Arquitecturas y Organización de Computadoras I
2° Cuatrimestre

TP N° 8 – Programación en lenguaje ensamblador MIPS con MIPSX

Facultad de Informática

Objetivo: Comprender la estructura de un programa en lenguaje ensamblador MIPS, convención de llamada a procedimientos y funciones.

Recursos y Bibliografía:

Arq. MIPS Vol I, II and III.
Programa mipsx desarrollado por la cátedra.
Apunte MIPS, sección 6.

1. ¿Qué fallos comete el siguiente código de función en cuanto a la convención de uso de registros y llamada a procedimiento y funciones?

```
fun:
        addiu   $sp,$sp,-32
        sw      $ra,28($sp)
        sw      $fp,24($sp)
        move    $fp,$sp
        li      $s0,1
        li      $t1,2
        jal     fun
        addi    $k0, $v0, 3
        add     $v0, $t1, $k0
        add     $v0, $s1, $k0
        move    $sp,$fp
        lw      $ra,28($sp)
        lw      $fp,24($sp)
        addiu   $sp,$sp,32
        jr      $ra
```

2. Traduzca al lenguaje ensamblador el programa en C que se encuentra debajo (respete la convención de llamada a procedimientos). Considere que char ocupa un byte e int una palabra. Verifique su correcto funcionamiento en *mipsx*.
   a. ¿por qué las variables **x** y **n** de pow no son las mismas que las variables globales **x** y **n** definidas antes de **main**?
   b. Indique cuales son las direcciones efectivas de **x** y **n** global, y de las variables **i** y **pow** en **pow()**.
   c. ¿Cuál es la dirección del último byte del segmento de texto? ¿Cuantas pseudo instrucciones tiene el programa?

```
int pow(char x, char n)
{
        int i;
        int pow = 1;

        for (i=0; i<n; i++)
                pow = pow * x;

        return pow;
}
```

```
char x = 30;
char n = 55;
int resultado = 0;

void main(void)
{
        resultado = pow(x, n);
}
```

Ejercicios de repaso:

3. Traduzca la siguiente estructura de un lenguaje de alto nivel a directivas de ensamblador de mips. Respete el orden de las variables. Tenga en cuenta que el string contiene un carácter con valor cero para indicar el final de la cadena.

```
struct {
      string user = "Peperina”;
      byte edad = 0x22;
      half puntaje = 16000;
      float altura = 1.76;
      half contador = -5;
      int key = 0xFA093319;
} usuario;
```

4. Represente un vuelco de memoria en hexadecimal (es decir, los contenidos de la memoria en hexadecimal) del segmento de datos que contiene a la estructura. Asuma que el segmento de datos comienza en la dirección **0x4000**.

5. ¿Cuál será el contenido de los registros **t0**, **t1**, **t2** y **t3**, luego de ejecutar el siguiente segmento de código? ¿Cuáles son las direcciones efectivas para cada una de las instrucciones de carga?

```
li    $t0, -1
lb    $t1, contador($t0)
lhu   $t2, contador
lw    $t3, user+4
```