Desenrollado de bucles

Salvador Carrillo Fuentes

Noviembre 2016

1 Introducción

El desenrollado de bucles es una técnica que permite obtener un mayor rendimiento en el tiempo de ejecución de los programas. El mayor rendimiento se obtiene mediante el aprovechamiento del paralelismo que se puede obtener en los bucles de acceso a los elementos de un array.

Esta técnica, generalmente, la aplica el compilador realizando varias copias del código del bucle, además de utilizar algún registro adicional del procesador (renombrado de registros) para eliminar ciertos riesgos que puedan aparecer en el nuevo código.

No siempre es fácil aplicar el desenrollado, ya que en ciertas situaciones el número de veces que se ejecuta un bucle no es un valor estático, sino que depende de una variable cuyo contenido no es conocido en tiempo de compilación. Otra dificultad añadida puede ser la no disponibilidad de registros para evitar ciertas dependencias de datos (`RAW`).

2 Resultados del fichero *P3.s*

Ciclos	3854
Instrucciones	3369
`CPI`	1.144
Riesgos `RAW`	481
Tamaño del código	76 *bytes*

3 Resultados del fichero *P3-2.s*

Desenrollado del bucle en dos copias. El bucle se ejecutará la mitad de veces que el original.

Ciclos	3494
Instrucciones	3009
`CPI`	1.161
Riesgos `RAW`	481
Tamaño del código	88 *bytes*

4 Resultados *P3-2-opt.s*

Reorganización del código para evitar dependencias de datos `RAW`.

Ciclos	3254
Instrucciones	3009
`CPI`	1.081
Riesgos `RAW`	241
Tamaño del código	88 *bytes*

5 Resultados *p3-4.s*

Desenrollado del bucle `LOOP` en cuatro copias usando los registros `r12` y `r13` para las copias. El bucle queda de la siguiente manera:

```
LOOP:
    daddi r1,r1,-16  ; Actualizar la variable índice

    lw r10,12(r1)    ; Leer un elemento de un vector
    daddi r10,r10,4  ; Sumar 4 al elemento
    sw r10,12(r1)    ; Escribir el nuevo valor

    lw r11,8(r1)     ; 2ª copia. Leer un elemento de un vector
    daddi r11,r11,4  ; Sumar 4 al elemento
    sw r11,8(r1)     ; Escribir el nuevo valor

    lw r12,4(r1)     ; 3ª copia. Leer un elemento de un vector
    daddi r12,r12,4  ; Sumar 4 al elemento
    sw r12,4(r1)     ; Escribir el nuevo valor

    lw r13,0(r1)     ; 4ª copia. Leer un elemento de un vector
    daddi r13,r13,4  ; Sumar 4 al elemento
    sw r13,0(r1)     ; Escribir el nuevo valor

    bne r1,r0,LOOP   ; Fin de vector?
    nop
    halt
```

Factor 4 sin optimizar:

Ciclos	3314
Instrucciones	2829
`CPI`	1.171
Riesgos `RAW`	481
Tamaño del código	112 *bytes*

6 Resultados *p3-4opt.s*

Se reorganiza el código del aparatdo anterior eliminando el mayor número de dependencias `RAW`. El bucle queda de la siguiente manera:

```
LOOP:
    daddi r1,r1,-16  ; Actualizar la variable índice

    lw r10,12(r1)    ; Leer un elemento de un vector
    lw r11,8(r1)     ; 2ª copia. Leer un elemento de un vector
    lw r12,4(r1)     ; 3ª copia. Leer un elemento de un vector
    lw r13,0(r1)     ; 4ª copia. Leer un elemento de un vector

    daddi r10,r10,4  ; Sumar 4 al elemento
    daddi r11,r11,4  ; Sumar 4 al elemento
    daddi r12,r12,4  ; Sumar 4 al elemento
    daddi r13,r13,4  ; Sumar 4 al elemento

    sw r10,12(r1)    ; Escribir el nuevo valor
    sw r11,8(r1)     ; Escribir el nuevo valor
    sw r12,4(r1)     ; Escribir el nuevo valor
    sw r13,0(r1)     ; Escribir el nuevo valor

    bne r1,r0,LOOP   ; Fin de vector?
    nop
    halt
```

Factor 4 (optimizado con reorganización de código):

Ciclos	3074
Instrucciones	2829
`CPI`	1.087
Riesgos `RAW`	241
Tamaño del código	112 *bytes*

7 Resultados fichero *p3-6.s*

Desenrollado del bucle `LOOP` en seis copias, utilizando los registros `r14` y `r15` para las nuevas copias.

Así queda el bucle:

```
LOOP:
    daddi r1,r1,-24  ; Actualizar la variable índice

    lw r10,20(r1)    ; Leer un elemento de un vector
    daddi r10,r10,4  ; Sumar 4 al elemento
    sw r10,20(r1)    ; Escribir el nuevo valor

    lw r11,16(r1)    ; 2ª copia. Leer un elemento de un vector
    daddi r11,r11,4  ; Sumar 4 al elemento
    sw r11,16(r1)    ; Escribir el nuevo valor

    lw r12,12(r1)    ; 3ª copia. Leer un elemento de un vector
    daddi r12,r12,4  ; Sumar 4 al elemento
    sw r12,12(r1)    ; Escribir el nuevo valor

    lw r13,8(r1)     ; 4ª copia. Leer un elemento de un vector
    daddi r13,r13,4  ; Sumar 4 al elemento
    sw r13,8(r1)     ; Escribir el nuevo valor

    lw r14,4(r1)     ; 5ª copia. Leer un elemento de un vector
    daddi r14,r14,4  ; Sumar 4 al elemento
    sw r14,4(r1)     ; Escribir el nuevo valor

    lw r15,0(r1)     ; 6ª copia. Leer un elemento de un vector
    daddi r15,r15,4  ; Sumar 4 al elemento
    sw r15,0(r1)     ; Escribir el nuevo valor

    bne r1,r0,LOOP   ; Fin de vector?
    nop
    halt
```

Factor seis sin optimizar:

Ciclos	3254
Instrucciones	2769
`CPI`	1.175
Riesgos `RAW`	481
Tamaño del código	136 *bytes*

8 Resultados *p3-6opt.s*

Reorganizamos el código del apartado anterior para eliminar las dependencias RAW.

Así queda el bucle:

```
LOOP:
    daddi r1,r1,-24  ; Actualizar la variable índice

    lw r10,20(r1)    ; Leer un elemento de un vector
    lw r11,16(r1)    ; 2ª copia. Leer un elemento de un vector
    lw r12,12(r1)    ; 3ª copia. Leer un elemento de un vector
    lw r13,8(r1)     ; 4ª copia. Leer un elemento de un vector
    lw r14,4(r1)     ; 5ª copia. Leer un elemento de un vector
    lw r15,0(r1)     ; 6ª copia. Leer un elemento de un vector

    daddi r10,r10,4  ; Sumar 4 al elemento
    daddi r11,r11,4  ; Sumar 4 al elemento
    daddi r12,r12,4  ; Sumar 4 al elemento
    daddi r13,r13,4  ; Sumar 4 al elemento
    daddi r14,r14,4  ; Sumar 4 al elemento
    daddi r15,r15,4  ; Sumar 4 al elemento

    sw r10,20(r1)    ; Escribir el nuevo valor
    sw r11,16(r1)    ; Escribir el nuevo valor
    sw r12,12(r1)    ; Escribir el nuevo valor
    sw r13,8(r1)     ; Escribir el nuevo valor
    sw r14,4(r1)     ; Escribir el nuevo valor
    sw r15,0(r1)     ; Escribir el nuevo valor

    bne r1,r0,LOOP   ; Fin de vector?
    nop
    halt
```

Factor seis, optimizado con reorganización de código:

Ciclos	3014
Instrucciones	2769
CPI	1.088
Riesgos RAW	241
Tamaño del código	136 *bytes*

9 Aceleración de cada versión del programa con respecto al original

`P3.s` = nº ciclos · `tck` = 3854 · `tck`

Versión del programa	Aceleración
P3-2.s	1.1030
P3-2opt.s	1.1843
P3-4.s	1.1629
P3-4opt.s	1.2537
P3-6.s	1.1843
P3-6opt.s	1.2786