<u>Notas</u>

LUI: (Load Upper Immediate) ORI: (OR Immediate)

LWC1: Load Word Coprocessor 1 (floating point)

El color que muestra el simulador depende del tipo de instrucción.

Paso 3)

Riesgos

```
cvt.d.s $f0, $f0 -> 1 ciclos de parada (ciclo 15) cvt.d.s $f2, $f2 -> 1 ciclos de parada (ciclo 18)
```

BUCLE

```
add.d $f8, $f8, $f6 -> 6 ciclos de parada (ciclo 25) div.d $f8, $f8, $f2 -> 3 ciclos de parada (ciclo 32) swc1 $f8, 0($t3) -> 9 ciclos de parada (ciclo 36)
```

bne \$t1, \$t6, loop -> 1 ciclos de parada del salto **BUCLE**

Inst. Primera Iteración (desde la etiqueta de loop:)

```
Instrucciones = 13
Total ciclos de parada = 18 datos + 1 control
Ciclos del bucle = 13 (1 instrucción por ciclo) + 19 (ciclos de parada) + 4 (rellenar el pipeline) = 36
```

Paso 4)

Inicialización (fuera del bucle) + Iteraciones (CUIDADO PREDICTOR) + finalización

```
Instrucciones totales = 15 + 13*100 + 2 (contando syscall) = 1317
Total ciclos de parada = 2 + [(99*19) + 18] + 0 = 1901
Ciclos totales: Instrucciones totales + Ciclos de parada totales + 4 (4 es porque la primera instrucción tarda 4 ciclos extra más el suyo propio en salir del pipeline. \rightarrow 1317 + 1901 + 4 = 3222
```

Nota: El simulador no cuenta el syscall como instrucción, con lo que aparecen 1316 en las estadísticas. Sin embargo, sí que lo cuenta para el total de ciclos.

CPI = ciclos / instrucciones \rightarrow 3222/1316 = 2.448 CPI

Paso 5)

Código optimizado

```
# Datos
.data
a: .float 10
b: .float 5
#vector x
   .double 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
x1: .double 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
x2: .double 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
x3: .double 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
# IEEE 754 64 bits, 1 = 0x3FF00000 00000000
#vector y
     у:
     v1:
y2:
     y3:
     y4:
# IEEE 754 64 bits, 10 = 0 \times 40240000 00000000
#vector z
z:
     .space 800
# Código: z[i]=(a*x[i]+y[i])/b
.text
.globl main
main:
      la $t1, x
                       #carga la dir de x[0] en $t1
      la $t2, y
                       #carga la dir de y[0] en $t2
      la $t3. z
                       #carga la dir de z[0] en $t3
                       #carga la dir de "a" en $t4
#carga la dir de "b" en $t5
      la $t4, a
      la $t5, b
      addi $t6,$t1,800
                       #preparamos el contador para detectar el final
                       #carga "a" en f0
      lwc1 $f0, 0($t4)
                       #carga "b" en f2
      lwc1 $f2, 0($t5)
                       #convertimos "a" en DP y lo guardamos en $f0:$f1
      cvt.d.s $f0, $f0
      cvt.d.s $f2, $f2
                       #convertimos "b" en DP y lo quardamos en $f2:$f3
loop:
      lwc1 $f4,0($t1)
                       #f4=x[i] (parte alta)
      lwc1 $f5,4($t1)
                       #f5=x[i] (parte baja)
      mul.d $f8,$f0,$f4
                       #f8=a*x[i]
      lwc1 $f6,0($t2)
                       #f6=y[i] (parte baja)
                       #f7=x[i] (parte baja)
      lwc1 $f7,4($t2)
      addi $t1,$t1,8
                       #actualizamos todos los índices
      addi $t2,$t2,8
      add.d $f8,$f8,$f6
                       #f8=a*x[i]+y[i]
      div.d $f8,$f8,$f2
                       #f8=(a*x[i]+y[i])/b
      swc1 $f8, 0($t3)
                       \#z[i]=(a*x[i]+y[i])/b (Parte alta)
      swc1 $f9, 4($t3)
                       \#z[i]=(a*x[i]+y[i])/b (Parte baja)
      addi $t3,$t3,8
      bne $t1, $t6, loop
                       #comprobamos si hemos terminado el bucle
      addi $v0, $0, 10
                       # llamada para salir del programa
      syscall.
```

Ciclos: 2820 CPI: 2.14285

Speedup = ((2.5243 / 2.14285) - 1)*100 = 17,8159 %

Paso 6)

Ciclos: 547 | Instrucciones: 259

CPI: 547/259 = 2,111969112

| - Estructurales: 210

Ciclos de parada | - Datos: 64

- Control: 9

Paso 7)

Ciclos: 387 | Instrucciones: 259

CPI: 387/259 = 1,494208494

| - Estructurales: 50

Ciclos de parada | - Datos: 64

| - Control: 9

Mejoran los riesgos estructurales, que son los únicos que deben verse afectados por aumentar las unidades funcionales.

Paso 8)

Unidades mínimas: 4 multiplicadores y 2 sumadores.

Pero, ¿por qué no 3 sumadores? → Existe un riesgo de datos que hace que el riesgo estructural del tercer add (add.d \$f20,\$f16,\$f18) nunca llegue a ocurrir.

CPI: 337/259 = 1,301158301

| - Estructurales: 0

Ciclos de parada | - Datos: 64

| - Control: 9

Estructurales : $0 \rightarrow \text{Las}$ unidades están segmentadas y disponibles en el ciclo siguiente. Como sólo se lanza una instrucción no pueden tener riesgos estructurales.

Datos: 3 de la suma (add.d \$f20,\$f16,\$f18) y 3 del store (swc1 \$f20, -8(\$t5)).

Control: 1 ciclo.

Ciclos totales: Instrucciones totales + Ciclos de parada totales + 4 (4 es porque la primera instrucción tarda 4 ciclos extra más el suyo propio en salir del pipeline.

Ciclos totales: 259 + [4 + (9*7) + 6] + 4 = 336

Nota: 4 ciclos de parada fuera del bucle + 9 iteraciones con 6 ciclos de parada por datos y 1 de control + 1 iteración sólo con ciclos de parada por datos + 4 ciclos de rellenar el pipeline.

CPI: 336/259 = 1.306