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) En esta ocasión se ha tomado la siguiente fórmula para el cálculo del número π "*Bailey–Borwein–Plouffe formula (BBP formula)*" de 1995. https://en.wikipedia.org/wiki/Bailey-Borwein-Plouffe_formula

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

b) La precisión obtenida depende del tipo de dato empleado. En esta práctica la importancia está en realizar una serie de cálculos. Se van a emplear word (entero de 32 bits) y float (IEEE 754 de 32 bits).

c) Dado que se pretenden ir obteniendo y almacenando las distintas partes de la fórmula para las distintas iteraciones estas se van a limitar a 5, desde k=0 a k=4.

d) Para cada iteración se guardarán en memoria los siguientes cálculos (no pueden estar precalculados):

a.	16^k	(word: potencias16)
b.	$1/16^k$	(float: calculoA)
c.	$8k$	(word: kpor8)
d.	$8k+1$	(word: kpor8mas1)
e.	$8k+4$	(word: kpor8mas4)
f.	$8k+5$	(word: kpor8mas5)
g.	$8k+6$	(word: kpor8mas6)
h.	$4/(8k+1)$	(float: calculoB)
i.	$2/(8k+4)$	(float: calculoC)
j.	$1/(8k+5)$	(float: calculoD)
k.	$1/(8k+6)$	(float: calculoE)
l.	$4/(8k+1) - 2/(8k+4) - 1/(8k+5) - 1/(8k+6)$	(float: calculoBCDE)
m.	$(1/16^k)(4/(8k+1) - 2/(8k+4) - 1/(8k+5) - 1/(8k+6))$	(float: calculoITE)
n.	El valor acumulado de π	(float: calculoPI)

e) Para esto se tendrá una variable que especificará el número de iteraciones y espacio suficiente para ir guardando los parciales en las distintas iteraciones (de ahí el `.space 5*4` para cada array (5 elementos de 32bits)): (El siguiente código no se puede modificar ni cambiar de orden, salvo el valor de `wIteraciones`)

```
.data
;; INICIO VARIABLES DE ENTRADA Y SALIDA: NO MODIFICAR ORDEN
;; VARIABLE DE ENTRADA:
wIteraciones:               .word       5
;; VARIABLES DE SALIDA:
potencias16:                .space      5*4
kpor8:                      .space      5*4
kpor8mas1:                  .space      5*4
kpor8mas4:                  .space      5*4
kpor8mas5:                  .space      5*4
kpor8mas6:                  .space      5*4
calculoA:                   .space      5*4
calculoB:                   .space      5*4
calculoC:                   .space      5*4
calculoD:                   .space      5*4
calculoE:                   .space      5*4
calculoBCDE:                .space      5*4
calculoITE:                 .space      5*4
calculoPI:                  .space      5*4
;; FIN VARIABLES ENTRADA Y SALIDA
;; A partir de aquí declarar las variables que se estimen necesarias
```

f) En la variable `wIteraciones` se tendrá el número máximo de iteraciones a realizar de 0 a 5 y debe de funcionar para cada uno de esos valores, en el caso de que sea 0 irá directamente al `trap 0`.

g) Adicionalmente en el registro `f31` se tendrá el valor de PI obtenido después de cada iteración.

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 según lo indicado.

3. Se deberá entregar

a) Los dos ficheros fuente, con las dos versiones del programa (normal y optimizada), comentadas.
b) Un documento explicando las mejoras realizadas y los resultados obtenidos.

Las pruebas a realizar se harán con la siguiente configuración, ver tabla (IMPORTANTE: forwarding disabled**)**

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

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.