Computer Architecture Lab

Experiment 2

Control Flow,I/O and Making Decisions.

Lab Instructor: Christos M.

Date Performed: 02/22/17

Date Submitted: 02/08/17

Name: Leonardo Roman lr534

Purpose:

The purpose of this lab is to learn how to use flow control(branch, jump instructions), inputs and outputs to the processor and how to make decisions in assembly language

Assignment 1:

In this assignment, we'll create a program that reads only non-negative integer and then calculates the sum of all values less than or equal to that same integer.

```
#This program reads a non-negative integer and calculates
#The sum of all values less than or equal to the entered integer
.data
message1: .asciiz "Enter a non-negative number: "
message2: .asciiz "The sum of all numbers less than or equal to "
message3: .asciiz " is "
message4: .asciiz "Please enter a positive number.\n"
li $s0, 0                #i = 0

.text
.globl main

main:
li $v0, 4                #to print a message
la $a0, message1         #Printing message 1
syscall

li $v0, 5                #to read an integer
syscall
move $t0, $v0            #storing the entered integer in $t0

bgt $t0,$0,l1            #if number entered is not negative go to l1
li $v0, 4                #to print a message
la $a0, message4         #get the address of message 4 to print message
syscall
j main                   #go back to main if any negative number was enter

l1:                      #lable 1 function
blt $s0,$t0,SUM          #if i <= num go to SUM
j END                    #else terminate

SUM:                     #Sum function
addi $s0,$s0,1           #i++
add $t1,$t1, $s0         #sum += i
j l1                     #go back to l1

END:                     #End function
li $v0, 4                #to print a message
la $a0, message2         #get address of message 2 to print message
syscall
move $a0, $t0            #assigned entered number as an argument to be printed
li $v0, 1                #to print an integer($a0) entered number
syscall
li $v0, 4                #to print a message
la $a0, message3         #get address of message 3 to print message
syscall
move $a0, $t1            #store value un $t1 into $a0 as argument to be printed
li $v0, 1                #printing value in $a0(final sum)
syscall
li $v0, 10               #finish program
syscall
```

Output

```
Console
Enter a non-negative number: 100
The sum of all numbers less than or equal to 100 is 5050
```

Assignment 2:

This program asks the user to enter a number and finds the closest prime number of number

Pseudocode

1. Prompt user to enter number
2. if number < 0 and number > 100 Error
3. for(i = 0; i < number; i++)
4. for(j = number; j > 0 ; j--)
5. if j%i == 0 then j is not prime
6. else Prime = j halt

```
#Leonardo Roman LR534
#This program asks the user to enter a number(n) and
#finds the closest prime number to number (n)
.data
message:  .asciiz "\nEnter a number from 0-100 or 00 to end program: "
message1: .asciiz "Number must be from 0-100.\n"
message2: .asciiz "The closes prime number to number "
message3: .asciiz " is: "

.text
.globl main

main:
li $s5,00
li $s6,1
addi $s0,$0,100
li $v0,4                    #to display a message
la $a0,message              #to display message 1
syscall                     #execute message 1
li $v0,5                    #to read an integer
syscall                     #execute and ask for number
move $s1,$v0                #move entered number by user into $t0
blt $s1,$0,OuttaRange       #if num<0 end
bgt $s1,$s0,OuttaRange      #if num>100 go back to main
beq $s1,$s5,end             #if number = 00 end program
beq $s1,$s6,PrimeOne        #if number = 00 end program
add $t0,$0,$s1              #$t0 = num
addi $t1,$0,2               #j = 2

Loop:
beq $t0,$t1,PrimeNumber     #if j == num then j is the prime number go to PrimeNUmber
div $t0,$t1                 #num/j
mfhi $s2                    #get the rimainder from hi and store it into $s2
beq $s2,$0,NumMinusMinus    #if $s2!=0, num is not a prime number go to NumMinusMinus
addi $t1,$t1,1              #j++;
j Loop                      #go back to Loop

NumMinusMinus:              #NumMinusMinus gets called when num/j!=0
addi $t0,$t0,-1             #num = num - 1;
addi $t1,$0,2               #reset j = 2
j Loop                      #go back to primeFunction


OuttaRange:                 #OuttaRange gets called if num<0 or num>100
li $v0,4
la $a0,message1             #Prints message "Number must be from 0-100."
syscall
j main                      #go back to main to re-enter number
```

```
PrimeNumber:                #PrimeNumber gets called when prime num gets found
li $v0,4
la $a0,message2             #Print message "The closes prime number to number "
syscall
li $v0,1
move $a0,$s1
syscall                     #Pint number entered by user
li $v0,4
la $a0,message3             #Print message " is: "
syscall
li $v0,1
move $a0,$t0                #Print prime number
syscall
j main

PrimeOne:
li $v0,4
la $a0,message2             #Print message "The closes prime number to number "
syscall
li $v0,1
move $a0,$s1
syscall                     #Pint number entered by user
li $v0,4
la $a0,message3             #Print message " is: "
syscall
li $v0,1
move $a0,$s1                #Print prime number
syscall
j main

end:
li $v0,4
la $a0,message2             #Print message "The closes prime number to number "
syscall
li $v0,1
move $a0,$s1
syscall                     #Pint number entered by user
li $v0,4
la $a0,message3             #Print message " is: "
syscall
li $v0,1
move $a0,$0                 #Print prime number
syscall
li $v0,10
syscall                     #End program
```

Output

```
Console

Enter a number from 0-100 or 00 to end program: 1
The closes prime number to number 1 is: 1
Enter a number from 0-100 or 00 to end program: 2
The closes prime number to number 2 is: 2
Enter a number from 0-100 or 00 to end program: 3
The closes prime number to number 3 is: 3
Enter a number from 0-100 or 00 to end program: 5
The closes prime number to number 5 is: 5
Enter a number from 0-100 or 00 to end program: 10
The closes prime number to number 10 is: 7
Enter a number from 0-100 or 00 to end program: 20
The closes prime number to number 20 is: 19
Enter a number from 0-100 or 00 to end program: 50
The closes prime number to number 50 is: 47
Enter a number from 0-100 or 00 to end program: 100
The closes prime number to number 100 is: 97
Enter a number from 0-100 or 00 to end program: 101
Number must be from 0-100.

Enter a number from 0-100 or 00 to end program: -1
Number must be from 0-100.

Enter a number from 0-100 or 00 to end program: 0
The closes prime number to number 0 is: 0
```

Assignment 3:

This program calculates two different sums of two numbers entered by user (n1 and n2). The first sum consist of all even numbers in n1 and n2. The second sum consist of all odd numbers greater than first number but less than the second number.

```
#Loenardo Roman lr534
#This program calculates the sume of two numbers n1 and n2.
#The first sum consist of all even numbers in n1 and n2
#The second sum consist of all odd numbers greater then n1 but less than n2.

.data
message1: .asciiz "Please enter two numbers from 0-1,000\n"
message2: .asciiz "Number 1: "
message3: .asciiz "Number 2: "
message4: .asciiz "Number must be from 0-1,000.\n"
message5: .asciiz "The sum of all even numbers in number1 and number2 is: "
message6: .asciiz "\nThe sum of all odd numbers greater than number 1 but less than number 2 is: "

.text
.globl main

main:
addi $s1,$0,1000            #max limit
li $v0,4                    #To print message1
la $a0,message1             #Store message address in $a0
syscall                     #Print message

Num1:                       #get first number form user
li $v0,4                    #To print string
la $a0,message2
syscall                     #print message 2
li $v0,5                    #Prompt user to enter number 1
syscall                     #Read integer
move $s0,$v0                #put number into $s0
blt $s0,$0,OuttaRange1      #if number is less than 0 go to OuttaRange1
bgt $s0,$s1,OuttaRange1     #if number is greater than 1000 go to OuttaRange2
add $t0,$0,$s0              #$t0 = num1

Num2:                       #get number 2 from user
li $v0,4                    #To print a string
la $a0,message3             #store message 3 address into $a0
syscall                     #Print message3
li $v0,5                    #Prompt user to enter number 2
syscall                     #Read integer
move $s0,$v0                #Put number in $s0
blt $s0,$0,OuttaRange2      #if number is less than 0 go to OuttaRange1
bgt $s0,$s1,OuttaRange2     #if number is greater than 1000 go to OuttaRange2
add $t1,$0,$s0              #$t1 = num2

LoopSumEven1:               #Sum all Even numbers in n1
addi $t2,$t2,0              #i=0
bgt $t2,$t0,LoopSumEven2    #if i>n1 All even numbers in n1 are sumed. Go to second sum.
add $s2,$s2,$t2             #sum+=i
addi $t2,$t2,2              #i+=2
j LoopSumEven1              #Keep looping till done.
```

```
LoopSumEven2:           #Sum all number even number in n2
addi $t3,$t3,0          #i=0
bgt $t3,$t1,EvenSum     #if i>n2 All even numbers in n1 are sumed. Go to second sum.
add $s3,$s3,$t3         #sum+=i
addi $t3,$t3,2          #i+=2
j LoopSumEven2          #Keep looping till done.

EvenSum:                #total Even_Sum = Sum_n1 + Sum_n2
add $s2,$s2,$s3         #$s2 = ESum_n1 + ESum_n2
li $v0,4                #To print a string
la $a0,message5         #Store message5 address in $a0
syscall                 #Print message5
li $v0,1                #To print an int
move $a0,$s2            #Put total sum in $a0 temporarily to be printed
syscall                 #Print total even sum
bne $t0,$t1,LoopSumOdds #if number 1 is not equal to number 2 go to LoopSumOdds

LoopSumOdds:            #Sum all odd in [n1,n2]
addi $s4,$0,2           #$s4 = 2(divisor)
bgt $t0,$t1,OddSum      #if n1 > n2 go to oddSum to print sum
div $t0,$s4             #n1/2 to find odd numbers
mfhi $t4                #Get remainder
bne $t4,$0,Increment_n1 #If remainder != 0 go to Increment_n1
addi $t0,$t0,1          #n1++
j LoopSumOdds           #Keep iterating to find all odd numbers

Increment_n1:           #Suming all odd numbers
add $s5,$s5,$t0         #sum += n1
addi $t0,$t0,1          #n1++
j LoopSumOdds           #go back to LoopSumOdds

OddSum:                 #Display sum
li $v0,4                #To print a string
la $a0,message6         #Store message6 address to be printed
syscall                 #Print message6
li $v0,1                #To print an int
move$a0,$s5             #Put total sum in $a0 temporarily to be printed
syscall                 #Print total odd sum
li $v0,10               #To end program
syscall                 #End program

OuttaRange1:            #validation 1 [0,1000]
li $v0,4
la $a0,message4
syscall
j Num1                  #Go back to Num1 to re-enter first number

OuttaRange2:            #Validation 2 [0,1000]
li $v0,4
la $a0,message4
syscall
j Num2                  #Go back to Num2 to re-enter first number
```

Output

Console

```
Please enter two numbers from 0-1,000
Number 1: 100
Number 2: 500
The sum of all even numbers in number1 and number2 is: 65300
The sum of all odd numbers greater than number 1 but less than number 2 is: 60000
```

Assignment 4:

In this assignment, we'll create a program prompts the user to enter the number of rows to print a diamond using starick characters. The diamond length should be equal to the number of rows entered by user.

Pseudocode

```
1. Enter number of rows
2. if (number%2==0)
3. for (i = 0; i < number; i++)
4. for (j = 0; j < number - i; j++ )
5. print(" ")
6. for ( j = number/2; j <= i; j++ )
7. print(" *")
8. print("\n")
9. for (i = number/2; i > 0; i-- )
10. for (j = number/2; j <= number-i; j++ )
11. print(" ")
12. for (j = i; j > 0; j-- )
13. print(" *")
14. System.out.print("\n")
15. if (number%2 != 0)
16. for (i = 0; i < number; i++)
17. for (j = 0; j < number-i; j++ )
18. print(" ")
19. for (j = number/2; j <= i; j++ )
20. print(" *")
21. print("\n")
22. for (i = number/2; i > 0; i-- )
23. for ( j = number/2; j <= number-i; j++ )
24. print(" ")
25. for (j = i; j > 0; j-- )
26. print(" *")
27. print("\n")
```

```
#Leonardo Roman lr534
#Lab#2 assignment 4
#This program asks the user to enter number of lines to print a diamond
#Diamond length should be equal the number to the number of lines entered by user
.data
meassage:   .asciiz "Enter number or rows: "
space:      .asciiz " "
starick:    .asciiz " *"
nextLine:   .asciiz "\n"
.text
.globl main

main:
li $v0,4                    #To print message
la $a0,meassage             #Store message address
syscall                     #Print message
li $v0,5                    #Prompt the user to enter number of rows for diamond
syscall
move $t1,$v0                #store number of rows in $t1
blt $t1,$0,end              #if number entered is negative terminate the program
li $s2,0                    #i = 0
li $s0,2                    #divisor $s0 = 2
div $t1,$s0                 #n/2
mflo $s1                    #store quotient in $s1 (if 7/3 then $s1 = 2)-> j = $s1
mfhi $s3                    #$s3 = remainder
beq $s3,$0,EvenLoop         #if remainder equals zero make even diamond
addi $t4,$s1,1              #$t4 is the upper limet of diamond quotient(n/2)+1
add $t3,$t4,$0              #$t3 is a copy of $t4 and equal quotient(n/2)+1

#-------------------Upper boddy of diamond-----------------------------------------------
DimondUpperBody:            #for(i=0;i<n;i++)
li $s1,0                    #reset j = 0
addi $s2,$s2,1              #i++
li $v0,4
la $a0,nextLine             #print("\n")
syscall
li $t0,0                    #j=0
ble $s2,$t4,Jloop2          #i<n go to jloop to print spaces
```

```
#-------------------Lower boddy of diamond-----------------------------------------------
DimondLowerBody:
mflo $s1                    #$s1 = 2(limit)
li $s2,0                    #$s2 = i = 0
bne $s3,$0,OddLoop          #if remainder does not equal 0 make odd diamond
addi $t4,$s1,0              #$t4 = 2 is the lower limit of diamond quotient(n/2)

Outer_loop:
li $t0,0                    #j = 0
li $t2,0                    #k = 0
addi $s2,$s2,1              #i++
ble $s2,$s1,Inner_Loop1     #if i <= n print space

Inner_Loop1:
bge $t0,$s2,Inner_Loop2     #if j => i go to Inner_Loop2 to print("*")
li $v0, 4                   #else print("-")
la $a0,space                #print(" ")
syscall
addi $t0,$t0,1              #j++
j Inner_Loop1

Inner_Loop2:
bge $t2,$t4,NewLine         #if k >= $s1 go to NewLine to print("\n")
li $v0,4                    #else print("*")
la $a0,starick
syscall
addi $t2,$t2,1              #k++
j Inner_Loop2

NewLine:
li $v0,4
la $a0,nextLine             #print("\n")
syscall
addi $t4,$t4,-1             #n--
beq $t4,$0,main             #if n = 0 finish program
j Outer_loop

OddLoop:
li $s2,2                    #$s3 = i = 2
li $t2,0                    #$t2 = k = 0
addi $t4,$s1,0              #$t4 = 2 is the lower limit of dimond quotient(n/2)
j Inner_Loop1

end:
li $v0,10
syscall
```

Output

```
Enter number or rows: 7

     *
    * *
   * * *
  * * * *
   * * *
    * *
     *
Enter number or rows: 9

      *
     * *
    * * *
   * * * *
  * * * * *
   * * * *
    * * *
     * *
      *
Enter number or rows: 4

   *
  * *
  * *
   *
Enter number or rows: 8

     *
    * *
   * * *
  * * * *
  * * * *
   * * *
    * *
     *
Enter number or rows:
```

Assignment 5

In this assignment, we'll create a program that asks the user to create a password. The password should contain only upper or lower case letters. Once the password has pass all requirements, user should re-enter the password for validation and has two chances if entered a wrong password before the program terminates.

```
#Leonardo Roman lr534
#This program prompts the user to enter a password
#Password should contained only upper or lower case letters
#and sould be re-entered for confirmation.
.data
message1: .asciiz "Set a pasword: "
message2: .asciiz "Failed! Please enter a password with the size of 8 to 12.Try again.\n"
message3: .asciiz "Pasword requires only upper and lower case letters. Try again.\n"
message4: .asciiz "Re-enter the password: "
message5: .asciiz "Incorrect, you have 2 more chances! Please re-enter password."
message6: .asciiz "Pasword is setup."
invalid:  .asciiz "Invalid password.\nProgram ended.\n"
password: .space  16

.text
.globl main

main:
li $t3,1                        #counter
li $s7,2                        #counter2
li $s0,0xa                      #$s0 = null char
li $s1,0x41                     #$s1 = 'A'
li $s2,0x5a                     #$s2 = 'Z'
li $s3,0x61                     #$s3 = 'a'
li $s4,0x7a                     #$s2 = 'z'
li $t0,0                        #$t0 = i = 0
addi $sp,$sp,-4                 #save a word space
sw $s7,0($sp)                   #store $s7 on stack

EnterPassword:
li $v0,4                        #Printing message
la $a0,message1
syscall
li $v0,8                        #Pront user to enter password
la $a0,password
li $a1,16
move $s5,$a0                    #$s5 = password[0](based address)
addi $sp,$sp,-4                 #save a word space
sw $s5,0($sp)                   #store password on stack
syscall
```

```
loop:
add $t1,$t0,$s5                 #$t1 = password[i]
lb $t2,0($t1)                   #load byte by byte
beq $t2,$s0,CheckValidPasswod   #end of password, null character
sge $t4,$t2,$s1                 #$t4 = 1 if $t2 >= 'A' else 0 invalid (not a letter)
sle $t5,$t2,$s2                 #$t5 = 1 if $t2 <= 'Z' else 0 invalid (not a letter)
sge $t6,$t2,$s3                 #$t6 = 1 if $t2 >= 'a' else 0 invalid (not a letter)
sle $t7,$t2,$s4                 #$t7 = 1 if $t2 <= 'z' else 0 invalid (not a letter)
and $t8,$t4,$t5                 #$t8 = $t4 && $t5
and $t9,$t6,$t7                 #$t9 = $t6 && $t7
or $s6,$t8,$t9                  #$s6 = $t8 || $t9
beq $s6,$0,Error2               #if $s6 == 0 then not valid password
addi $t0,$t0,1                  #i++
j loop

CheckValidPasswod:
li $t3,8                        #$s2 = 8
blt $t0,$t3,Error               #if i < 8 go to error
addi $t3,$t3,4                  #$s2 = 12
bgt $t0,$t3,Error               #if i > 12 go to error
lw $s7,0($sp)                   #$s7 = 2
li $t0,0                        #reset counter

ReEnterPassword:
li $v0,4                        #Printing message
la $a0,message4                 #"Re-enter password"
syscall
li $v0,8                        #Pront user to enter password
la $a0,password                 #put password in $a0
li $a1,16                       #Length of address
move $t9,$a0                    #$s5 = password[0](based address)
syscall

loop2:
add $t1,$t0,$t9                 #$t1 = REpassword[i]
lb $t2,0($t1)                   #load byte by byte
lw $t3,0($sp)                   #$t3 = $s5
add $t4,$t0,$t3                 #$t4 = password[i]
lb $t7,0($t4)                   #load byte by byte
bne $t2,$t7,Error3              #if REpassword[i] != password[i] break
addi $t0,$t0,1                  #i++
beq $t2,$s0,ValidPassword       #end of password, null character
j loop2                         #keep looping

ValidPassword:                  #check for validation
li $v0,4                        #to print string
la $a0,message6                 #Valid password
syscall                         #Print valid pass
li $v0 10                       #end program
syscall
```

```
#-----------------------------Errors-----------------------------------------------------
#----------------------------------------------------------------------------------------
Error:                          #error 1 if pass < 0 or pass > 12 characters
beq $s7,$0,end                  #terminate program if no more chances
li $v0,4
la $a0,message2                 #Display proper message
syscall
addi $s7,$s7,-2                 #counter2--(user gets one more chance)
j EnterPassword                 #go back to reenter pass

Error2:                         #error 2 if password contains any char that is not a letter
beq $t3,$0,end                  #terminate program if no more chances
li $v0,4
la $a0,message3                 #Display proper message
syscall
addi $t3,$t3,-1                 #counter--(user gets one more chance)
j EnterPassword                 #go back to reenter pass

Error3:                         #error 3 if re-enter pass does not match entered pass
beq $s7,$0,end                  #terminate program if no more chances
li $v0,4
la $a0,message5                 #Display proper message
syscall
addi $s7,$s7,-1                 #counter2--(user gets two more chance)
j ReEnterPassword               #go back to re-enter pss

end:                            #End program at error
addi $sp,$sp,8                  #re store stack
li $v0,4
la $a0,invalid
syscall
li $v0 10
syscall
```

Output

```
Console
Set a pasword: ascd,asd
Pasword requires only upper and lower case letters. Try again.
Set a pasword: adcs,cda
Invalid password.
Program ended.
```

```
Console
Set a pasword: asdfgtredfgvds
Failed! Please enter a password with the size of 8 to 12.Try again.
Set a pasword: asdf
Invalid password.
Program ended.
```

```
Set a pasword: abcdABCD
Re-enter the password: abcdABCD
Pasword is setup.
```

Assignment 6:

This program first sorts an array of numbers and then asks the user to enter a number and if the number is in the array it prints out the position where number. If number is not found, the number will be added

```
.data
array:          .word   4, 5, 23, 5, 8, 3, 15, 67, 8, 9, 0xFF

sortedArray:    .asciiz "Sorted Array: "
key:            .asciiz "\nEnter a number to search or negative integer to end: "
message1:       .asciiz "\nNumber not found. Added to array.\n"
message2:       .asciiz "\nNumber at index "
space:          .asciiz ", "
message3:       .asciiz "\nProgram Terminated."

.text
.globl main

main:
jal ArraySize                   #get array size
jal SortArray                   #sort array
jal PrintArray                  #print array

prompt:
li $v0,4                        #print message
la $a0,key                      #load key address to be printed
syscall
li $v0,5                        #Prompt the user to enter key
syscall
move $s0,$v0                    #$s0 = $v0 = key
blt $s0,$0,End                  #if $s0 = negative number, terminate
jal SearchKey                   #call SearchKey
jal Add                         #call add and add if key was not found
j main                          #go back to main

ArraySize:
li $t1,0                        #t1 = 0 (for size)
la $t0,array                    #t0 = base address
li $t9,0xFF                     #t9 = 0xFF last word

Loop:
lw $t2, 0($t0)                  #t1 = A[i]
beq $t2,$t9,EndLoop             #if word is 0xFF, end loop
addi $t1,$t1,1                  #i++
addi $t0,$t0,4                  #increment t0 to next word
j Loop                          #jump to Loop

EndLoop:
addi $sp,$sp,-4                 #adjust stack pointer and save one word space
sw $t1,0($sp)                   #store size on stack
jr $ra                          #return to caller

SortArray:
lw $t9,0($sp)                   #$t9 = size of array
addi $t9,$t9,-1                 #size--
li $t7, 0                       #$t7 = 0
```

```
OutterLoop2:
beq $t7,$t9,EndLoop2        #if $t7 = $t9, end loop
la $t0,array                #$t0 = address of array
li $t8,0                    #$t8 = 0

InnerLoop:
beq $t8,$t9,EndInnerLoop    #if $t8 = $t9, go end loop
lw $t1,0($t0)               #$t1 = A[i]
lw $t2,4($t0)               #$t2 = A[i+1]
slt $t3,$t2,$t1             #$t3 = 1, if A[i+1] < A[i]
bne $t3,$0,Swap             #if $t3 = 1, jump to swap
addi $t8,$t8,1              #$t8++
addi $t0,$t0,4              #$t0 += 4
j InnerLoop                 #jump to InnerLoop

Swap:
sw $t2,0($t0)               #A[i] = A[i+1]
sw $t1,4($t0)               #A[i+1] = A[i]
addi $t0,$t0,4              #t0 += 4
addi $t8,$t8,1              #t8++
j InnerLoop                 #jump to InnerLoop

EndInnerLoop:
addi $t7,$t7,1              #$t7++
j OutterLoop2               #jump to OutterLoop2

EndLoop2:
jr $ra                      #return to caller

PrintArray:
li $v0,4                    #To print string
la $a0,sortedArray          #load address to a0
syscall
la $t0,array                #$t0 = address of array
li $t9,0xFF                 #$t9 = 0xFF

Loop2:
li $v0,1                    #To print string
lw $a0,0($t0)               #load address to a0
beq $a0,$t9,EndLoop3
syscall
li $v0,4                    #to print string
la $a0,space                #load address of space to a0
syscall
addi $t0,$t0,4              #$t0 += 4
j Loop2                     #jump to Loop2

EndLoop3:
jr $ra                      #return to caller
```

```
EndLoop3:
jr $ra                          #return to caller

SearchKey:
lw $t9,0($sp)                   #$t9 = size of array
li $t7,0                        #$t7 = 0 (to get index)
la $t0,array                    #$t0 = address of array

LinearSearch:
lw $t1,0($t0)                   #$t1 = A[i]
beq $t1,$s0,KeyFound            #A[i] = key, jump to found
addi $t0,$t0,4                  #next word
addi $t7,$t7,1                  #next index
beq $t7,$t9,KeyNotFound         #end of array, jump to not found
j LinearSearch                  #jump to LinearSearch

KeyFound:
li $v0, 4                       #print string
la $a0, message2                #$a0 = address of message2
syscall
li $v0,1                        #print int
move $a0,$t7                    #$a0 = index
syscall
j prompt                        #jump to prompt

KeyNotFound:
jr $ra

Add:
la $t0,array                    #$t0 = address of array
li $t9,0xFF                     #$t9 = 0xFF
lw $t8,0($sp)                   #$t8 = size of array
sll $t8,$t8,2                   #$t8 = 4*$t8
add $t0,$t0,$t8                 #$t0 = $t0 + $t8 (end of array)
sw $s0,0($t0)                   #add number to array
sw $t9,4($t0)                   #end of array
li $v0,4                        #print string
la $a0,message1                 #$a0 = address of KeyNotFound
syscall
jr $ra

End:
li $v0,4                        #print string
la $a0,message3                 #$a0 = message3
syscall
li $v0,10                       #exit
syscall
```

Output

```
Console
Sorted Array: 3, 4, 5, 5, 8, 8, 9, 15, 23, 67,
Enter a number to search or negative integer to end: 1

Number not found. Added to array.
1, 3, 4, 5, 5, 8, 8, 9, 15, 23, 67,
Enter a number to search or negative integer to end: 67

Number at index 10
Enter a number to search or negative integer to end: 2

Number not found. Added to array.
C1, 2, 3, 4, 5, 5, 8, 8, 9, 15, 23, 67,
Enter a number to search or negative integer to end: 2

Number at index 1
Enter a number to search or negative integer to end: -1

Program Terminated.
```

Conclusion

Flow control in this lab was the core structure for every assignment. Every loop was possible to perform their tasks by using branches and jump instructions. Other boolean logics were used in other to find all true values so all loops perform their given tasks. For larger programs such as assignments 5 and 6 the use of stack pointer was necessary for reuse of registers. All Pseudocode lead to building process of all programs and all programs ran successfully.