# PRÁCTICA 2:
*Implementación y optimización de un cálculo en ensamblador DLX*

## 1. Descripción de la práctica

El objetivo de la práctica es el desarrollo y optimización de un código que realice el siguiente cálculo:

a) Calcular una secuencia de números "estilo 3n+1" partiendo de un valor inicial dado (`valor_inicial` *word* con rango de 1 a 100, incluidos), siendo que si su valor es, por ejemplo 3, la secuencia obtenida sería 3-10-5-16-8-4-2-1 (**ES OBLIGATORIO USAR ESTE ALGORITMO DE RESOLUCIÓN**).

$$secuencia[0] = valor\_inicial$$

$$secuencia[n] = \begin{cases} \dfrac{secuencia[n-1]}{2}, si\ secuencia[n-1] es\ par \\ secuencia[n-1]x3+1, \quad si\ secuencia[n-1] es\ impar \end{cases}$$

b) La secuencia terminará cuando alcancemos el valor 1.

c) La secuencia habrá que almacenarla a partir de la posición `secuencia`, y en la variable `secuencia_tamanho` habrá que almacenar el número de términos de la secuencia.

d) En las variables `secuencia_maximo` y `secuencia_valor_medio` deberán almacenarse el valor máximo obtenido en la secuencia y el valor medio de todos los términos de la misma.

e) Además, se desea rellenar una lista de la siguiente manera (`lista`).

   a. Siendo `vT=secuencia_tamanho`, `vIni=valor_inicial`, `vMax=secuencia_maximo` y `vMed=secuencia_valor_medio`
   b. `lista = [ vIni*vT, vMax*vT, vMed*vT, (vIni/vMax)*vT, (vIni/vMed)*vT, (vMax/vIni)*vT, (vMax/vMed)*vT, (vMed/vIni)*vT, vMed/vMax)*vT ]`

f) Para finalizar se calculará el valor medio de los elementos de `lista` y se almacenará en la variable `lista_valor_medio`

g) Como parámetro de entrada se tendrá únicamente `valor_inicial` y podrá modificarse (1-100, ambos incluidos)

h) En las variables `secuencia`, `secuencia_tamanho`, `secuencia_maximo`, `secuencia_valor_medio`, `lista`, `lista_valor_medio` al finalizar la ejecución del programa deberán aparecer los valores pedidos.

```
.data
;; VARIABLES DE ENTRADA Y SALIDA: NO MODIFICAR ORDEN
; VARIABLE DE ENTRADA: (SE PODRA MODIFICAR EL VALOR ENTRE 1 Y 100)
valor_inicial:              .word         97

;; VARIABLES DE SALIDA:
secuencia:                  .space        120*4
secuencia_tamanho:          .word         0
secuencia_maximo:           .word         0
secuencia_valor_medio:      .float        0
lista:                      .space        9*4
lista_valor_medio:          .float        0
;; FIN VARIABLES DE ENTRADA Y SALIDA
```

## 2. Se pide

a) Realizar dos versiones del cálculo pedido:
   a. Una versión no optimizada que realice el cálculo (**empleando bucles**)
   b. Una versión optimizada del cálculo realizado empleando las técnicas habituales de uso de registros adicionales, reordenación de código, desenrollamiento de bucles, etc.

b) Se debe mantener el orden de las variables de entrada y salida en memoria.

c) En ambas versiones el resultado debe ser almacenado en las variables de memoria indicadas en el enunciado y el valor de entrada se puede cambiar.

## 3. Se deberá entregar

a) Los dos ficheros con las dos versiones del programa (normal y optimizada), comentadas.

b) Un documento explicando las mejoras realizadas y comparación de resultados obtenidos.

**Las pruebas a realizar se harán con la siguiente configuración, ver tabla, y para la comparación de ciclos se emplearán como valor inicial 97, 66 y 10, pero deberá funcionar para todas las configuraciones de entrada permitidas:**

| CONFIGURACIÓN | |
|---|---|
| Memory size: | 0x8000 |
| faddEX-Stages: | 1 |
| faddEX-Cycles: | 2 |
| fmulEX-Stages: | 1 |
| fmulEX-Cycles: | 5 |
| fdivEX-Stages: | 1 |
| fdivEX-Cycles: | 19 |
| Forwarding: | enabled |

| ESTADÍSTICAS | |
|---|---|
| Total | |
| Nº de ciclos: | |
| Nº de instrucciones ejecutadas (IDs): | |
| Stalls | |
| RAW stalls: | |
| LD stalls: | |
| Branch/Jump stalls: | |
| Floating point stalls: | |
| WAW stalls: | |
| Structural stalls: | |
| Control stalls: | |
| Trap stalls: | |
| Total | |
| Conditional Branches | |
| Total: | |
| Tomados: | |
| No tomados: | |
| Instrucciones Load/Store | |
| Total: | |
| Loads: | |
| Stores: | |
| Instrucciones de punto flotante | |
| Total: | |
| Sumas: | |
| Multiplicaciones: | |
| Divisiones: | |
| Traps | |
| Traps: | |

## 4. Lugar de entrega

La entrega se realizará en Studium en las fechas indicadas. Se subirá un único archivo (.zip, .rar, etc.) que contenga lo contemplado en el punto 3.

## 5. Evaluación de la práctica

Para aprobar la práctica se deberán entregar **las dos versiones y que el resultado sea correcto, para cualquier valor de entrada válido**.
A partir de ahí, según lo entregado, **se obtendrá mayor o menor calificación** dependiendo del número de ciclos empleados para la ejecución en la versión optimizada (**la que haya conseguido un menor número de ciclos de manera correcta para una o varios valores de entrada válidos**), entre todas las prácticas entregadas, la documentación entregada, etc.
La nota final obtenida por cada persona en las prácticas vendrá corregida por un factor real comprendido entre 0 y 1 según la defensa realizada de las mismas.

# La detección de copia parcial o total de la práctica conllevará la suspensión de las prácticas.

**Cualquier modificación de la práctica se notificará en Studium.**