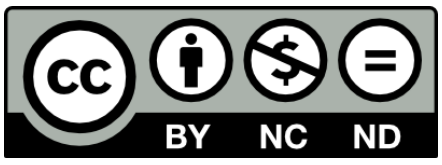
Arquitectura de Computadores

Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Arquitectura de Computadores

Los Del DGIIM, `losdeldgiim.github.io`

José Juan Urrutia Milán

Granada, 2023-2024

Índice general

0.1. Arquitecturas ILP 4
0.1.1. Cuestiones 10

0.1. Arquitecturas ILP

Ejercicio 0.1.1. Considere el fragmento de Código 1.

```
1  lw  r1,0x1ac    ; r1 <-- M(0x1ac)
2  lw  r2,0xc1f    ; r2 <-- M(0xc1f)
3  add r3,r0,r0    ; r3 <-- r0+r0
4  mul r4,r2,r1    ; r4 <-- r2*r1
5  add r3,r3,r4    ; r3 <-- r3+r4
6  add r5,r0,0x1ac ; r5 <-- r0+0x1ac
7  add r6,r0,0xc1f ; r6 <-- r0+0xc1f
8  sub r5,r5,#4    ; r5 <-- r5 - 4
9  sub r6,r6,#4    ; r6 <-- r6 - 4
10 sw  (r5),r3     ; M(r5) <-- r3
11 sw  (r6),r4     ; M(r6) <-- r4
```

Código fuente 1: Código para trabajar.

Suponiendo que se pueden captar, decodificar, y emitir cuatro instrucciones por ciclo, indique el orden en que se emitirán las instrucciones para cada uno de los siguientes casos:

1. Una ventana de instrucciones centralizada con emisión ordenada.

   Se encuentra resuelto en la Tabla 1.

2. Una ventana de instrucciones centralizada con emisión desordenada.

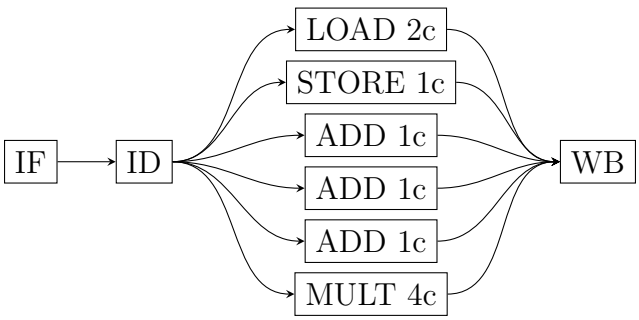
   Se encuentra resuelto en la Tabla 2.

3. Una estación de reserva de tres líneas para cada unidad funcional, con envío ordenado.

   Se encuentra resuelto en la Tabla 3.

Nota: considere que hay una unidad funcional para la carga (2 ciclos), otra para el almacenamiento (1 ciclo), tres para la suma/resta (1 ciclo), y una para la multiplicación (4 ciclos). También puede considerar que, en la práctica, no hay límite para el número de instrucciones que pueden almacenarse en la ventana de instrucciones o en el buffer de instrucciones.

Según el enunciado del ejercicio, el cauce de instrucciones tiene la siguiente forma:



Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`lw r1, 0x1ac`	IF	ID	EX	EX										
`lw r2, 0xc1f`	IF	ID			EX	EX								
`add r3, r0, r0`	IF	ID			EX									
`mul r4, r2, r1`	IF	ID					EX	EX	EX	EX				
`add r3, r3, r4`		IF	ID								EX			
`add r5, r0, 0x1ac`		IF	ID								EX			
`add r6, r0, 0xc1f`		IF	ID								EX			
`sub r5, r5, #4`		IF	ID									EX		
`sub r6, r6, #4`			IF	ID								EX		
`sw (r5), r3`			IF	ID									EX	
`sw (r6), r4`			IF	ID										EX

Tabla 1: Ejecución con ventana centralizada y emisión ordenada del Ejercicio 0.1.1.

Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12
`lw r1, 0x1ac`	IF	ID	EX	EX								
`lw r2, 0xc1f`	IF	ID			EX	EX						
`add r3, r0, r0`	IF	ID	EX									
`mul r4, r2, r1`	IF	ID					EX	EX	EX	EX		
`add r3, r3, r4`		IF	ID								EX	
`add r5, r0, 0x1ac`		IF	ID	EX								
`add r6, r0, 0xc1f`		IF	ID	EX								
`sub r5, r5, #4`		IF	ID		EX							
`sub r6, r6, #4`			IF	ID	EX							
`sw (r5), r3`			IF	ID								EX
`sw (r6), r4`			IF	ID							EX	

Tabla 2: Ventana centralizada y emisión desordenada del Ejercicio 0.1.1.

Además, para cada caso, se ha desarrollado una tabla que muestra la evolución de la ejecución del código según el número de ciclos.

Ejercicio 0.1.2. Cosidere el fragmento de Código 2.

Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12	13
`lw r1, 0x1ac`	IF	ID	EX	EX									
`lw r2, 0xc1f`	IF	ID			EX	EX							
`add r3, r0, r0` (1)	IF	ID	EX										
`mul r4, r2, r1`	IF	ID					EX	EX	EX	EX			
`add r3, r3, r4` (2)		IF	ID								EX		
`add r5, r0, 0x1ac` (3)		IF	ID	EX									
`add r6, r0, 0xc1f` (1)		IF	ID	EX									
`sub r5, r5, #4` (3)		IF	ID		EX								
`sub r6, r6, #4` (1)			IF	ID	EX								
`sw (r5), r3`			IF	ID								EX	
`sw (r6), r4`			IF	ID									EX

Tabla 3: Ejecución con ventanas repartidas y emisión ordenada del Ejercicio 0.1.1.

```
1  lw   r3,0x10a   ; r3 <-- M(0x10a)
2  addi r2,r0,#128 ; r2 <-- r0+128
3  add  r1,r0,0x0a ; r1 <-- r0+0x0a
4  lw   r4,0(r1)   ; r4 <-- M(r1)
5  lw   r5,-8(r1)  ; r5 <-- M(r1-8)
6  mult r6,r5,r3   ; r6 <-- r5*r3
7  add  r5,r6,r3   ; r5 <-- r6+r3
8  add  r6,r4,r3   ; r6 <-- r4+r3
9  sw   0(r1),r6   ; M(r1) <-- r5
10 sw   -8(r1),r5  ; M(r1-8) <-- r5
11 sub  r2,r2,#16  ; r2 <-- r2-16
```

Código fuente 2: Código para trabajar.

Suponga que se ejecuta en un procesador superescalar que es capaz de captar 4 instrucciones/ciclo, de decodificar 2 instrucciones/ciclo; de emitir utilizando una ventana de instrucciones centralizada 2 instrucciones/ciclo; de escribir hasta 2 resultados/ciclo en los registros correspondientes (registros de reorden, o registros de la arquitectura según el caso), y completar (o retirar) hasta 3 instrucciones/ciclo.

Indique el número de ciclos que tardaría en ejecutarse el programa suponiendo finalización ordenada y:

1. Emisión ordenada.

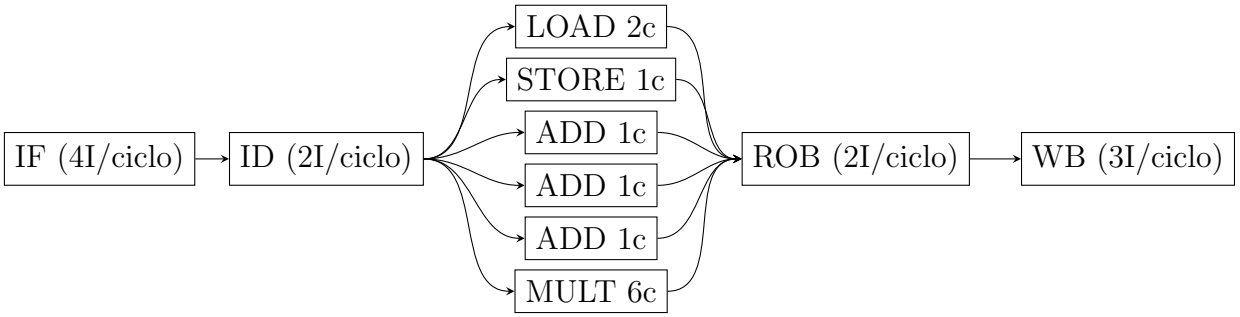
   Se encuentra resuelto en la Tabla 4.

2. Emisión desordenada.

   Se encuentra resuelto en la Tabla 5. Notemos que, como no pueden escribir en el ROB más de una instrucción por ciclo, la instrucción `sub r2,r2,#16` ha de esperar un ciclo más.

Nota: Considere que tiene una unidad funcional de carga (2 ciclos), una de almacenamiento (1 ciclo), tres unidades de suma/resta (1 ciclo), y una de multiplicación (6 ciclos), y que no hay limitaciones para el número de líneas de la cola de instrucciones, ventana de instrucciones, buffer de reorden, puertos de lectura/escritura etc.

Según el enunciado del ejercicio, el cauce de instrucciones tiene la siguiente forma:



Ejercicio 0.1.3. En el problema anterior:

Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12
`lw r3,0x10a`	IF	ID	EX	EX	ROB	WB						
`addi r2,r0,#128`	IF	ID	EX	ROB		WB						
`add r1,r0,0x0a`	IF		ID	EX	ROB	WB						
`lw r4,0(r1)`	IF		ID		EX	EX	ROB	WB				
`lw r5,-8(r1)`		IF		ID			EX	EX	ROB	WB		
`mult r6,r5,r3`		IF		ID					EX	EX	EX	EX
`add r5,r6,r3`		IF			ID							
`add r6,r4,r3`		IF			ID							
`sw 0(r1),r6`			IF			ID						
`sw -8(r1),r5`			IF			ID						
`sub r2,r2,#16`			IF				ID					

Instrucción / Ciclos	13	14	15	16	17	18	19
`lw r3,0x10a`							
`addi r2,r0,#128`							
`add r1,r0,0x0a`							
`lw r4,0(r1)`							
`lw r5,-8(r1)`							
`mult r6,r5,r3`	EX	EX	ROB	WB			
`add r5,r6,r3`			EX	ROB	WB		
`add r6,r4,r3`			EX	ROB	WB		
`sw 0(r1),r6`				EX	ROB	WB	
`sw -8(r1),r5`					EX	ROB	WB
`sub r2,r2,#16`					EX	ROB	WB

Tabla 4: Emisión ordenada del código del Ejercicio 0.1.2.

Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12
`lw r3,0x10a`	IF	ID	EX	EX	ROB	WB						
`addi r2,r0,#128`	IF	ID	EX	ROB		WB						
`add r1,r0,0x0a`	IF		ID	EX	ROB	WB						
`lw r4,0(r1)`	IF		ID		EX	EX	ROB	WB				
`lw r5,-8(r1)`		IF		ID			EX	EX	ROB	WB		
`mult r6,r5,r3`		IF		ID					EX	EX	EX	EX
`add r5,r6,r3`		IF			ID							
`add r6,r4,r3`		IF			ID		EX	ROB				
`sw 0(r1),r6`			IF			ID		EX	ROB			
`sw -8(r1),r5`			IF			ID						
`sub r2,r2,#16`			IF				ID	EX		ROB		

Instrucción / Ciclos	13	14	15	16	17	18	19
`lw r3,0x10a`							
`addi r2,r0,#128`							
`add r1,r0,0x0a`							
`lw r4,0(r1)`							
`lw r5,-8(r1)`							
`mult r6,r5,r3`	EX	EX	ROB	WB			
`add r5,r6,r3`			EX	ROB	WB		
`add r6,r4,r3`					WB		
`sw 0(r1),r6`					WB		
`sw -8(r1),r5`				EX	ROB	WB	
`sub r2,r2,#16`						WB	

Tabla 5: Emisión desordenada del código del Ejercicio 0.1.2.

Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12
`lw r3,0x10a`	IF	ID	EX	EX	ROB	WB						
`addi r2,r0,#128`	IF	ID	EX	ROB		WB						
`add r1,r0,0x0a`	IF	ID	EX	ROB		WB						
`lw r4,0(r1)`	IF	ID		EX	EX	ROB	WB					
`lw r5,-8(r1)`		IF	ID	EX	EX	ROB	WB					
`mult r6,r5,r3`		IF	ID			EX	EX	EX	EX	EX	EX	ROB
`add r5,r6,r3`		IF	ID									EX
`add r6,r4,r3`		IF	ID			EX	ROB					
`sw 0(r1),r6`			IF	ID			EX	ROB				
`sw -8(r1),r5`			IF	ID								
`sub r2,r2,#16`			IF	ID	EX							

Instrucción / Ciclos	13	14	15	16	17	18	19
`lw r3,0x10a`							
`addi r2,r0,#128`							
`add r1,r0,0x0a`							
`lw r4,0(r1)`							
`lw r5,-8(r1)`							
`mult r6,r5,r3`	WR						
`add r5,r6,r3`	ROB	WB					
`add r6,r4,r3`		WB					
`sw 0(r1),r6`		WB					
`sw -8(r1),r5`	EX	ROB	WB				
`sub r2,r2,#16`			WB				

Tabla 6: Emisión desordenada del código de las mejoras Ejercicio 0.1.2.

1. Indique qué mejoras realizaría en el procesador para reducir el tiempo de ejecución en la mejor de las opciones sin cambiar el diseño de las unidades funcionales (multiplicador, sumador, etc.) y sin cambiar el tipo de memorias ni la interfaz entre procesador y memoria (no varía el número de instrucciones captadas por ciclo).

   En primer lugar, consideramos que se decodifican el mismo número de instrucciones que se captan, eliminante el cuello de botella en la decodificación que teníamos en el problema anterior. Además, consideramos que no hay límite en el número de instrucciones por ciclo que se emiten, escriben en el ROB o completan. Por último, consideramos que disponemos de tantas unidades funcionales como se necesiten, para evitar así riesgos estructurales. Con estas mejoras, hemos llegado a la tabla 6.

2. ¿Qué pasaría si además se reduce el tiempo de multiplicación a la mitad?

   Resuelto en la tabla 7. En este caso, tenemos una latencia inicial de 6 ciclos de reloj. Sabiendo que se ejecutan 11 instrucciones en 12 ciclos de reloj, tenemos que:

   $$T(n) = 12 = TLI + (n-1)CPI = 6 + (11-1)CPI \Rightarrow CPI = \frac{12-6}{11-1} = \frac{6}{10} = 0{,}6$$

   Como $CPI = 0{,}6 < 1$, podemos decir que el procesador es superescalar, pero no se trata de su mejor versión, ya que podemos llegar incluso a 3 instrucciones por ciclo.

Ejercicio 0.1.4. En el caso descrito en el problema 0.1.3, indique cómo evolucionaría el buffer de reorden utilizado para implementar finalización ordenada, en la mejor de las opciones.

Instrucción / Ciclos	1	2	3	4	5	6	7	8	9	10	11	12
`lw r3,0x10a`	IF	ID	EX	EX	ROB	WB						
`addi r2,r0,#128`	IF	ID	EX	ROB		WB						
`add r1,r0,0x0a`	IF	ID	EX	ROB		WB						
`lw r4,0(r1)`	IF	ID		EX	EX	ROB	WB					
`lw r5,-8(r1)`		IF	ID	EX	EX	ROB	WB					
`mult r6,r5,r3`		IF	ID			EX	EX	EX	ROB	WR		
`add r5,r6,r3`		IF	ID						EX	ROB	WR	
`add r6,r4,r3`		IF	ID			EX	ROB				WR	
`sw 0(r1),r6`			IF	ID			EX	ROB			WR	
`sw -8(r1),r5`			IF	ID						EX	ROB	WB
`sub r2,r2,#16`			IF	ID	EX							WB

Tabla 7: Emisión desordenada reduciendo la multiplicación del Ejercicio 0.1.2.

Ejercicio 0.1.5. En un procesador superescalar con renombramiento de registros se utiliza un buffer de renombramiento para implementar el mismo. Indique como evolucionarían los registros de renombramiento al realizar el renombramiento para las instrucciones del Código 3.

```
1  mul r2, r0, r1   ; r2 <-- r0*r1
2  add r3,r1,r2     ; r3 <-- r1+r2
3  sub r2,r0,r1     ; r2 <-- r0-r1
4  add r3,r3,r2     ; r3 <-- r3+r2
```

Código fuente 3: Instrucciones para renombrar

Ejercicio 0.1.6. Considere el bucle del Código 4.

```
1  i=1;
2  do
3  {
4      b[i]=a[i]*c;
5      c=c+1;
6      if (c>10) then goto etiqueta; // 1
7      i=i+1;
8  } while (i<=10); // 2
9  etiqueta: //..
```

Código fuente 4: Bucle a considerar.

Indique cuál es la penalización efectiva debida a los saltos, en función del valor inicial de `c` (número entero), considerando que el procesador utiliza:

1. Predicción fija (siempre se considera que se va a producir el salto).
2. Predicción estática (si el desplazamiento es negativo se toma y si es positivo no).
3. Predicción dinámica con un bit (`1=Saltar`; `0=No Saltar`; Inicialmente está a 1).

Nota: La penalización por saltos incorrectamente predichos es de 4 ciclos y para los saltos correctamente predichos es 0 ciclos.

Ejercicio 0.1.7. En la situación descrita en el problema 0.1.6. ¿Cuál de los tres esquemas es más eficaz por término medio si hay un 25 % de probabilidades de que c sea menor o igual a 0, un 30 % de que sea mayor o igual a 10; y un 45 % de que sea cualquier número entre 1 y 9, siendo todos equiprobables?

Ejercicio 0.1.8. En un programa, una instrucción de salto condicional (a una dirección de salto anterior) dada tiene el siguiente comportamiento en una ejecución de dicho programa:

$$SSNNNSSNSNSNSSSSSN$$

donde S indica que se produce el salto y N que no. Indique la penalización efectiva que se introduce si se utiliza:

1. Predicción fija (siempre se considera que se no se va a producir el salto).
2. Predicción estática (si el desplazamiento es negativo se toma y si es positivo no).
3. Predicción dinámica con dos bits, inicialmente en el estado (11).
4. Predicción dinámica con tres bits, inicialmente en el estado (111).

Nota: La penalización por saltos incorrectamente predichos es de 5 ciclos y para los saltos correctamente predichos es 0 ciclos.

0.1.1. Cuestiones

Cuestión 0.1.1. ¿Qué tienen en común un procesador superescalar y uno VLIW? ¿En qué se diferencian?

Cuestión 0.1.2. ¿Qué es un buffer de renombramiento? ¿Qué es un buffer de reordenamiento? ¿Existe alguna relación entre ambos?

Cuestión 0.1.3. ¿Qué es una ventana de instrucciones? ¿Y una estación de reserva? ¿Existe alguna relación entre ellas?

Cuestión 0.1.4. ¿Qué utilidad tiene la predicación de instrucciones? ¿Es exclusiva de los procesadores VLIW?

Cuestión 0.1.5. ¿En qué momento se lleva a cabo la predicción estática de saltos condiciones? ¿Se puede aprovechar la predicción estática en un procesador con predicción dinámica de saltos?

Cuestión 0.1.6. ¿Qué procesadores dependen más de la capacidad del compilador, un procesador superescalar o uno VLIW?

Cuestión 0.1.7. ¿Qué procesadores tienen microarquitecturas con mayor complejidad hardware, los superescalares o los VLIW? Indique algún recurso que esté presente en un procesador superescalar y no sea necesario en uno VLIW.

Haga una lista con 5 microprocesadores superescalares o VLIW que se hayan comercializado en los últimos 5 años indicando alguna de sus características (frecuencias, núcleos, tamaño de cache, etc.).