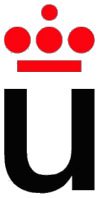

Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INTELIGENCIA ARTIFICIAL

FUNDAMENTOS DE ARQUITECTURA DE COMPUTADORES

Proyecto 2: programación en lenguaje ensamblador

Enunciado: grupo 3

Curso 2022-23

EJERCICIO 1: EXPRESIÓN CON ENTEROS

Realizar un programa en ensamblador que evalúe la siguiente expresión con enteros:

$$w = x + y^2 + 3 \cdot z$$

A continuación se muestra una posible solución en C:

```
// Variables estáticas que deben existir en memoria
int x, y, z, w;

int main (void) {
     // Asignar valor a x, y y z
     w = x + y * y + 3 * z;
     // Escribir valor de w
     return 0;
}
```

Escribir la solución en un fichero llamado **expresion_enteros.asm**.

EJERCICIO 2: ESTRUCTURAS DE CONTROL

Realizar un programa en ensamblador que calcule la suma de los **n** primeros elementos (**n** > 0) de la serie cuyo término general se calcula así:

$$a_i = \begin{cases} 1 & n = 1 \\ 3 \cdot a_{i-1} - i & n\ par \\ 3 \cdot a_{i-1} - i + 1 & n\ impar\ y\ mayor\ que\ 1 \end{cases}$$

Se puede partir de la siguiente solución en C:

```
// Variables estáticas que deben existir en memoria
int suma, n;
// variables que pueden residir exclusivamente en registros
register int i, ai, ai_1;

int main (void) {
     // Asignar valor a n
     ai_1 = 1;
     suma = 1;
     i = 2;
     while (i <= n) {
          ai = 3 * ai_1 - i;
          if (i % 2 == 1)
               ai = ai + 1;
          suma = suma + ai;
          ai_1 = ai;
          i = i + 1;
     }
     // Escribir valor de suma
     return 0;
}
```

Escribir la solución en un fichero llamado **suma_serie.asm**.

EJERCICIO 3: VECTORES

Realizar un programa en ensamblador que, dado un vector de 16 elementos de tipo entero, cuente cuántos de ellos tienen un valor igual a 0.

Dado que introducir el vector por teclado es una tarea tediosa que habría que repetir en cada prueba, se permite definir el vector mediante directivas **.word**. A modo de ejemplo, se muestra a continuación cómo crear un vector con 8 elementos de tipo entero (siempre dentro de la sección **.data**):

```
vector:   .word     10,20,30,40
          .word     50,60,70,80
```

Todos los elementos se ubicarán a partir de una cierta dirección que equivaldrá al nombre **vector**, y cada uno ocupará 4 bytes.

Se podrá partir de la siguiente solución en C:

```
// variables estáticas que deben existir en memoria
int x[16]={ … }
int cuenta, suma;
// variables que pueden residir exclusivamente en registros
register int i, tmp;

int main (void) {
     cuenta = 0;
     i = 0;
     do {
          tmp = x[i];
          if (tmp == 0)
               cuenta = cuenta + 1;
          i = i + 1;
     } while (i == 16);
     // Escribir valor de cuenta
     return 0;
}
```

Escribir la solución en un fichero llamado **procesar_vector.asm**.

EJERCICIO 4: CADENAS DE CARACTERES

Realizar un programa en ensamblador que, dadas dos cadenas de caracteres las concatene, quedando primero el contenido de la primera cadena y después (sin carácter nulo entre medias) el contenido de la segunda (copiando el carácter nulo al final del todo). La cadena resultante quedará grabada en una tercera cadena, que tendrá espacio de sobra para que quepa la concatenación de ambas.

Se podrá partir de la siguiente solución en C:

```
// variables estáticas que deben existir en memoria
char cadena1[32], cadena2[32], cadena3[64];
// Esta será la longitud máxima de las tres cadenas
// variables que pueden residir exclusivamente en registros
register int i, j, tmp;
```

```
int main (void) {
     // Rellenar cadenas de entrada
     i = 0;
     for (;;) {
          tmp = cadena1[i];
          if (tmp == '\0') break;
          cadena3[i] = tmp;
          i = i + 1;
     }
     j = 0;
     for (;;) {
          tmp = cadena2[j];
          if (tmp == '\0') break;
          cadena3[i] = tmp;
          i = i + 1;
          j = j + 1;
     }
     cadena3[i] = '\0';
     // Escribir cadena resultante
     return 0;
}
```

Escribir la solución en un fichero llamado **procesar_cadena.asm**.

EJERCICIO 5: COMA FLOTANTE

En este ejercicio se programará la evaluación de una expresión equivalente a la del ejercicio 1, pero ahora empleando datos en coma flotante de precisión simple.

Es decir, que se pide realizar un programa en ensamblador que evalúe la siguiente expresión:

$$w = x + y^2 + 3 \cdot z$$

A continuación se muestra una posible solución en C:

```
// Variables estáticas que deben existir en memoria
float x, y, z, w;

int main (void) {
     // Asignar valor a x, y y z
     w = x + y * y + 3.0 * z;
     // Escribir valor de w
     return 0;
}
```

Escribir la solución en un fichero llamado **expresion_float.asm**.

EJERCICIO ADICIONAL

Realizar un programa en ensamblador de MIPS32 que calcule el inverso de la raíz cuadrada de un número positivo **Z** mediante el método de Newton-Raphson.

El método parte de la función siguiente:

$$f(x) = \frac{1}{x^2} - Z$$

cuya solución es precisamente

$$x = \frac{1}{\sqrt{Z}}$$

La ecuación de recurrencia es:

$$x_{i+1} = \frac{x_i \cdot (3 - Z \cdot {x_i}^2)}{2}$$

Se puede partir de la siguiente solución en C:

```
// Variables estáticas que deben existir en memoria
float z, semilla, tol, invraiz2;
// variables que pueden residir exclusivamente en registros
register float xi, xi1;
int main (void) {
     // Introducir z y semilla
     if (z < 0.0) // z debe ser positivo
          return -1;
     xi1 = semilla;
     tol = 1e-5;    // valor recomendado para la tolerancia
     do {
          xi = xi1;
          xi1 = (xi * (3 – z * xi * xi)) / 2;
     ] while (fabs(xi1-xi) >= tol);
     invraiz2 = xi1;
     // Escribir resultado
     return 0;
}
```

El valor recomendado para la semilla depende del valor de **Z**, mientras que la tolerancia se puede prefijar de antemano o introducir por consola.

Escribir la solución en un fichero llamado **Newton_Raphson.asm**.

Para que las pruebas funcionen correctamente es preciso elegir una semilla suficientemente próxima a la solución correcta. Para ello se sugiere representar gráficamente la función **f(x)** y elegir como semilla un punto que garantice la convergencia del método. Para representar la función gráficamente se pueden emplear herramientas como la calculadora gráfica de **Geogebra** (https://www.geogebra.org/calculator), sustituyendo **Z** en la función por el valor del dato del cual se quiere calcular el inverso de su raíz cuadrada.