**Character and String Manipulation, Arrays and Other Memory Operands**

**LAB # 06**



**Fall 2022**

**CSE304L Computer Organization & Architecture**

Submitted by:

Danyal Khursheed (20pwcse1959)

Maaz Habib (20pwcse1952)

Hamid (20pwcse1969)

Class Section: **C**

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Amaad Khalil**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Character and String Manipulation, Arrays and Other Memory Operands**

MIPS assembly language provides instructions for manipulating characters and strings, working with arrays, and accessing other memory operands. Here's a brief overview of these concepts in MIPS assembly:

Character and string manipulation: MIPS assembly language provides instructions for manipulating individual characters and strings of characters. For example, the lb and sb instructions can be used to load and store single characters from memory, while the lw and sw instructions can be used to load and store strings of characters.

Arrays: MIPS assembly language provides instructions for working with arrays of data. For example, you can use the la instruction to load the address of an array into a register, and then use the lw and sw instructions to load and store individual elements of the array. You can also use loops and conditional statements to manipulate arrays in more complex ways.

Other memory operands: MIPS assembly language provides instructions for accessing other memory operands, such as data structures, pointers, and variables. For example, the lw and sw instructions can be used to load and store values stored at memory addresses, while the la instruction can be used to load the address of a data structure into a register.

This is a basic overview of character and string manipulation, arrays, and other memory operands in MIPS assembly language. Please refer to the MIPS assembly language documentation for more information and advanced usage.

**Task #01**

Take input from user and develop a CV using string instructions.

**Code:**

.data

CV:.asciiz "...................................................................................................................

Name:.asciiz "Name: "

FatherName:.asciiz "FatherName: "

Education:.asciiz "Education: "

Skills:.asciiz "Skills: "

Experience:.asciiz "Experience: "

End:.asciiz "..................................................................................................................."

name:.space 20

Father\_Name:.space 20

eduction:.space 150

experienc:.space 20

skills:.space 150

str1:.asciiz " Please enter your Name: "

str2:.asciiz " Please enter your Father Name: "

str3:.asciiz " Please enter your Education: "

str4:.asciiz "Tell us about your Experience: "

str5:.asciiz "What are your Skills: "

.text

main:

li $v0, 4 #print string

la $a0, str1

syscall #system call

li $v0, 8 #Enter string from user

la $a0, name

li $a1, 20

syscall #system call

li $v0, 4 #print string

la $a0, str2

syscall #system call

li $v0, 8 #Enter string from user

la $a0, Father\_Name

li $a1, 20

syscall #system call

li $v0, 4 #print string

la $a0, str3

syscall #system call

li $v0, 8 #Enter string from user

la $a0, eduction

li $a1, 150

syscall #system call

li $v0, 4 #print string

la $a0, str4

syscall #system call

li $v0, 8 ##Enter string from user

la $a0, experienc

li $a1, 20

syscall #system call

li $v0, 4 #print string

la $a0, str5

syscall #system call

li $v0, 8 ##Enter string from user

la $a0, skills

li $a1, 150

syscall #system call

li $v0, 4 #print string

la $a0, CV

syscall #system call

li $v0, 4 #for name

la $a0, Name

syscall

li $v0, 4

la $a0, Name

syscall #system call

li $v0, 4 #for father name

la $a0, FatherName

syscall

li $v0, 4

la $a0, Father\_Name

syscall #system call

li $v0, 4 #for education

la $a0, Education

syscall

li $v0, 4

la $a0, eduction

syscall #system call

li $v0, 4 #for experience

la $a0, Experience

syscall

li $v0, 4

la $a0, experienc

syscall #system call

li $v0, 4 #for skills

la $a0, Skills

syscall

li $v0, 4

la $a0, skills

syscall #system call

li $v0, 4 #print string

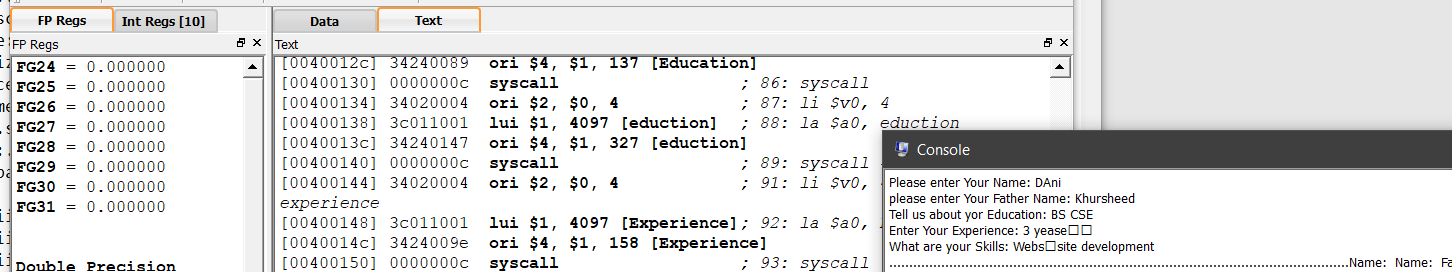
la $a0, End

syscall

li $v0, 10 #sys call code 10 is for exit

syscall #system call

**Output:**



**Task # 02**

Takes two numbers A and B from user, and prints out all the multiples of A from A to A \* B.

**Code:**

.data

str1: .asciiz"Please Enter value of A:"

str2: .asciiz"Please Enter value for B: "

str3: .asciiz"so the result is Multiples A to A\*B:"

str4: .asciiz" "

.text

main:

li $v0, 4 #print string

la $a0, str1

syscall #system call

li $v0, 5 #input value from user

syscall #system call

move $t0, $v0 #move number to t0

li $v0, 4 #print string

la $a0, str2

syscall #make sys call

li $v0, 5 #input value from user

syscall #system call

move $t1, $v0 #move the no read into #t1

li $v0, 4 #print string

la $a0, str3

syscall #system call

li $t2, 1

Multiplication:

mul $t3, $t0, $t2 #compute the multiplication

move $a0, $t3

li $v0, 1 #load system call print\_int

syscall

li $v0, 4 #print string

la $a0, str4

syscall #system call

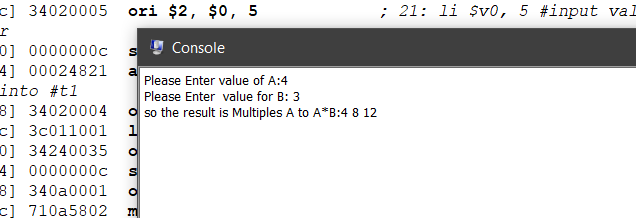
addi $t2, $t2, 1 #add t2 and t1

ble $t2, $t1, Multiplication #branch t2<t1

li $v0, 10 #sys call code 10 is for exit

syscall #make sys call

**Output:**



**Task # 03**

Write the program that reads an integer x from the user and outputs 20x without using mul and mult. Note that you can rewrite 20x as a sum of two terms each of which multiplies x by a power of 2.

**Code:**

data

str1: .asciizPlease Enter value of: "

str2: .asciiz" so the value of 20\*x: "

.text

main:

li $v0, 4 #print string

la $a0, str1

syscall #system call

li $v0, 5 #take value from user

syscall #system call

move $t0, $v0 #move number to t0

sll $t1, $t0, 2 #shift left by 2 bits

sll $t2, $t0, 4 #shift left by 4 bits

add $t3, $t1, $t2 #calculate 20\*x

li $v0, 4 #print string

la $a0, str2

syscall

move $a0, $t3

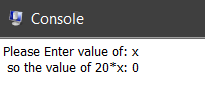
li $v0, 1 #load system call print\_int

syscall

li $v0, 10 #v0=10 is for exit

syscall #system call

**Output:**

****