PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Transposición de matriz de caracteres en MIPS

Claudio Pastenes Necul
Luciano Portales Jasme
2021

Tabla de contenido

1. Introducción

Los MIPS son una familia de procesadores de arquitectura RISC, que fueron desarrollados por MIPS Technologies. Estos procesadores fueron integrados en varios sistemas, dispositivos y consolas de videojuegos.

Actualmente estos microprocesadores fueron descontinuados debido al avance de las tecnologías competidoras. Sin embargo, actualmente ha encontrado su nicho siendo emulado para el aprendizaje y uso del lenguaje ensamblador.

El lenguaje ensamblador es un lenguaje de bajo nivel, el cual, al contrario de los de alto nivel, se basa en el manejo de los registros de procesador teniendo un mayor control sobre los datos, pero con poca flexibilidad.

Durante este texto analizaremos un problema de manejo de registros y valores usando el lenguaje ensamblador de MIPS emulado por la aplicación Simula3MS, para, de esta manera, demostrar nuestro dominio sobre el lenguaje, manejo de registros y arquitectura del hardware.

2. Problema

Se debe realizar un programa que cumpla con las siguientes funcionalidades:

- Leer una cadena de caracteres y valide que contenga 9, 16 o 25 bytes. Si el largo no corresponde a dichos tamaños enviar un mensaje de error.
- Si el tamaño es correcto, tratar la cadena como si fuera una matriz cuadrada (de 3x3, 4x4 o 5x5), y generar una nueva cadena que contenga la matriz transpuesta.
- Desplegar el resultado por pantalla

Ejemplo:

- Cadena leída: ABCDEFGHIJKLMNOP
- Tratada como matriz:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

- Transpuesta:

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

- Cadena resultante para imprimir: AEIMBFJNCGKODHLP

3. Resolución del Problema

3.1. Análisis

1. Se nos pide que carguemos una cadena de caracteres, y que esta contenga 9, 16 0 25 bytes.

Los caracteres en MIPS están en código ASCII, por lo cual cada uno pesa 1 byte, por lo tanto, lo que en realidad se nos pide que leamos una cadena de largo 9, 16 o 25

2. Se nos pide una matriz cuadrada

Los largos de las cadenas que se nos piden son cuadrados, lo que significa que el número de filas y columnas al crear la matriz son iguales. Esta característica puede volver más amena la creación de un algoritmo

3. Matriz transpuesta

La matriz transpuesta nos indica que, al leer los datos de la matriz original, en lugar de leer M_i [Fila, Columna], se lee M_i [Columna, Fila]. Si lo traspasamos a un arreglo, entonces seria A_i [i]; A_{i+1} [i+$\sqrt{largo}$]

3.2. Algoritmo

Siguiendo los puntos del análisis concluimos que el problema puede ser resuelto con un algoritmo de un ciclo el cual se muestra en la Ilustración 1.

Siguiendo este diagrama:

1. Entra la cadena
2. Se crea una referencia a una cadena nueva B
3. Se inicializan los valores i, j, k en 0, y el valor largo como el largo de la cadena ingresada
4. En caso de que largo sea 9, S será 3; en caso de largo= 16, S=4; en caso de largo=25, S=5.; en caso de ninguno de los anteriores el proceso se termina
5. Si i es diferente a “largo”, entonces se pasa al paso 6, de lo contrario se termina el programa
6. Si j=S, entonces el valor de J se cambia a 0 y el valor de k aumenta en uno, de lo contrario se salta hasta el paso 7
7. El carácter en la posición [j * s + k] de la cadena b adquiere el valor del carácter en la posición [i] de la cadena “cadena”. Luego se regresa al Paso 5.

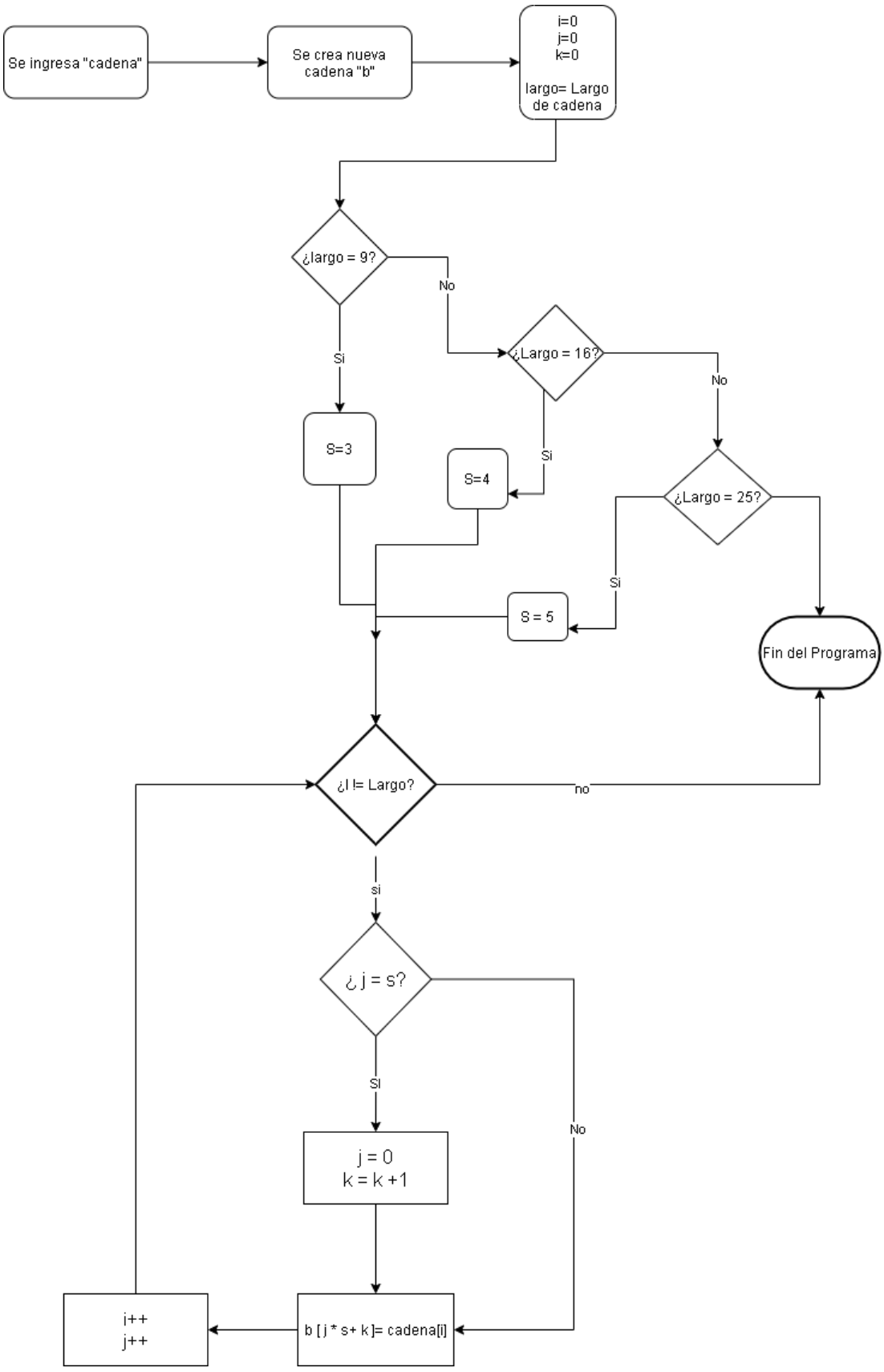
Se ingresa "cadena"
Se crea nueva cadena "b"
i=0
j=0
k=0
largo= Largo de cadena
¿largo = 9?
No
Si
¿Largo = 16?
No
Si
S=3
S=4
¿Largo = 25?
Si
S = 5
Fin del Programa
¿I != Largo?
no
si
¿ j = s?
SI
No
j = 0
k = k +1
j++
j++
b [j * s+ k]= cadena[i]

Ilustración 1 Algoritmo para Stirng traspuesta

3.3. Código en alto nivel

Para volver más amena el traspaso de un algoritmo conceptual a lenguaje ensamblador, decidimos, primero, traspasarlo a un lenguaje más familiar y conocido. El Lenguaje que elegimos s para esto es C, y el código en el cual adaptamos el diagrama se muestra en la Ilustración 2

```c
#include <stdio.h>

int main() {
    char a[] = "abcdefghijklmnop";
    char b[] = "
    int largo = 16;
    int i = 0;
    int j = 0;
    int k = 0;

    while (i != largo) {

        if (j == 4) {
            j = 0;
            k+=1;
        }


        b[j * 4 + k] = a[i];

        i++;
        j++;
    }


    printf("%s", b);

    return 0;
}
```

Ilustración 2 String transpuesta en C

3.4. Código en ensamblador

Desde aquí se deja adjunto el código de ensamblador en Simula3MS:

```
1. .data
2.     cadenatranspuesta: .space 25
3.     cadena: .space 25
4.     largo: .word 0
5.     msgError: .asciiz "Largo de cadena invalida"
6.
7. .text
8. .globl main
9. main:
10.   # registros para condiciones
11.      li $t0,9
12.       li $t1,16
13.       li $t2,25
14.
15.   #cargar variables en registros
16.      la $a0,cadena
17.
18.   # leer string
19.      addi $v0,$zero,8
20.       syscall
21.
22.   #-------------------- validación del string -------------------------------#
23.      bne $a1,$t0,sino             # condicion largo de string leido distinto de 9
24.   si:
25.       addi $t0,$zero,3
26.       j finsi
27.   sino:
28.       bne $a1,$t1,sino2            # condicion largo de string leido distinto de 16
29.   si2:
30.       addi $t0,$zero,4
31.       j finsi
32.   sino2:
33.       bne $a1,$t2,sino3           # condicion largo de string leido distinto de 25
34.   si3:
35.       addi $t0,$zero,5
36.       j finsi
37.   sino3:
38.       la $a0,msgError                # se carga en registro a0 mensaje de error
39.      addi $v0,$zero,4
40.       syscall                                #se muestra mensaje de error
41.      j finprograma                  # salto a fin del programa
42.   finsi:
43.       add $t1,$zero,$a1          # largo del string leido
44.      addi $t2,$zero,0            # i = 0
45.      addi $t3,$zero,0            # j = 0
46.      addi $t4,$zero,0            # k = 0
47.
```

```
48.       la $t5,cadena
49.       la $t6,cadenatranspuesta
50.
51.   # --------------- algoritmo matriz inversa del string ----------------- #
52.   mientras:
53.       beq $a1,$t2,finmientras            # mientras a1 <> t2 se sigue con el ciclo
54.
55.       beq $t0,$t3,simientras
56.       j finsimientras
57.       simientras:
58.           addi $t3,$zero,0           # j = 0
59.          addi $t4,$t4,1             # k++
60.       finsimientras:
61.
62.       # en base a la fórmula filas * n columnas + columna
63.
64.       mult $t3,$t0      # se cambia las variables de fila por columna y columna por
    filas y se hace la multiplicación
65.       mflo $t8                      # se obtiene el valor de la multiplicación
66.       add $t8,$t8,$t4               # se suma la fila j del recorrido con la
    multiplicación entre columna * n columnas
67.
68.       lb $t7,($t5)                # almacena en registro t7 la letra de la cadena
69.       add $t6,$t6,$t8
70.       sb $t7,($t6)              # carga la letra del registro t7 en puntero a letra
71.       sub $t6,$t6,$t8
72.
73.       addi $t3,$t3,1           # se incrementa en una unidad
74.       addi $t5,$t5,1          # se pasa al siguiente carácter de la cadena
75.       addi $t2,$t2,1          # se aumenta el valor de i en una unidad
76.
77.       j mientras
78.   finmientras:
79.
80.   #------------------- se imprime la nueva cadena generada -------------------------#
81.       la $a0,cadenatranspuesta
82.       li $v0,4
83.       syscall
84.
85.   #------------------------ fin  del programa ----------------------------------#
86.   finprograma:
87.       addi $v0,$zero,10
88.       syscall
```

4. Conclusión

En conclusión, el desarrollo de prácticas en lenguaje ensamblador, permitieron comprender en mayor profundidad el cómo funcionan los registros, el como son manipularlos y como también utilizar los llamados al sistema, así también se expande el conocimiento para no quedarse solo con el desarrollo de programas en un lenguaje de alto nivel.

En cuanto al desarrollo del problema, se aplicaron los métodos vistos en clases, más específicamente, la fórmula para manipular una cadena como matriz, como también se desarrolló el problema utilizando diagramas de flujos y aplicándolo en un lenguaje de alto nivel, para su posterior implementación en el lenguaje ensamblador; siendo el aplicarlo en este último el desafío mayor, pues se tuvieron que tomar varias cuestiones en cuenta, como los saltos condicionales e incondicionales, los manejos de registros y por último, los llamados al sistema.