Arquitecturas y Organización de Computadoras I
2° Cuatrimestre

TP N° 7 – Programación en lenguaje ensamblador MIPS con MIPSX

Facultad de Informática

Objetivo: Comprender la estructura de un programa en lenguaje ensamblador MIPS básico. Introducción a la interfaz `mipsx` y al conjunto de instrucción MIPS.

Recursos y Bibliografía:

Arq. MIPS Vol I, II and III.

Programa `mipsx` desarrollado por la cátedra.

1. Cree una función que calcule el factorial de un número, utilizando la definición recursiva, y respetando la convención de llamada a procedimiento.

```
factorial(0) = 1
factorial(N) = N * factorial(N-1)
```

2. Traduzca el siguiente segmento de código de alto nivel a ensamblador de MIPS, sabiendo que todas las variables están almacenadas como enteros. Recuerde cargar el valor de aquellas variables e inicializar los registros que se usen como fuente de una operación, y almacenar los valores resultantes.

```
int a = 10;
int b = 4000;
int c;
c = b * a;
for (i=0; i<c; i++) {
      a = a + 1;
}
```

3. ¿Cuántas instrucciones ensamblador ocupa el programa desarrollado en el punto 2? ¿Cuántas instrucciones máquina? ¿Cuántas pseudo instrucciones hay? Si un procesador MIPS (sin pipeline) tiene un reloj de 2Khz ¿Cuánto tiempo tardaría en ejecutarse el programa?

4. Cree una función que tenga como parámetro un puntero (un entero que contiene la dirección de una variable) a una cadena almacenada como asciiz y reemplace las letras mayúsculas por minúsculas (la cadena puede contener caracteres que no sean letras). Realice el reemplazo sobre la cadena original. Respete la convención de llamada a procedimiento.

5. Dado el siguiente programa escrito en código ensamblador de mips, suponiendo que el compilador traduce cada pseudo instrucción a 2 instrucciones máquina, y la dirección de la primera instrucción es 0x40090:

a. ¿Cuántos bytes ocupa el segmento de texto?
b. ¿Cuál es la dirección de la última instrucción?
c. ¿Cuál es la dirección del último byte del segmento de texto?

```
li $t0, 176
li $t7, -186000
lw $t2, memoria($t0)
lb $t2,12($s0)
la $t2, memoria
```

```
and $t3, $t4, $t4
lh $t1, memoria+2($t0)
addi $t2, $t2, 16
```
NOTA: la dirección del segmento de datos (etiqueta memoria) es 0x41000