# Sesión 3: Unrolling y salto

## Técnica de optimización: Loop unrolling

En esta sesión se comparan las prestaciones de diferentes versiones de un código sobre un procesador segmentado teniendo en cuenta las transformaciones que se pueden aplicar a los programas para optimizar su ejecución en un procesador segmentado con salto fijo. Para ello se utilizará el siguiente programa (escrito en C) con **N par**:

```
void Mul(float *a, float *b, int N){
       for ( i = 0; i<N; i++)
           b[i] = a[i] * b[i];
}
```

1. Repaso. En Moodle encontrarás el código traducido a instrucciones MIPS (S3-unroll.s). Configura el simulador para utilizar salto fijo no efectivo con las latencias por defecto. Reorganiza el código para mejorar las prestaciones de la ejecución con planificación estática ¿cuál es la aceleración?

2. Una técnica para obtener un mayor rendimiento en los bucles que accede a vectores es el *desenrollado de bucles (Loop Unrolling)*; como el nombre indica, se realizan múltiples copias del cuerpo del bucle y se planifican juntas instrucciones que pertenecen a iteraciones diferentes. Aplica dicha técnica al código anterior y compara el rendimiento entre el código original y el código desenrollado planificado. Una versión en C de desenrollado de bucles es la siguiente:

```
assert(N%2 == 0);
void Mul(float *a, float *b, int N){
       for ( i = 0; i<N; i+=2){
            b[i] = a[i] * b[i];
            b[i+1] = a[i+1] * b[i+1];
       }
}
```

# Técnicas de procesamiento de salto

El objetivo de esta parte es comparar las dos técnicas de procesamiento de salto que se estudian en teoría: salto fijo no efectivo y salto retardado. También se pretende comprobar en qué consiste la técnica de salto retardado y cómo se puede mejorar el código con técnicas de optimización.

Cargar y ejecutar en la configuración segmentada el siguiente código:

```
 .text
.globl main

main:
          la $t0, array
          la $t4, count
          lw $t1, 0($t4)
          addi $t2, $0, 0
          addi $t0, $t0, 20

Loop:
          lw $t3, 0($t0)
          add $t2, $t3, $t2
          addi $t1, $t1, -1
          addi $t0, $t0, -4
          bne $t1, $0, Loop
          sw $t2, 0($t4)

fin:
         addi $v0, $0, 10
         syscall

.data
array: .word 3, 4, 9, 8, 5, 1, 3
count: .word 6
```

Analiza qué está haciendo el código y contesta a la batería de preguntas del cuestionario habilitado en la página de la asignatura en el Moodle de la UDC (`https://udconline.udc.gal`).

Cargar y ejecutar en la configuración segmentada el siguiente código:

```
.data
  N: .word 10
  v1: .float 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  v2: .float 1, 2, 3, 1, 2, 3, 1, 2, 3, 1
  r: .float 0

.text
.globl main

main:
  la $t0, N
  lw $t0, 0($t0)
  la $t1, v1
  la $t2, v2
  la $t3, r
  lwc1 $f12, 0($t3)      # f12 = 0.0

Loop:
  lwc1 $f2, 0($t1)       # f2 = v1[i]
  add.s $f12, $f12, $f2  # f12 += f2
  lwc1 $f2, 0($t2)       # f2 = v2[i]
  mul.s $f4, $f2, $f2    # f4 = f2 * f2
  swc1 $f4, 0($t1)       # v1[i] = f4
  addi $t1, $t1, 4
  addi $t2, $t2, 4
  addi $t0, $t0, -1
  bne $t0, $0, Loop
  nop

end:
  swc1 $f12, 0($t3)
  addi $v0, $0, 2
  syscall

  addi $v0, $0, 10
  syscall
```

Analiza qué está haciendo el código y contesta a la batería de preguntas del cuestionario habilitado en la página de la asignatura en el Moodle de la UDC (`https://udconline.udc.gal`).