Laboratorio de Estructura de Computadores
Curso 2016-2017

Práctica 4:
Programación en Ensamblador

La entrega estará compuesta de un único fichero que incluya:

- Todos los ficheros ensamblador solicitados.
- Un fichero llamado “autores.txt” que contenga los nombres completos e identificadores de la pareja.

Todos estos ficheros deben entregarse en un único fichero comprimido.

Los ejercicios deberán subirse a la actividad correspondiente de Moodle, identificada por el número de práctica (ej. La práctica 2 se entrega en la tarea «Entrega Práctica 2»). Sólo es necesario que suba los ficheros un miembro de la pareja. En caso de que ambos miembros de la pareja suban un fichero a Moodle, se corregirá sólo uno.

Todos los ficheros deberán contener en la cabecera los nombres de los autores y el identificador de la pareja. Asimismo, se recuerda que los ficheros deberán estar correctamente comentados. Dado que no se entrega una memoria adicional, los comentarios de los ficheros deben ser claros y concisos. No atenerse a estas normas implicará una penalización en la nota.

El límite de entrega de esta práctica será el día anterior al comienzo de la siguiente práctica. Las parejas que no hayan entregado en plazo y forma serán calificadas con un cero en esta práctica.

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. Compilación de código C (3 puntos)

En este ejercicio se debe escribir en ensamblador un programa que tenga funcionalidad equivalente al programa en C propuesto y verificar su correcto funcionamiento.

```
#define N 10 //Deberá funcionar para valores de N entre 5 y 20
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 1: reserve con la directiva .space, para el vector C tanta memoria como haga falta para que su tamaño pueda ser de hasta 20 elementos.

Nota 2: el valor de N hay que leerlo desde memoria. Utilice las etiquetas A, B, C y N para estos cuatro elementos en memoria de datos. Tenga cuidado de no sobrescribir ningún valor de memoria indebido cuando rellene el vector C.

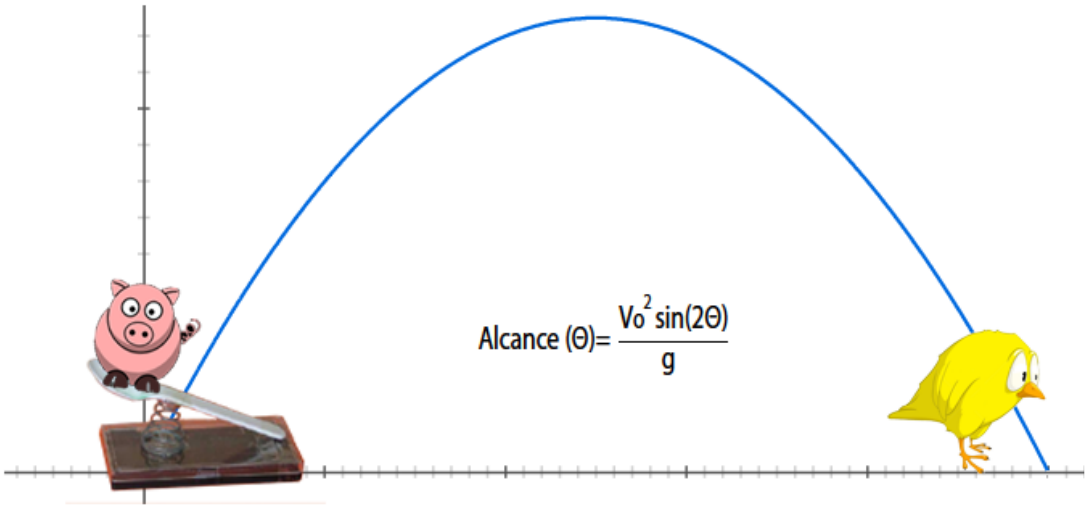
Entrega

Se debe entregar el fichero "Vectores.asm". Debido a la dificultad de lectura y comprensión del código ensamblador, es especialmente importante en esta práctica comentar el código de forma que se entienda fácilmente su funcionalidad.

Ejercicio 2. Alcance en un movimiento parabólico (4 puntos)

Una empresa de videojuegos os ha encargado realizar un juego llamado “Gorrinos enojados: La revancha” en versión máquina recreativa con procesador MIPS.

Vuestro analista de sistemas os ha pedido que, como versión previa, calculéis el alcance (distancia final en el eje x), dependiendo del ángulo de impulso, de un cerdo que se impulsa con una catapulta. Para hacer el juego más sencillo, se considera que todos los cerdos se impulsan con la misma velocidad inicial, 50 m/s, y se considera que la aceleración de la gravedad es 10 m/s^2. La fórmula del alcance está en la figura. El resultado se dará como un entero, redondeando al entero más cercano.



Dado que el cálculo de funciones trigonométricas es complejo, se debe precalcular el resultado de las mismas. Además, gracias a que la velocidad y la aceleración son conocidas, en vez de precalcular simplemente la función trigonométrica, se puede precalcular directamente la fórmula completa del alcance. Así, todos los posibles alcances del cerdo pueden estar almacenados en memoria, en función de su ángulo de impulso. De esa forma, si se quieren calcular alcances para ángulos enteros entre 0 y 45 grados inclusives, únicamente hace falta precalcular 46 valores y el programa accederá al valor adecuado.

El analista os pide que **necesariamente** se desarrolle una función llamada ***calculaAlcance***, la cual accederá a la posición donde se encuentre el alcance precalculado correspondiente al ángulo pedido, y lo devolverá como resultado. Dicha función recibe su argumento (ángulo en grados) **a través de la pila del sistema** y devuelve el resultado (alcance en metros) en la misma pila. Además, el programa debe inicializar el puntero a pila (*$sp*) al valor 0x4000.

El ángulo del que se calculará su alcance se leerá de la posición de memoria indicada por la etiqueta *angulo* y el resultado se guardará en la posición de memoria indicada por la etiqueta *resultado*.

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

```
int angulo = 10;
int resultado;
const int lut[] = { 0, 9, 17, 26, 35, … }

int calculaAlcance (int theta)
{
    return lut[theta];
}

int main()
{
        int res;

        res = calculaAlcance(angulo);

        while(1);
}
```

Se pide que el programa permita calcular el alcance del cerdo para ángulos enteros entre 0 y 45 grados. Se facilita un plantilla, con los valores correspondientes a los ángulos [0-40]. **Deben completarse las 5 posiciones restantes [41-45].**

Objetivo

Aprender a utilizar el simulador del MIPS e implementar un sencillo programa en ensamblador que utiliza la pila. Para la realización de este ejercicio se debe configurar la memoria como **<<Compacta con dirección de inicio del segmento text en x0>>.**

Entrega

Se debe entregar el fichero "Alcance.asm". Debido a la dificultad de lectura y comprensión del código ensamblador, es especialmente importante en esta práctica comentar el código de forma que se entienda fácilmente su funcionalidad.

Ejercicio 3. Desensamblar (3 puntos)

En el ejercicio 3 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, comprender su funcionamiento, y calcular el resultado de cada instrucción. Si se quiere simular el programa desensamblado en el simulador, se debe configurar la memoria como **<<Compacta con dirección de inicio del segmento text en x0>>.**

Entrega

Se debe entregar el fichero "Desensamblar.asm". Este fichero debe contener el programa convertido a lenguaje ensamblador. Cada instrucción debe ir comentada el resultado que genera. Por ejemplo:

addi $1, $0, 5 # R1 <= R0 + 5 = 5