Laboratorio de Fundamentos de Microprocesadores
Curso 2018-2019

Práctica 3:
Programación en Ensamblador

Tutorial de uso del simulador del procesador MIPS y ejemplo de programa en ensamblador

Durante este tutorial se explicará el funcionamiento del simulador del procesador MIPS, que se puede descargar de *Moodle* (material de la unidad 3). El ejemplo propuesto realizará en ensamblador un programa que realice la misma función que el siguiente código C:

```
int a = 20;
int b = 10;
int c;

int main()
{

    if(a < b)
       c = b;
    else
       c = a;

   while(1);
}
```

Ejercicio 1. Desensamblar

En el ejercicio 1 se proporciona un programa (Desensamblar.txt) compatible con la arquitectura MIPS descrita en las clases de teoría. En este ejercicio se debe desensamblar este programa, es decir, convertir las instrucciones en notación hexadecimal a notación ensamblador, para poder comprender la funcionalidad del programa.

El programa suministrado contiene dos segmentos, código y datos. Algunas instrucciones hacen referencia a posiciones del segmento de datos.

Objetivo

Desensamblar el programa propuesto y comprender su funcionamiento, para ello, se sugiere que con cada instrucción ensamblador, se incluya un comentario que explique la funcionalida que genera. Comprobar con el simulador Mars de MIPS, si el programa desensamblado coincide con el programa en código máquina. Para ello se debe configurar la memoria como **<<Compacta con dirección de inicio del segmento text en x0>>.**

Ejercicio 2. Llamada a función y paso de parámetros

En este ejercicio se debe escribir en ensamblador un programa que tenga funcionalidad equivalente al programa en C propuesto y verificar su correcto funcionamiento. Para ello, debe entender cómo codificar una llamada a función en ensamblador y su paso de parámetros.

El paso de parámetros y el retorno realícelo utilizando la pila.

Se adjunta un código equivalente en *C* del ejercicio:

```
int X = 10;
int Y = 4;
int R;

int calculaSumaMult (int a, int b)
{
    return (a+b)*2;
}

int main()
{
        R = calculaSumaMult(X,Y);

        while(1);
}
```

Tenga en cuenta que la instrucción principal del programa C incluye múltiples operaciones:

- Lectura de la variable X de memoria
- Lectura de la variable Y de memoria
- Escritura en la pila las dos variables leídas
- Llamada a función
- Recuperación del retorno guardado en pila
- Escritura del retorno en la variable R de memoria

A su vez, la instrucción principal de la función *calculaSumaMult* incluye las siguientes operaciones:

- Recuperación de los parámetros leyendo la pila
- Suma de los parámetros
- Multiplicación por 2 de la suma anterior
- Escritura del resultado en pila
- Retorno a main

Objetivo

Aprender a realizar llamadas a funciones y gestionar el paso de parámetros y retornos. Para la realización de este ejercicio se debe configurar la memoria como **<<Compacta con dirección de inicio del segmento text en x0>>.**

Ejercicio 3. Compilación de código C

En este ejercicio se debe escribir en ensamblador un programa que tenga funcionalidad equivalente al programa en C propuesto y verificar su correcto funcionamiento. Para ello, debe entender cómo codificar un bucle *for* en ensamblador y cómo realizar lecturas y escrituras de un vector.

```
#define N 10
int A[N]={2,2,4,6,5,6,7,8,9,10};
int B[N]={-1,-5,4,10,1,-2,5,10,-10,0};
int C[N];

int main()
{
      int i;
      for(i=0; i<N; i++)
          C[i]=A[i]+B[i]*4;
      while(1);
}
```

Objetivo

Comprender el problema propuesto, plantear e implementar una solución para dicho problema y verificar su correcto funcionamiento. Para la realización de este ejercicio se debe configurar la memoria como **<<Compacta con dirección de inicio del segmento text en x0>>**. Para ello vaya a “Settings -> Memory Configuration…” y seleccione “Compact, Text at Address 0”. Repita estos pasos para todos los ejercicios de esta práctica.

Nota: el valor de N hay que leerlo desde memoria (el *define* se convierte a una variable normal de memoria). Utilice las etiquetas A, B, C y N para estos cuatro elementos en memoria de datos.

Otros ejercicios.

- ¿Qué cambios serían necesarios en el ejercicio 2 para realizar el paso de parámetros y retorno con los registros $a0-$a3 y $v0-$v1 respectivamente?
- ¿Qué cambios serían necesarios en el ejercicio 3, tanto en código C como en ensamblador, para implementar la misma funcionalidad utilizando un bucle while?
- ¿Cómo se accedería a la tercera posición de un vector cuya dirección base está referenciada por la etiqueta “vector”?