

CSE 3038 Computer Organization

Programming Project 1

Koray Emre Şenel – 150117037

Mehmet Etka Uzun – 150118504

Alperen Köker – 150117035

Ahmet Turgut - 150117046

Question 1 : In this program, program takes input as a string and find the occurrence of chars.

```
countChar:
    li $t0, 0
    li $t1, 26
loopZero:
    beq $t1, $t0, exitZero
    sb $zero, myArray($t0)
    sb $zero, myArray2($t0)
    sb $zero, myArray3($t0)
    addi $t0, $t0, 1
    j loopZero
exitZero:
```

```
li $v0, 4
la $a0, enterStrText
syscall
li $v0, 8
la $a0, inputString
li $a1, 256
syscall
```

```
li $t3, -87 #t3 = -87
la $s0, charA #s0 = 'a'
la $s1, inputString #s1 = input
```

```
loop:
    lb $t1, 0($s1) #t1 = input[i]
    lb $t2, 0($s0) #t2 = 'a'
    beqz $t1, exit
    ble $t1, 90, fun
    sub $t0, $t1, $t2 #t0 = t1 - t2
    beq $t3, $t0, exit #if t0 == -87 exit
    lb $t4, myArray($t0)
    addi $t4, $t4, 1
    sb $t4, myArray($t0)
    addi $s1, $s1, 1
    j loop
```

```
fun:
    addi $t5, $t1, 32
    sub $t0, $t5, $t2 #t0 = t1 - t2
    beq $t3, $t0, exit
    lb $t4, myArray($t0)
    addi $t4, $t4, 1
    sb $t4, myArray($t0)
    addi $s1, $s1, 1
    j loop
```

```
exit:
```

In this part we reset the array.

Because when user call this part of program again , program add the new value to old result.

Program take input from user as a string. After that operation program store that input in s1 register.

In this part of the program iterates all chars in string. When program iterate, it counts them and stores in myArray. If program encounter uppercase, program calls the function which convert uppercase to lowercase.

In this function, program add 32 ASCII value of the char. After, program obtain lowercase version of the char.

```

li $t0, 0
li $t1, 26
loop2:
    beq $t1, $t0, exit2
    sb $t0, myArray2($t0)
    addi $t0, $t0, 1
    j loop2
exit2:

```

In this part program fills myArray value of in order to 0 to 25.

```

-- -- --
loop42:
    beq $t1, $t0, exit42
    lb $t2, myArray($t0)
    sb $t2, myArray3($t0)
    addi $t0, $t0, 1
    j loop42
exit42:

```

This part of code copy the myArray to myArray3.

```

li $t0, 0 # t0 = 0
li $t1, 25 # t1 = 25

forloop:
    beq $t1, $t0, exit3 # if t0 == t1 exit3
    li $t2, 0 # t2 = 0
    sub $t3, $t1, $t0 # t3 = $t1 - $t0
    forloopInner:
        beq $t3, $t2, exit4 # if t3 == t2 exit4
        addi $t4, $t2, 1 # t4 = t2 + 1
        lb $t5, myArray3($t2) # t5 = myArray[t2]
        lb $t6, myArray3($t4) # t6 = myArray[t4]
        blt $t5, $t6, fun1 # if t5 < t6 fun1
        addi $t2, $t2, 1 # t2++
        j forloopInner
    fun1:
        lb $t5, myArray2($t2) # t5 = myArray2[t2]
        lb $t6, myArray2($t4) # t6 = myArray2[t4]
        sb $t6, myArray2($t2) # myArray2[t2] = t6
        sb $t5, myArray2($t4) # myArray2[t4] = t5
        lb $t5, myArray3($t2) # t5 = myArray2[t2]
        lb $t6, myArray3($t4) # t6 = myArray2[t4]
        sb $t6, myArray3($t2) # myArray2[t2] = t6
        sb $t5, myArray3($t4) # myArray2[t4] = t5
        addi $t2, $t2, 1 # t2++
        j forloopInner

```

We implement bubble sort. When we sort myArray, we do not change myArray because we do not want to lose ASCII value.

This part is basic swap method for bubble sort.

Example Output:

```

Please select an option: 1
Enter the String: This is Computer Organization Course!
Character Occurence
o          4
i          3
r          3
s          3
t          3
a          2
c          2
e          2
n          2
u          2
g          1
h          1
m          1
p          1
z          1
Welcome to our MIPS project!
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 1
Enter the String: How can i go to Taksim?
Character Occurence
o          3
a          2
i          2
t          2
c          1
g          1
h          1
k          1
m          1
n          1
s          1
w          1

```

Question 2:

In this part of project, program sort the given array.

```
sortNumbers:
    la $a0, enterStrText    # Load and print string asking for string
    li $v0, 4
    syscall

    la $a0, space2
    li $v0, 4
    syscall
    li $v0, 8                # take in input

    la $a0, buffer          # load byte space into address
    li $a1, 1024            # allot the byte space for string

    move $t0, $a0           # save string to t0
    syscall

    li $t1, 0 # arrays input
    li $t5, 10 # 10
    la $t6, newline # to end loop
    lb $t6, 0($t6)
    li $t7, 1 # minusFlag
    li $t8, 0
    li $t9, 0
```

In this part, program take the input from user and set the value what we need.

```
loopSort:
    lb $t2, 0($t0) # load the next character into t2  1 6 23 -31
    beq $t2, $t6, exitSort
    lb $t4, minus
    beq $t2, $t4, minusFlag
    lb $t3, space2
    beq $t2, $t3, arrayFill # increment the string pointer
    mul $t1, $t1, $t5
    andi $t2, $t2, 0x0F # where $t0 contains the ascii digit .
    add $t1, $t1, $t2
    addi $t0, $t0, 1
    j loopSort # return to the top of the loop

exitSort:
    mul $t1, $t1, $t7
    addi $t8, $t8, 1
    sb $t1, xd($t9)
    li $t0, 0 # t0 = 0
    subi $t9, $t8, 1 # t9 = t8 - 1

forloopSort:
    beq $t9, $t0, exit3Sort # if t0 == t1 exit3
    li $t2, 0 # t2 = 0
    sub $t3, $t9, $t0 # t3 = $t1 - $t0

forloopInnerSort:
    beq $t3, $t2, exit4Sort # if t3 == t2 exit4
    sll $t7, $t2, 2
    addi $t4, $t7, 4 # t4 = t2 + 1
    lb $t5, xd($t7) # t5 = myArray[t2]
    lb $t6, xd($t4) # t6 = myArray[t4]
    bgt $t5, $t6, fun1Sort # if t5 < t6 fun1
    addi $t2, $t2, 1 # t2++
    j forloopInnerSort

fun1Sort:
    lb $t5, xd($t7) # t5 = myArray2[t2]
    lb $t6, xd($t4) # t6 = myArray2[t4]
    sb $t6, xd($t7) # myArray2[t2] = t6
    sb $t5, xd($t4) # myArray2[t4] = t5
    addi $t2, $t2, 1 # t2++
    j forloopInnerSort

exit4Sort:
    addi $t0, $t0, 1 # t0++
    j forloopSort
```

In this part, as we mentioned in question 1, we implement bubble sort. This implementation has only one difference, sorting the values in ascending order. fun1Sort method swap values of the array for bubble sort.

-

Example Output:

```
Please select an option: 2
Enter the String: 2 3 -6 4 22 -12
-12
-6
2
3
4
22
Welcome to our MIPS project!
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 2
Enter the String: 0 4 0 -12 12 33
-12
0
0
0
4
12
33
```

Question 3:

In this program, find the prime numbers in given intervals.

```
prime:
    la $a0, enterStrText    # Load and print string asking for string
    li $v0, 4
    syscall

    la $a0, space2
    li $v0, 4
    syscall
    li $v0, 8               # take in input

    la $a0, buffer          # load byte space into address
    li $a1, 1024            # allot the byte space for string

    move $t0, $a0           # save string to t0
    syscall
    li $t1, 0
    li $t5, 10
    la $t6, newLine
    lb $t6, 0($t6)
```

This part of program takes an input from user.

```

loopPrime:
    lb    $t2,0($t0)
    beq   $t2,$t6,exitPrime
    mul   $t1,$t1,$t5
    andi  $t2,$t2,0x0F
    add   $t1,$t1,$t2
    addi  $t0,$t0,1
    j     loopPrime

```

In this part, program convert string to integer from input.

```

exitPrime:
    addi  $t0,$zero,2 # $t0 = 2 = p
    addi  $t2,$zero,0
    addi  $t3,$zero,0
    addi  $t4,$zero,0
    addi  $t5,$zero,0
    addi  $t6,$zero,0
    addi  $t9,$zero,0
    j     calculatePrime

```

In exitPrime initializes registers with zero. And we add 2 to the \$t0. We decide P as \$t0 register.

```

calculatePrime:
    mul   $t2,$t0,$t0 #t0 = p , $t2 = p*p
    addi  $t3,$zero,2 # t3 = i = 2 for print section
    bgt   $t2,$t1, printPrime #greater than p^2 > n
    add   $t4,$zero,$t0 #t4 = p
    sll   $t4,$t4,2      #t4=p*4
    lw    $t5,array($t4) #t5 = array[p]
    addi  $t6,$zero,1    #t6=1 true
    mul   $t7,$t0,$t0    #t7 = i = p^2
    beq   $t5,$t6, forloop2Prime
    addi  $t0,$t0,1 # p++
    j     calculatePrime

```

```

for (int p = 2; p * p <= n; p++){
    if (prime[p] == true)
    {
        for (int i = p * p; i <= n; i += p)
            prime[i] = false;
    }
}

```

In this part we calculate P value. (\$t2= P*P)(Orange section in C code)
 If p*p greater than value of input jump to printPrime section.
 We control array[p] is equal to 1.
 If array[p] equal is 1 jump to forloop2Prime section.

```

forloop2Prime:
    bgt   $t7,$t1,increment
    sll   $t4,$t7,2      #t4 = p*4
    sb    $zero,array($t4) #array[i] = 0
    add   $t7,$t7,$t0    #i = i + p
    j     forloop2Prime

```

We update all multiples of p. (Green section in C code.)

```

printPrime:
    bgt $t3,$t1,exit2Prime
    add $t4,$zero,$t3      #t4 = i
    sll $t4,$t4,2          #t4=i*4
    lw  $t5,array($t4)     #t5 = array[i]
    addi $t6,$zero,1       #t6 = 1
    beq $t5,$t6,printFuncPrime
    addi $t3,$t3,1 #i = i+1
    j printPrime
printFuncPrime:
    addi $t9,$t9,1
    addi $t3,$t3,1
    j printPrime

```

Calculate prime numbers 0 to value of input.

```

exit2Prime:
    la $a0,($t9)
    li $v0, 1
    syscall
    li $v0, 4
    la $a0, space
    syscall
    j menu

```

In this section print number of prime to the console. Then jump menu section.

```

increment:
    addi $t0,$t0,1 # p++
    j calculatePrime

```

Example Output:

```

Please select an option: 3
Enter the String: 100
25
Welcome to our MIPS project!
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 3
Enter the String: 1000000
78498

```

Question 4:

In this part of project, program take two input from user as a String and convert the Huffman Code Tree.

```
li $v0, 4
la $a0, HufmannStr1
syscall

li $v0, 8
la $a0, inputString #take input
li $a1, 256
syscall

li $t3, -87 #t3 = -87
la $s0, charA #s0 = 'a'
la $s1, inputString #s1 = input
```

In this part of program, it takes input .

After that part, we use the same calculate occurrence and sorting algorithm in question1.

```
storeArrayLoop:
    beq $t0,$zero,initializeForAddingNode
    subi $t0,$t0,1
    lb $t2, myArray2($t0)
    lb $t5, 0($s0) #t2 = 'a'
    add $t5, $t5, $t2
    move $s2, $t5

    sw $s2,arrayChar($t1) #node store in arrayChar

    lb $t4, myArray3($t0)
    move $s2, $t4
    sw $s2,arrayData($t1) #node.data store in arrayData

    addi $t1,$t1,4 #increment adress value of arrays
    addi $t9,$t9,1 ##increment queue size

j storeArrayLoop
```

In this part of program, we store the sorted chars and occurrences in arrayChar and arrayData.

startAddingNode:

```

beq $t9,$s1,startAddingNodeExit
lb $s4,arrayDelete($s0) #store 0 in arrayDelete
bne $s4,$zero,returnAddingNode
lb $s2,arrayData($s0) #x for left of node
sb $s1, arrayDelete($s0) #x's delete is 1
mul $s6,$t8,4 #size*4 = 6*4 =24
sb $s0, arrayLeft($s6) #left added
addi $s0,$s0,4 #t0+4 = $t0

lb $s3,arrayData($s0) #y for right of node
sb $s1, arrayDelete($s0) #y's delete is 1
add $s5,$s2,$s3 #x.data+y.data

sb $s5, arrayData($s6) #newArray indexing

sb $s0, arrayRight($s6) #right added

addi $t9,$t9,-1
addi $t8, $t8, 1
move $s5,$t8 #calculate root node
addi $s5,$s5,-1
sll $s5,$s5,2 #calculate root node address

j sorting

```

findLeftOrRightPattern:

```

beq $t4,$s5,printConvertString #if last node found jump print

lb $t7,arrayLeft($t3) #t7 is arrayLeft data
lb $t8,arrayRight($t3) #t8 is arrayRight data

beq $t7,$t4,leftEqual #if 5 == arrayLeft
beq $t8,$t4,rightEqual #if 5 == arrayRight any element jump rightEqual

addi $t3,$t3,4 #increment address value of arrays
j findLeftOrRightPattern

```

leftEqual:

```

add $t4,$t3,$zero #address of node in t4
li $s0,0 #for print 0
mul $s3,$s1,4
sb $s0,arrayConvertedString($s3) #store 0 in arrayConvertedString
addi $s1,$s1,1 #increment size
li $t3,0
j findLeftOrRightPattern

```

rightEqual:

```

add $t4,$t3,$zero #address of node in t4
li $s0,1 #for print 1
mul $s3,$s1,4
sb $s0,arrayConvertedString($s3) #store 1 in arrayConvertedString
addi $s1,$s1,1 #increment size
li $t3,0
j findLeftOrRightPattern

```

printConvertString:

```

blt $s3,$zero,convertHuffman
lb $t4,arrayConvertedString($s3)
addi $s3,$s3,-4 #decrease address of array

move $s2, $t4
li $v0, 1
move $a0, $s2
syscall

j printConvertString

```

exitConvert:

```

li $v0, 4
la $a0, space
syscall
j menu

```

In this part of program, we have the arrays that sorted by their values. We get the first and second index of array, then check if them not used by looking at arrayDelete array. After that we sum these indexes and add this new value to arrayData array. Also we store the addresses of these summed indexes in arrayRight and arrayLeft to use it like Trees. We mark these used indexes and send arrayData to sort again.

In this part of program, we trace arrayLeft and arrayRight to find corresponding data. When we find it, if we are in arrayLeft we write 0. Otherwise we write 1. Then we change the data and do this process until data is last node.

In this part we reverse the Huffman code that we found because we start from end of the tree. And we print this then continue with other character if we have any.

ALGORITHM:

Visualized version of algorithm that we used for structures.

array Data	1	2	3	4	5	6										
array Char	E	A	C	D	F	B										
array Left	1	1	1	1	1	1										
array Right	1	1	1	1	1	1										
array Delete	0	0	0	0	0	1										
	0	4	8	12	16	20	24	28	32	36	40					

1+2 →

array Data	1	2	3	4	5	6	3									
array Char	E	A	C	D	F	B	-									
array Left	1	1	1	1	1	1	0									
array Right	1	1	1	1	1	1	1									
array Delete	1	1	0	0	0		0									
	0	4	8	12	16	20	24	28	32	36	40					

Sorting

array Data	1	2	3	3	4	5	6									
array Char	E	A	C	-	D	F	B									
array Left	1	1	1	0	1	1	1									
array Right	1	1	1	4	1	1	1									
array Delete	1	1	0	0	0	0	0									
	0	4	8	12	16	20	24	28	32	36	40					

min two value

EXAMPLE OUTPUT:

```

Welcome to our MIPS project!
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 4
Enter the string to construct Huffman Code: AABBBCCBDDDDDEFFFFBC
Enter the string to be converted using Huffman Code:ABC
111110110
Welcome to our MIPS project!
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: |

```

Menu:

This program has a menu that including all options. There are welcome, menu and input system calls in this part.

```
Welcome to our MIPS project!
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 1
Enter the String: 2
```

Console part

menu:

```
li $v0, 4
la $a0, menuText
syscall
li $v0, 5
syscall
move $t0, $v0
beq $t0, 1, countChar
beq $t0, 2, sortNumbers
beq $t0, 3, prime
beq $t0, 4, huffman
beq $t0, 5, exitCode
```

Code part