Arquitecturas y Organización de Computadoras I
2° Cuatrimestre

TP N° 9 – 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) Traduzca el siguiente programa en lenguaje de alto nivel a ensamblador de MIPS, reserve espacio en la pila para las variables locales.

```
int fact(int n)
{
	int res;
	if(n!=0){
		res = fact(n-1) * n;
	} else {
		res = 1;
	}
	return res;
}

int resultado;
void main()
{
	int resultado;
	resultado = fact(3);
}
```

2) Al ejecutar el programa del punto anterior en MIPSX ¿En qué direcciones de memoria se almacenan las distintas instancias de la variable **`res`**? ¿Qué valores toman los registros `sp` y `fp`?
3) Desarrollar un programa que detecte si el contenido en la etiqueta img es una imagen con formato PNG. Si el contenido en img es una imagen PNG el programa debe escribir la letra 'P' en la dirección de memoria definida con la etiqueta formato. Si no es PNG entonces se debe escribir la letra 'X' en la dirección de memoria definida con la etiqueta formato.

   El formato PNG está definido por una cabecera de 8 bytes:

   - El primer byte tiene que ser el valor 0x89
   - Los 3 bytes siguientes tiene que ser el texto ascii PNG
   - Los 4 bytes siguientes tienen que ser los valores (aquí están expresados en base 16): 0D0A1A0A

   Si los primeros 8 bytes de una imagen coinciden con la definición de la cabecera PNG entonces se considera a los datos una imagen PNG.
   Ayuda: Una posibilidad es definir una cabecera PNG real en memoria, y luego comparar la cabecera de img con la cabecera real, byte a byte.

```
.data
memoria:
img:         .byte # bytes de una posible imagen en formato PNG
formato:     .space 1   # resultado de la detección del formato

        .text
        .globl main
        .globl __start

__start:
main:
             # COMPLETAR

# retorna al SO
        move $a0, $0
        li      $v0, 4001
        syscall
```

4) Desarrolle una función que tenga como parámetro de entrada la dirección a un arreglo de enteros, y la cantidad de enteros dentro del arreglo, y retorne el mayor valor del arreglo.