Lab: 5: Character and String Manipulation, Arrays, and Other Memory Operands

Objective:

To explore string manipulation and memory operations in MIPS Assembly Language.

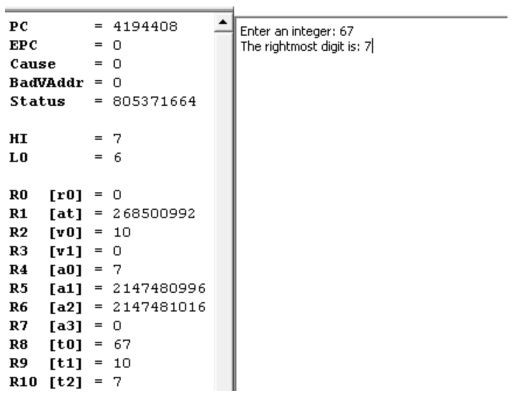
Task 1:

Create a program to prompt the user for an integer, read the input, and output the rightmost digit.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

#Task 1

.data
prompt1: .asciiz "Enter an integer: "
prompt2: .asciiz "The rightmost digit is: "

.text
    main:
        li $v0, 4                # syscall for print_string
        la $a0, prompt1           # load address of prompt
        syscall

        li $v0, 5                # syscall for read_int
        syscall
        move $t0, $v0            # store the integer in $t0

        li $t1, 10               # load 10 into $t1
        div $t0, $t1             # divide $t0 by 10
        mfhi $t2                 # move the remainder (rightmost digit) into $t2

        li $v0, 4                # syscall for print_string
        la $a0, prompt2         # load address of message
        syscall

        li $v0, 1                # syscall for print_int
        move $a0, $t2            # move the rightmost digit to $a0
        syscall

    fini:
        li $v0, 10               # syscall for exit
        syscall
```

Output:



```
PC       = 4194408
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 7
LO       = 6

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 7
R5  [a1] = 2147480996
R6  [a2] = 2147481016
R7  [a3] = 0
R8  [t0] = 67
R9  [t1] = 10
R10 [t2] = 7
```

```
Enter an integer: 67
The rightmost digit is: 7
```

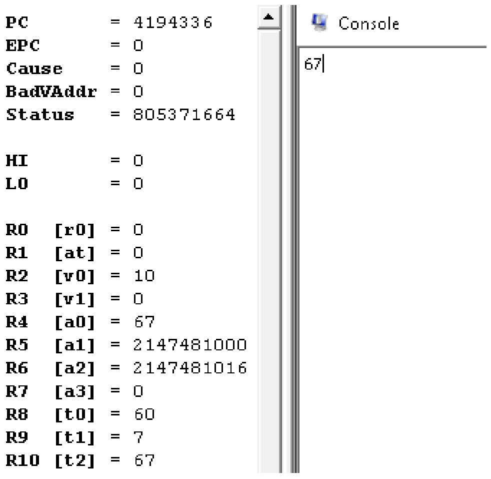
Task 2:

Modify `Lab5_1.asm` to print strings on separate lines using a newline character `\n`.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

#Task 2

.data
prompt1: .asciiz "Enter an integer:\n"
prompt2: .asciiz "The rightmost digit is:\n"

.text
    main:
        li $v0, 4               # syscall for print_string
        la $a0, prompt1          # load address of prompt
        syscall

        li $v0, 5               # syscall for read_int
        syscall
        move $t0, $v0           # store the integer in $t0

        li $t1, 10              # load 10 into $t1
        div $t0, $t1            # divide $t0 by 10
        mfhi $t2                # move the remainder (rightmost digit) into $t2

        li $v0, 4               # syscall for print_string
        la $a0, prompt2         # load address of message
        syscall

        li $v0, 1               # syscall for print_int
        move $a0, $t2           # move the rightmost digit to $a0
        syscall

    fini:
        li $v0, 10              # syscall for exit
        syscall
```

Output:



```
PC        = 4194336
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 805371664

HI        = 0
LO        = 0

R0  [r0]  = 0
R1  [at]  = 0
R2  [v0]  = 10
R3  [v1]  = 0
R4  [a0]  = 67
R5  [a1]  = 2147481000
R6  [a2]  = 2147481016
R7  [a3]  = 0
R8  [t0]  = 60
R9  [t1]  = 7
R10 [t2]  = 67
```

```
67
```

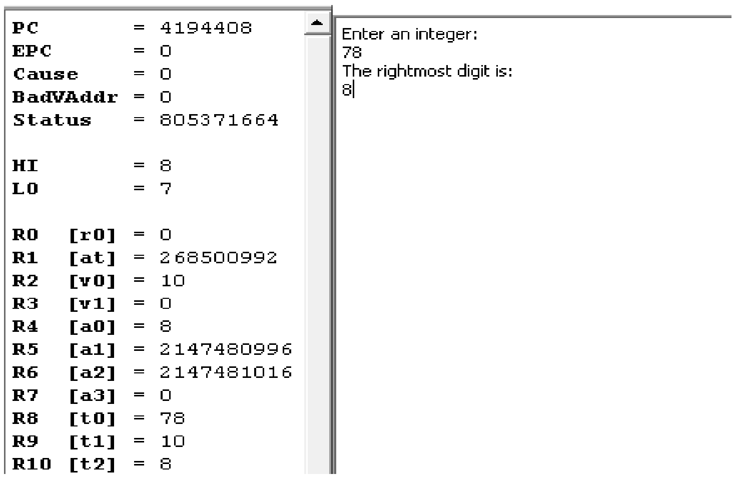
Task 3:

Investigate the difference between `.ascii` and `.asciiz` directives.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

#Task 3

.data
prompt: .ascii "Enter an integer...\n"  # Using .ascii without null-termination
.byte 0                                 # Manually add null terminator
message: .asciiz "The rightmost digit is:\n"  # Null-terminated message string
promptA: .word prompt                   # A pointer to the address of prompt

.text
    main:
        lw $a0, promptA($0)
        li $v0, 4
        syscall

        li $v0, 5
        syscall
        move $t0, $v0

        li $t1, 10
        div $t0, $t1
        mfhi $t2

        addi $a0, $a0, 21
        li $v0, 4
        syscall

        li $v0, 1
        move $a0, $t2
        syscall

    fini:
        li $v0, 10
        syscall
```

Output:



```
PC       = 4194408
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 8
LO       = 7

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 8
R5  [a1] = 2147480996
R6  [a2] = 2147481016
R7  [a3] = 0
R8  [t0] = 78
R9  [t1] = 10
R10 [t2] = 8
```

```
Enter an integer:
78
The rightmost digit is:
8
```

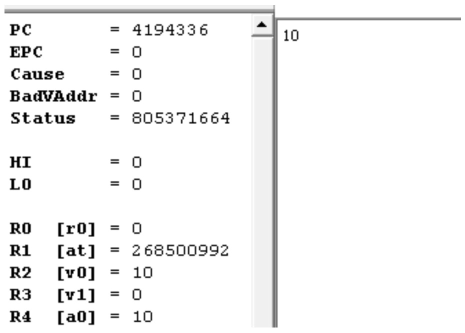
Task 4:

Write a function to calculate the length of a null-terminated string using the `jal` instruction.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

#Task 4

.data
entry: .ascii "Ahsan Raza"
.byte 0
entryA: .word entry
.text
main:
    add $s0, $ra, $0
    lw $a0, entryA($0)

    jal length
    move $a0, $v0
    li $v0, 1
    syscall

    add $ra, $s0, $0              # Restore the return address from $s0
fini:
    jr $ra                        # Return from main
length:
    add $v0, $0, $0               # Initialize length counter in $v0 to 0
length_loop:
    lb $t0, 0($a0)                # Load byte from string into $t0
    beq $t0, $0, length_end       # If byte is null (0), end loop
    addi $v0, $v0, 1              # Increment length counter
    addi $a0, $a0, 1              # Move to next byte in string
    j length_loop                 # Repeat loop
length_end:
    jr $ra                        # Return to caller with length in $v0
```

Output:



```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 10
```

```
10
```

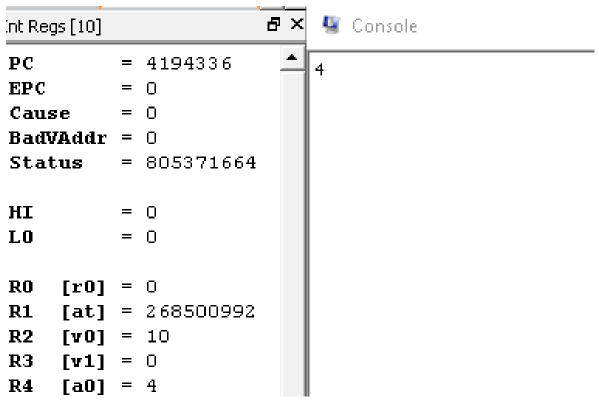
Task 5:

Modify the program to count elements in a null-terminated array of words.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

#Task 5

.data
entry: .word 5, 12, 7, 2    # Define the array of words
       .word 0              # Null terminator for the array
entryA: .word entry         # Pointer to the address of entry

.text
main:
    add $s0, $ra, $0         # Save return address in $s0
    lw $a0, entryA($0)       # Load address of entry into $a0

    jal length               # Jump to length subroutine and link
    move $a0, $v0            # Move length of array into $a0 for printing
    li $v0, 1
    syscall

    add $ra, $s0, $0         # Restore the return address from $s0
fini:
    jr $ra
length:
    add $v0, $0, $0          # Initialize length counter in $v0 to 0
length_loop:
    lw $t0, 0($a0)           # Load word from array into $t0
    beq $t0, $0, length_end  # If word is zero, end loop
    addi $v0, $v0, 1         # Increment length counter
    addi $a0, $a0, 4         # Move to next word (4 bytes)
    j length_loop            # Repeat loop
length_end:
    jr $ra                   # Return to caller with length in $v0
```

Output:



```
PC         = 4194336
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 805371664

HI         = 0
LO         = 0

R0   [r0]  = 0
R1   [at]  = 268500992
R2   [v0]  = 10
R3   [v1]  = 0
R4   [a0]  = 4
```

```
4
```