the sum of the column in register $v0.

# A procedure that receives in $a0 the address of an array, in $a1 the number of

# rows, and in $a2 the number of columns, and in $a3 the index of the column

# to be summed, and that computes the sum of a given column.

CSum:

# Compute starting address of the column index

# $a1 --> the number of rows

# $a2 --> the number of columns

# $a3 --> column to be summed

# $t0 --> the offset

add $t0, $a0, $a3

li $v0, 0

CSumHelper:

lb $t1, 0($t0)

addu $v0, $v0, $t1

add $t0, $t0, $a2

addi $a1, $a1, -1

bnez $a1, CSumHelper

jr $ra

1. A procedure, ArrayRowSum, that displays the sums of all rows in the array based on using RSum procedure. Assume that the procedure receives as parameters the address of the array in register $a0, the number of rows in register $a1, and the number of columns in register $a2.

# A procedure that receives in $a0 the address of an array, in $a1 the number of

# rows, and in $a2 the number of columns, and that displays the sums of all rows

# in the array based on using RSum procedure.

ArrayRowSum:

# $a0 --> address # $a1 --> numOfR # $a2 --> numOfC # $t3 --> return address

move $t3, $ra

move $t2, $a1

move $a1, $a2

li $a2, 0

subu $sp $sp $t2

ArrayRowSumHelper:

# $a0 --> address # $a1 --> numOfC # $a2 --> counter # $t2 --> numOfR

jal RSum

sb $v0 0($sp)

addiu $sp $sp 1

addi $a2, $a2, 1

move $a1,$s1

bne $a2,$t2, ArrayRowSumHelper

la $a0, msg9

li $v0, 4

syscall

xor $a2, $a2, $a2

subu $sp, $sp, $t2

loopRSums:

la $a0, msg10

li $v0, 4

syscall

move $a0, $a2

li $v0, 1

syscall

la $a0, ':'

li $v0, 11

syscall

lb $a0, 0($sp)

li $v0, 1

syscall

addi $a2, $a2, 1

addiu $sp, $sp, 1

li $v0, 4

la $a0, newLine

syscall

bne $a2, $t2, loopRSums

move $ra, $t3

jr $ra

1. A procedure, ArrayColSum, that displays the sums of all columns in the array based on using CSum procedure. Assume that the procedure receives as parameters the address of the array in register $a0, the number of rows in register $a1, and the number of columns in register $a2.

# A procedure that receives in $a0 the address of an array, in $a1 the number of

# rows, and in $a2 the number of columns, and that displays the sums of all rows

# in the array based on using RSum procedure.

ArrayColSum:

# $a0 --> address # $a1 --> numOfR # $a2 --> numOfC # $t3 --> return address move $t3, $ra

li $a3, 0

ArrayColSumHelper:

# $a0 --> address # $a1 --> numOfR # $a2 --> numOfC # $a3 --> counter

jal CSum

sb $v0 ,0($sp)

addiu $sp $sp 1

addi $a3, $a3, 1

move $a1,$s0

bne $a3,$a2 ArrayColSumHelper

la $a0, msg11

li $v0, 4

syscall

xor $a3, $a3, $a3

subu $sp $sp $a2

loopCSums:

la $a0, msg12

li $v0, 4

syscall

move $a0, $a3

li $v0, 1

syscall

la $a0, ':'

li $v0, 11

syscall

lb $a0, 0($sp)

li $v0, 1

syscall

addi $a3, $a3, 1

addiu $sp, $sp, 1

li $v0, 4

la $a0, newLine

syscall

bne $a3, $a2, loopCSums

move $ra, $t3

jr $ra

1. Ask the user to enter number rows R and number of columns, C, and read it.

#Ask the user to enter number of rows

la $a0, msg1

li $v0, 4

syscall

li $v0, 5

syscall

move $s0, $v0 # Number of rows

#Ask the user to enter number of columns

la $a0, msg2

li $v0, 4

syscall

li $v0, 5

syscall

move $s1, $v0 # Number of columns

1. Ask the user to enter an RxC matrix of integers and read it.

#Ask the user to enter RxC matrix of integers

la $a0, msg3a

li $v0, 4

syscall

move $a0, $s0

li $v0, 1

syscall

li $a0,'x'

li $v0, 11

syscall

move $a0, $s1

li $v0, 1

syscall

la $a0, msg3b

li $v0, 4

syscall

#reading RxC matrix of integers into an array

la $s2, Array

mul $t1, $s0, $s1 #number of elements in the array

move $t2, $s2 #array address

li $t0, 0 #array counter

readLoop:

li $v0, 5

syscall

move $t3, $v0

sb $t3, 0($t2)

addiu $t2, $t2, 1 #increment the address

addiu $t0, $t0, 1 #increment the counter

blt $t0, $t1, readLoop # if counter < number of elements

1. Print a menu from which the user can select one of the following options:

1. Print the Entered Array

2. Print Sum of a Row

3. Print Sum of a Column

4. Print Rows Sum

5. Print Columns Sum

6. Exit the program

1. The solution along with the code:

# $s0 --> NumOfRows

# $s1 --> numOfCols

# $s2 --> array adress

################# Data segment #####################

.data

msg1: .asciiz "Enter number of rows: "

msg2: .asciiz "Enter number of columns: "

msg3a: .asciiz "Enter an array of "

msg3b: .asciiz " integers:\n"

menuText: "\nSelect one of the following functions:\n1. Print the Entered Array.\n2. Print Sum of a Row.\n3. Print Sum of a Column.\n4. Print Rows Sum.\n5. Print Columns Sum.\n6. Exit the program.\n"

cases: .word case0, case1, case2, case3, case4, case5

msg4a: .asciiz "Array of "

msg4b: .asciiz " integers is:\n"

msg5: .asciiz "Enter a row number: "

msg6a: .asciiz "Sum of row number "

msg7: .asciiz "Enter a column number: "

msg8a: .asciiz "Sum of column number "

msg6b\_8b: .asciiz " is: "

msg9: .asciiz "Array rows sum are:\n"

msg10: .asciiz "Sum of row "

msg11: .asciiz "Array columns sum are:\n"

msg12: .asciiz "Sum of column "

msg13: .asciiz "Invalid Choice... Please re-select a valid one.\n"

msg14: .asciiz "Row Dose Not Exist... Please re-enter a valid one.\n"

msg15: .asciiz "Column Dose Not Exist... Please re-enter a valid one.\n"

newLine: .asciiz "\n"

Array: .space 400 # it is assumed that we have a max. array size of 20x20

################# Code segment #####################

.text

.globl main

main: # main program entry

#Ask the user to enter a number of rows

la $a0, msg1

li $v0, 4

syscall

li $v0, 5

syscall

move $s0, $v0 # Number of rows

#Ask the user to enter a number of columns

la $a0, msg2

li $v0, 4

syscall

li $v0, 5

syscall

move $s1, $v0 # Number of columns

#Ask the user to enter RxC matrix of integers

#Printing messgae for reading the array

la $a0, msg3a

li $v0, 4

syscall

move $a0, $s0

li $v0, 1

syscall

li $a0,'x'

li $v0, 11

syscall

move $a0, $s1

li $v0, 1

syscall

la $a0, msg3b

li $v0, 4

syscall

#reading RxC matrix of integers into an array

la $s2, Array

mul $t1, $s0, $s1 #number of elements in the array

move $t2, $s2 #array address

li $t0, 0 #array counter

readLoop:

li $v0, 5

syscall

move $t3, $v0

sb $t3, 0($t2)

addiu $t2, $t2, 1 #increment the address

addiu $t0, $t0, 1 #increment the counter

blt $t0, $t1, readLoop # if counter < number of elements

doWhile:

#print the menu text

la $a0, menuText

li $v0, 4

syscall

li $v0, 5

syscall

subiu $t0, $v0, 1

bltz $t0, error # if $t0 < 0, go to default

li $t1, 5 # maximum number of cases (0,1)

bgt $t0, $t1, error # if $t0 > $t1, go to default

la $t2, cases # Load the address of cases array to $t2

sll $t0, $t0, 2 #($t0\*4) to calculate the offset of the chosen case in the array

addu $t2, $t2, $t0 # add the offset to the address of cases array

lw $t3, 0($t2) # load the chosen case’s address to $t3

jr $t3 # jump to the chosen case’s address

#Print the Entered Array:

case0:

la $a0, msg4a

li $v0, 4

syscall

move $a0, $s0

li $v0, 1

syscall

li $a0,'x'

li $v0, 11

syscall

move $a0, $s1

li $v0, 1

syscall

la $a0, msg4b

li $v0, 4

syscall

move $a0, $s2 # $a0 = address of array

move $a1, $s0 # $a2 = numOfRows

move $a2, $s1 # $a2 = numberOfColumn

jal printA

j doWhile

#Print Sum of a Row :

case1:

# Getting index Row

la $a0, msg5

li $v0, 4

syscall

li $v0, 5

syscall

move $a2, $v0 # row to be summed

bge $a2, $s0, wrongR

bltz $a2, wrongR

move $a0, $s2 # $a0 = address of array

move $a1, $s1 # $a1 = numberOfColumn

jal RSum

move $s3, $v0

la $a0, msg6a

li $v0, 4

syscall

move $a0, $a2

li $v0, 1

syscall

la $a0, msg6b\_8b

li $v0, 4

syscall

move $a0, $s3

li $v0, 1

syscall

li $v0, 4

la $a0, newLine

syscall

j doWhile

#Print Sum of a Column:

case2:

# Getting index Column

la $a0, msg7

li $v0, 4

syscall

li $v0, 5

syscall

move $a3, $v0 # column to be summed

bge $a3, $s1, wrongC

bltz $a3, wrongC

move $a0, $s2 # $a0 = address of array

move $a1, $s0 # $a1 = numberOfRows

move $a2, $s1 # $a2 = numberOfColumns

jal CSum

move $s4, $v0

la $a0, msg8a

li $v0, 4

syscall

move $a0, $a3

li $v0, 1

syscall

la $a0, msg6b\_8b

li $v0, 4

syscall

move $a0, $s4

li $v0, 1

syscall

li $v0, 4

la $a0, newLine

syscall

j doWhile

#Print Rows Sum:

case3:

move $a0, $s2 # $a0 = address of array

move $a1, $s0 # $a1 = numberOfRows

move $a2, $s1 # $a2 = numberOfColumns

jal ArrayRowSum

j doWhile

#Print Columns Sum:

case4:

move $a0, $s2 # $a0 = address of array

move $a1, $s0 # $a1 = numberOfRows

move $a2, $s1 # $a2 = numberOfColumns

jal ArrayColSum

j doWhile

# Close the program

case5:

li $v0, 10

syscall

wrongR:

li $v0, 4

la $a0, msg14

syscall

j case1

wrongC:

li $v0, 4

la $a0, msg15

syscall

j case2

error:

li $v0, 4

la $a0, msg13

syscall

j doWhile

###############################################################

# A procedure that receives in $a0 the address of an array,

# in $a1 the number of rows, and in $a2 the number of columns

# and displays the array.

printA:

move $t0, $a0

numOfRaws:

move $t1, $a2

numOfCulmns:

lb $a0, 0($t0)

li $v0, 1

syscall

li $v0, 11

la $a0, ' '

syscall

addi $t0, $t0, 1

addi $t1, $t1, -1

bnez $t1, numOfCulmns

li $v0, 4 # system call code for printing string = 4

la $a0, newLine # load address of string to be printed into $a0

syscall

addi $a1, $a1, -1

bnez $a1, numOfRaws

jr $ra

###############################################################

# A procedure that receives in $a0 the address of an array,

# in $a1 the number of columns, and in $a2 the index of the row to be summed

# and that computes the sum of a given row..

RSum:

# Compute starting address of the row index

# $a1 --> the number of cols

# $a2 --> the row to be summed

# $t0 --> the offset

mul $t0, $a1, $a2

add $t0, $t0, $a0

li $v0, 0

RSumHelper:

lb $t1, 0($t0)

addu $v0, $v0, $t1

addi $t0, $t0, 1

addi $a1, $a1, -1

bnez $a1, RSumHelper

jr $ra

###############################################################

# A procedure that receives in $a0 the address of an array,

# in $a1 the number of rows, and in $a2 the number of columns,

# and in $a3 the index of the column to be summed,

# and that computes the sum of a given coulmn..

CSum:

# Compute starting address of the column index

# $a1 --> the number of rows

# $a2 --> the number of columns

# $a3 --> column to be summed

# $t0 --> the offset

add $t0, $a0, $a3

li $v0, 0

CSumHelper:

lb $t1, 0($t0)

addu $v0, $v0, $t1

add $t0, $t0, $a2

addi $a1, $a1, -1

bnez $a1, CSumHelper

jr $ra

###############################################################

# A procedure that receives in $a0 the address of an array,

# in $a1 the number of rows, and in $a2 the number of columns,

# and that displays the sums of all rows in the array

# based on using RSum procedure.

ArrayRowSum:

# $a0 --> address

# $a1 --> number of rows

# $a2 --> number of colums

# $t3 --> return address ($ra)

move $t3, $ra

move $t2, $a1

move $a1, $a2

li $a2, 0

subu $sp $sp $t2

ArrayRowSumHelper:

# $a0 --> address

# $a1 --> number of colums

# $a2 --> counter

# $t2 --> number of rows

jal RSum

sb $v0 0($sp)

addiu $sp $sp 1

addi $a2, $a2, 1

move $a1,$s1

bne $a2,$t2 ArrayRowSumHelper

la $a0, msg9

li $v0, 4

syscall

xor $a2, $a2, $a2

subu $sp $sp $t2

loopRSums:

la $a0, msg10

li $v0, 4

syscall

move $a0, $a2

li $v0, 1

syscall

la $a0, ':'

li $v0, 11

syscall

la $a0, ' '

li $v0, 11

syscall

lb $a0, 0($sp)

li $v0, 1

syscall

addi $a2, $a2, 1

addiu $sp, $sp, 1

li $v0, 4

la $a0, newLine

syscall

bne $a2, $t2, loopRSums

move $ra, $t3

jr $ra

###############################################################

# A procedure that receives in $a0 the address of an array,

# in $a1 the number of rows, and in $a2 the number of columns,

# and that displays the sums of all rows in the array

# based on using RSum procedure.

ArrayColSum:

# $a0 --> address

# $a1 --> number of rows

# $a2 --> number of colums

# $t3 --> return address ($ra)

move $t3, $ra

li $a3, 0

ArrayColSumHelper:

# $a0 --> address

# $a1 --> the number of rows

# $a2 --> the number of columns

# $a3 --> counter

jal CSum

sb $v0 ,0($sp)

addiu $sp $sp 1

addi $a3, $a3, 1

move $a1,$s0

bne $a3,$a2 ArrayColSumHelper

la $a0, msg11

li $v0, 4

syscall

xor $a3, $a3, $a3

subu $sp $sp $a2

loopCSums:

la $a0, msg12

li $v0, 4

syscall

move $a0, $a3

li $v0, 1

syscall

la $a0, ':'

li $v0, 11

syscall

la $a0, ' '

li $v0, 11

syscall

lb $a0, 0($sp)

li $v0, 1

syscall

addi $a3, $a3, 1

addiu $sp, $sp, 1

li $v0, 4

la $a0, newLine

syscall

bne $a3, $a2, loopCSums

move $ra, $t3

jr $ra

###############################################################

1. Discussion:

This program read an array with RxC elements, and ask the user to choose one of the next 5 functions (with validation):

1) Print the Entered Array

2) Print Sum of a Row

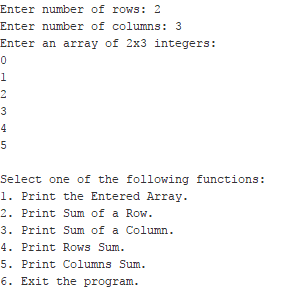
3) Print Sum of a Column

4) Print Rows Sum

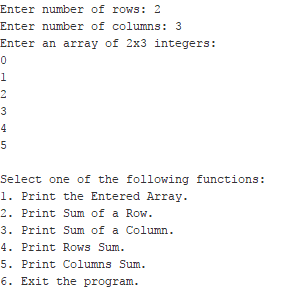
5) Print Columns Sum

Some functions, such as functions 4 and 5, use other functions inside them, like functions 2 and 3. (Snapshot of each attached below).

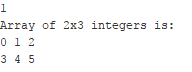
Read An Array:



Main Menu:



Print the Entered Array:



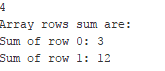
Print Sum of a Row:



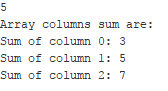
Print Sum of a Column:



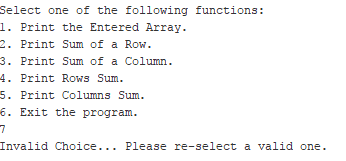
Print Rows Sum:

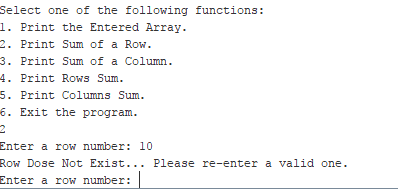


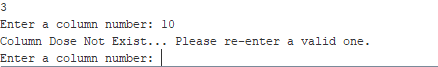
Print Columns Sum:



Input Validation:







The difficulty we faced is how to call a function from another function, but we thought for three hours and half and we solved it correctly (we think!) without any problems using the stack segment to store the values in and load it back if we want it again. It was very interesting moment after we know how to solve it. Eventually we learned a lot from this awesome homework and it was neither a hard nor an easy actually in between (we think also!).