

# INFORME DE PROGRAMA

1.- El siguiente programa lo que hace es devolvernos el tamaño de una cadena de caracteres o un "string".

## REVISIÓN POR PARTES:

```
length.asm*
15  _start:                # execution starts here
16      la $t2, str         # t2 points to the string
17      li $t1, 0           # t1 holds the count
18  nextCh:
19      lb $t0, ($t2)        # get a byte from string
20      beqz $t0, strEnd     # zero means end of string
21      add $t1, $t1, 1      # increment count
22      add $t2, $t2, 1      # move pointer one character
23      j nextCh            # go round the loop again
24
25  strEnd:
26      la $a0, ans         # system call to print
27      li $v0, 4           # out a message
28      syscall
29
30      move $a0, $t1        # system call to print
31      li $v0, 1           # out the length worked out
32      syscall
33
34      la $a0, endl        # system call to print
35      li $v0, 4           # out a newline
36      syscall
37
38      li $v0, 10          # au revoir...
39      syscall
```

1.- Almacenamos un puntero en \$t2 que apunte hacia la primera posición de mi *str*. Para almacenar el conteo de caracteres a \$t1 le damos el valor de 0.

```
length.asm*
15  _start:                # execution starts here
16      la $t2, str         # t2 points to the string
17      li $t1, 0           # t1 holds the count
```

2.- Para saber el tamaño se hace un bucle. \$t0 almacena un byte de mi *string* y se verifica si es igual que 0; de ser así que finalice el bucle y vaya a *strEnd*, de no ser así que sume +1 al contador de \$t1 y también aumentar el puntero para que apunte al siguiente byte. Al finalizar todo esto se hace un salto a mi bucle de nombre *nextCh*.

```
18  nextCh:
19      lb $t0, ($t2)        # get a byte from string
20      beqz $t0, strEnd     # zero means end of string
21      add $t1, $t1, 1      # increment count
22      add $t2, $t2, 1      # move pointer one character
23      j nextCh            # go round the loop again
24
25  strEnd:
```

3.- Para imprimir los resultados primero en \$a0 le mandamos *ans* que en este caso imprime un *string* "Length is " y llamamos a la función 4 (imprimir cadena) de \$v0.

```
26      la $a0, ans         # system call to print
27      li $v0, 4           # out a message
28      syscall
```

4.- Una vez colocado la frase "Length is " y haber llamado a la función de imprimir, ahora modificamos \$a0 y que tome el valor de \$t1 que vendría a ser nuestro contador que almacena la cantidad de caracteres de *str*. Usamos la función 1(imprimir entero) en \$v0.

```
30      move $a0,$t1          # system call to print
31      li $v0,1              # out the length worked out
32      syscall
```

5.- Ahora hacemos los mismo que el paso 3, solo que en \$a0 se le da *endl* que en este caso solo da un salto de línea.

```
34      la $a0,endl          # system call to print
35      li $v0,4              # out a newline
36      syscall
```

6.- Llamamos a la función 10 que sale del programa.

```
38      li $v0,10
39      syscall               # au revoir...
```

### FINALIZANADO EL PROGRAMA:

Le asignamos valores a *str*, *ans* y *endl*.

- *str* = hello world
- *ans* = Length is
- *endl* = \n (salto de línea)

Al ejecutar el programa toma los valores dados y al final imprime es largo de la cadena almacenada en *str*.

The screenshot shows the Mars MIPS simulator interface. The assembly code window displays the following code:

```
46      .data
47          str: .asciiz "hello world"
48          ans: .asciiz "Length is "
49          endl: .asciiz "\n"
50
51      ##
52      ## end of file length.a
53
```

Below the code window, the status bar shows "Line: 26 Column: 8" and a checked box for "Show Line Numbers".

The "Mars Messages" window shows the output of the program:

```
Length is 11
-- program is finished running --
```

A "Clear" button is visible in the bottom left corner of the messages window.