Arquitectura de Computadores - 22GIIN

Actividad 4 - Portafolio

Gagliardo Miguel Angel

01 de Febrero de 2024

EJERCICIO 1

a) Identificar únicamente aquellas instrucciones donde ocurren dependencia verdadera de datos y explique las razones.

Para iniciar, recordemos que la **dependencia verdadera de datos** es aquella se produce entre dos instrucciones, cuando una instrucción trata de leer un operando antes que sea escrito por la otra instrucción (**RAW** - **R**ead **A**fter **W**rite).

Hecha esta definición, las instrucciones donde ocurren dependencia verdadera de datos son:

- **I3:** Depende del resultado de I1 e I2, ya que utiliza el valor cargado en **R1 y R3**.
- **I4:** Depende del resultado de I3, ya que almacena el valor de **R5** en memoria (**Mem[R0]**)
- **I6:** Depende del resultado de I1 e I5, ya que utiliza el valor cargado en **R1** y **R7**.
- **I7:** Depende del resultado de I6, ya que almacena el valor de **R8** en Memoria (**Mem[R0+16]**).

b) Muestre cómo se comporta el cauce, sin resolver la dependencia verdadera de datos. Indique los ciclos de retraso.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
I1	F1	D1	E1	M1	**W1 (R1)**														
I2		F2	D2	E2	M2	**W2 (R3)**													
I3			F3	F3	F3	D3	E3 (ADD)	M3	**W3 (R5)**										
I4				x	x	F4	F4	F4	D4	E4	M4	W4							
I5						x	x	x	F5	D5	E5	M5	**W5 (R7)**						
I6										F6	F6	F6	D6	**E6 (ADD)**	M6	**W6 (R8)**			
I7											x	x	F7	F7	F7	D7	E7	M7	W7

Figura 1. Comportamiento del cauce sin resolver la dependencia verdadera de datos

En la *Figura 1* vemos cómo se comporta el cauce.

- Las **celdas con una cruz en rojo** indican la detención de las instrucciones, dado que se necesita un dato que aún no ha sido escrito por la instrucción anterior.
- Las celdas en color **amarillo claro** indican la detención de las instrucciones en su primer ciclo de **Extraccion de la Funcion (F)**.

Podemos ver que los **ciclos de retraso** es **9**, mientras que la **cantidad total de ciclos es de 19.**

c) ¿Indique cómo resolvería cada una de las dependencias de datos? Explique, justifique y razone la técnica y/o estrategia utilizada para la solución.

Para este caso en particular, dado que hay varias opciones y que el ejercicio no especifica preferencia, podemos resolver las dependencias insertando la instrucción **NOP (No Operación)** entre las operaciones dependientes.

Finalmente, las instrucciones quedarán así:

ID Instrucción	Instrucciones	Semántica
I1.	ld #R1, 10(#R0);	R1 = Mem[R0+10]
I2.	ld #R3, 4(#R0);	R3 = Mem[R0+4]
I3.	**NOP;**	
I4.	**NOP;**	
I5.	add #R5, #R1, #R3;	R5 = R1 + R3
I6.	**NOP;**	
I7.	**NOP;**	
I8.	st #R5, 0(#R0);	Mem[R0] = R5
I9.	ld #R7, 8(#R0);	R7 = Mem[R0+8]
I10.	**NOP;**	
I11.	**NOP;**	
I12.	add #R8, #R1, #R7;	R8 = R1 + R7
I13.	**NOP;**	
I14.	**NOP;**	
I15.	st #R8, 16(#R0);	Mem[R0+16] = R8

Tabla 1. Instrucciones una vez insertada la instrucción NOP

Si bien es verdad que **NOP** no es la única opción, por ejemplo también se podrían **reordenar las instrucciones** o **anticipar**, para este caso en particular he elegido **NOP** por una cuestión de facilidad en su implementación. **NOP** es sin dudas la opción más sencilla de implementar a la hora de resolver las dependencias, pero no la más eficiente. Respecto a esto último cabe resaltar que así como su implementación es es más fácil, también aumenta la cantidad final de instrucciones a 15, un aumento del ~114% (de 7 a 15 instrucciones). Haciendo menos eficiente el programa.

d) Muestre cómo se comporta el cauce**, luego de resolver los problemas de dependencia verdadera de datos. Indique los ciclos de retraso**

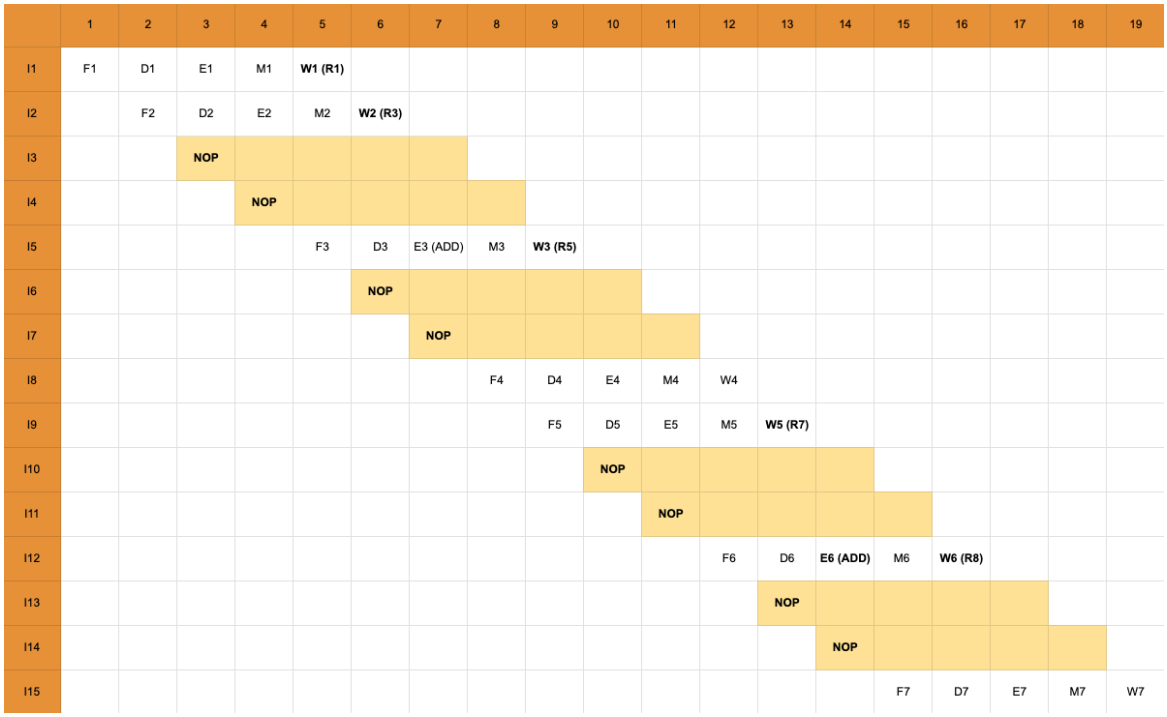

Figura 2. Comportamiento del cauce al salvar la dependencia usando NOP

Luego de insertar la instrucción **NOP** , se visualiza como el cauce puede ahora avanzar sin esperar por las dependencias de datos.

Por último tal como se observa en la Como vemos, **la cantidad de ciclos de reloj totales sigue siendo de 19.**

EJERCICIO 2

Se pide:

- **Resolver la dependencia de control (salto condicional) de manera que se mejore el rendimiento del cauce.**
- **Indique cuál técnica utiliza y muestre cómo se comporta el cauce.**

Para resolver la dependencia de control y mejorar el rendimiento del cauce, podemos utilizar la **técnica de predicción de salto** y en particular asumiendo que **el salto no es tomado**.

En este caso, la instrucción I4 es una instrucción de salto condicional a la etiqueta **SubRut1** (I2). Para mejorar el rendimiento, **podemos asumir que el salto no es tomado** y continuar ejecutando instrucciones en ese camino.

Así se comportaría el cauce utilizando esta técnica. Se observan un total de **10 ciclos de reloj**

	1	2	3	4	5	6	7	8	9	10
I1	F1	D1	E1 (MUL)	M1	**W1 (R3)**					
I2		F2	D2	E2 (SUB)	M2	**W2 (R5)**				
I3			F3	D3	E3 (DIV)	M3	**W3 (R**			
I4				F4	D4	E4	M4	**W4 (WB)**		
I5					F5	D5	E5	M5	**W5 (R8)**	
I6						F6	D6	E6	M6	**W6 (R3)**

Figura 3. Comportamiento del cauce al resolver la dependencia de control

BIBLIOGRAFIA UTILIZADA

Figueras, G. E. (2019). ARQUITECTURA DE COMPUTADORES. Manual del curso.

Universidad Internacional de Valencia. **Tema 5 y Tema 6**