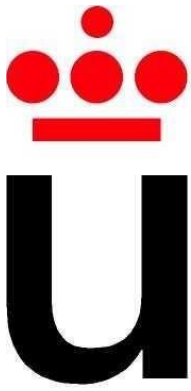
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA



Universidad
Rey Juan Carlos

PRÁCTICA 2
ARQUITECTURA DE COMPUTADORES

Integrantes del grupo: Alberto Martín Amengual, Álvaro Martínez Quiroga.

SECCIÓN 1:

Apartado 1: código para calcular el área de un círculo.

DATA

```
.data

resultado:                                  .double 0
titulo:                                     .asciiz "Calculo circulo.\n"
pregunta:                                   .asciiz "Radio= "
str:                                        .asciiz "Resultado= "
PI:                                         .double 3.14159265358979
CONTROL:                                    .word32 0x10000
DATA:                                       .word32 0x10008
```

TEXT

```
.text

lwu r8,DATA(r0)                             ;r8 para CONTROL
lwu r9,CONTROL(r0)                          ;r9 para DATA

daddi r11,r0,4                              ;Activado para imprimir
string

daddi r1,r0,titulo                          ;Apuntamos a titulo
sd r1,(r8)                          ;Preparamos los datos apuntados a imprimir
sd r11,(r9)                                 ;Imprimimos titulo

daddi r1,r0,pregunta                        ;Apuntamos a pregunta
sd r1,0(r8)                         ;Preparamos los datos apuntados a imprimir
sd r11,0(r9)                                ;Imprimimos pregunta

l.d f5, PI(r0)                              ;Carga de PI
daddi r3,r0,8                       ;Preparamos para CONTROL
sd r3,0(r9)                 ;CONTROL = 8 (input de tipo int o float por teclado)
l.d f1,0(r8)                                ;f1 es el numero introducido
cvt.d.l f1, f1      ;Controlamos el numero entero introducido, lo convertimos a tipo float
mul.d f2, f1, f1                            ;Calculamos
mul.d f3, f2, f5
daddi r1, r0, str                           ;Carga para str
daddi r2, $zero, 3                          ;Para CONTROL = 3
sd r1, (r8)
sd r11, (r9)                                ;Control imprime str
s.d f3, (r8)
sd r2, (r9)                                 ;Control imprime el resultado

s.d f3, resultado(r0)                       ;Guardamos el resultado

halt
```

Ejecutando el código anterior, sin mejoras, obtenemos los siguientes resultados:

- Ciclos: 52.
- CPI: 2.130 ciclos por instrucción.
- 24 paradas por RAWs.

Apartado 2: ejecución del código anterior con mejoras en el hardware del procesador y técnicas de adelantamiento.

- ✓ Una opción muy obvia de evitar paradas seria omitir los mensajes de "titulo", "pregunta" y "str" y únicamente esperar a que el usuario introduzca el número del radio del circulo y después de haber hecho los cálculos imprimirlo por pantalla y guardarlo en memoria o directamente que este sea guardado en memoria sin avisar.
- ✓ Activando "Delay Slot" y habilitando el adelantamiento obtenemos los siguientes resultados:
  - Ciclos: 39.
  - CPI: 1.625 ciclos por instrucción.
  - 18 paradas por RAWs.

SECCIÓN 2:

Apartado 3: estudio del código “SAXPY.s”, sin opciones de adelantamiento habilitadas ni condiciones de salto.

- Ciclos: 307.
- CPI: 2.952 ciclos por instrucción.
- 190 pardas por RAWs.

CICLO	INSTRUCCIÓN 1	INSTRUCCIÓN 2	REGISTRO CAUSA	RESUELTO CICLO	TIPO
7	cvt.d.l f2,f2	mtc1 r2, f2	f2	9	RAW
10	mul.d f3,f1,f2	cvt.d.l f2,f2	f2	12	RAW
14	add.d f5,f3,f4	l.d f4, y(r1)	f4	22	RAW
23	cvt.l.d f6,f5	add.d f5,f3,f4	f5	28	RAW
31	bne r1, r3, loop	daddi r1, r1, 8	r1	33	RAW

A partir de este momento la ejecución del programa vuelve a entrar en la misma secuencia de instrucciones por la ejecución de la instrucción **bne r1, r3, loop** dando las siguientes instrucciones:

- Ciclo 37: se repite RAW ciclo 7.
- Ciclo 40: se repite RAW ciclo 10.
- Ciclo 44: se repite RAW ciclo 14.
- Ciclo 53: se repite RAW ciclo 23.
- Ciclo 61: se repite RAW ciclo 31.

Podemos suponer que estos riesgos seguirán ocurriendo hasta que la condición de salida obtenga el resultado **r1 == r3** y podremos pasar a la ejecución de la instrucción **halt** o de fin de programa.

- Ciclo 303: la comprobación resulta en verdadera (r1==r3) y se comienza a ejecutar la instrucción de fin de programa.

Apartado 4: ejecución del código con “Forwarding” o adelantamiento habilitado.

- Ciclos: 217.
- CPI: 2.087 ciclos por instrucción.
- 150 paradas por RAWs.
- Speedup: CPI NUEVO / CPI ANTIGUO = 2.087 / 2.952 = 0.70.

Apartado 5: reordenación de Código: (más adelantamientos). Nota que se ha eliminado la instrucción **cvt.l.d f6,f5** ya que el registro **f6** no se utiliza y consume ciclos cada una de las ejecuciones del bucle. Hemos marcado en naranja las instrucciones que cambian de lugar.

CODIGO
.text
daddi r2, r0, 2 ;alpha
daddi r1, r0,0
;aquí estaba daddi r2, r0, 2
daddi r3, r0, 80
loop:
mtc1 r2, f2
l.d f1, x(r1)
;aquí estaba mtc1 r2, f2
cvt.d.l f2, f2
mul.d f3, f1, f2
l.d f4, y(r1)
add.d f5,f3,f4
;cvt.l.d f6, f5 ;f6 no se usa? ¿Se puede obviar esta instrucción?
s.d f5, s(r1)
daddi r1, r1, 8
bne r1, r3, loop
halt

- Ciclos: 207.
- CPI: 2.202 ciclos por instrucción.
- 150 paradas por RAWs.

Apartado 6: activamos la opción de salto más adecuada para la ejecución y lo corremos con esta mejora más las anteriores.

! Hemos activado la opción de “Branch target buffer” ya que si activamos “Delay slot” el procesador pasara a ejecución después de la primera iteración del bucle la instrucción de fin de programa “halt” y el programa acabara.

Obtenemos los siguientes resultados:

- Ciclos: 202.
- CPI: 2.149 ciclos por instrucción.
- 150 paradas por RAWs.
- Speedup: CPI NUEVO / CPI ANTIGUO = 2.149 / 2.202 = 0.97 (prácticamente igual el rendimiento al ejecutar).

SECCIÓN 3:

Apartado 7: estudio del código "Fahrenheit2celsius.s", sin opciones de adelantamiento habilitadas ni condiciones de salto.

- Ciclos: 478.
- CPI: 4.641 ciclos por instrucción.
- 362 pardas por RAWs.

CICLO	INSTRUCCIÓN 1	INSTRUCCIÓN 2	REGISTRO CAUSA	RESUELTO CICLO	TIPO
5	l.d f3, f(r1)	daddi r1, r0, 0	r1	6	RAW
8	sub.d f7, f3, f4	l.d f4, s(r0)	f4	10	RAW
13	div.d f8, f5, f6	l.d f6, d(r0)	f6	15	RAW
16	mul.d f9, f7, f8	div.d f8, f5, f6	f8	37	RAW
38	s.d f9, c(r1)	mul.d f9, f7, f8	f9	47	RAW
49	bne r1, r10, loop	daddi r1, r1, 8	r1	51	RAW

Al igual que en el ejemplo anterior el bucle se repite un número de veces, en este caso el bucle se repetirá hasta 10 veces y después de ellos para ejecutar la instrucción "halt".

! En el ciclo 474 la instrucción de condición de salto Dara verdadera (r1 == r10) y comenzara la ejecución de la instrucción halt.

Apartado 8: ejecución del código con "Forwarding" o adelantamiento habilitado.

- Ciclos: 396.
- CPI: 3.845 ciclos por instrucción.
- 480 pardas por RAWs (podemos observar una mejora del número de ciclos, pero empeora el número de paradas por riesgos de tipo RAW).
- Speedup: CPI NUEVO / CPI ANTIGUO = 3.845 / 4.641 = 0.82

Apartado 9: reordenación de Código: (más adelantamientos). Hemos marcado en naranja las instrucciones que cambian de lugar.

CODIGO
.text daddi r1,r0,0 ;instrucción que sube daddi r10,r0,80 ;aquí estaba: daddi r1,r0,0 loop: l.d f3,f(r1) l.d f4,s(r0) ;aquí estaba: sub.d f7,f3,f4 l.d f5,n(r0) l.d f6,d(r0) sub.d f7,f3,f4 ;instrucción que baja div.d f8,f5,f6 mul.d f9,f7,f8 s.d f9,c(r1) daddi r1,r1,8 bne r1,r10,loop halt

- Ciclos: 386.
- CPI: 3.748 ciclos por instrucción.
- 450 pardas por RAWs.
- Speedup: CPI NUEVO / CPI ANTIGUO = 3.748 / 3.845 = 0.97

Apartado 10: activamos la opción de salto más adecuada para la ejecución y lo corremos con esta mejora más las anteriores.

! Hemos activado la opción de “Branch target buffer” ya que si activamos “Delay slot” el procesador pasara a ejecución después de la primera iteración del bucle la instrucción de fin de programa “halt” y el programa acabara.

Obtenemos los siguientes resultados:

- Ciclos: 381.
- CPI: 3.699 ciclos por instrucción.
- 450 paradas por RAWs.
- Speedup: CPI NUEVO / CPI ANTIGUO = 3.699 / 3.748 = 0.98 (prácticamente igual el rendimiento al ejecutar).