Arquitectura de Computadoras I

Ingeniería de Sistemas – Curso 2013

Práctica 3: Riesgos en procesadores segmentados

1. ¿Cuáles de las instrucciones siguientes no corresponden a una arquitectura RISC? En su caso sustituya la instrucción por la instrucción o secuencia de instrucciones RISC que realicen la misma función.

```
ADD R3, R28, R32    ; R3<= R28+ R32
ST R2, 4(R27)       ; M[R27+4] <= R2
ADD R4, (R6), $34   ; R4 <= M[R6] + M [34]
LD R17, (R21)       ; R17 <= M[R21]
```

2 - Proponga traducciones a MIPS para las siguientes instrucciones simples escritas en lenguaje C:

a) a = b + 777 ; con a ≡ t2 y b ≡ t5

b) x[20] = x[21] + c ; con c ≡ t3 y x[0] ≡ memoria[100]

c) x[i] = x[i-1] + c ; con i ≡ t7, c ≡ t5 y x[0] ≡ memoria[100]

d) - - i ; con i ≡ t1

3. Se tiene un procesador segmentado con ejecución en orden (emisión y finalización en orden). La segmentación se produce en 5 etapas: (1) captura de instrucción, (2) decodificación y lectura de operandos de registros, (3) ejecución, (4) acceso a memoria de datos, (5) escritura en registros. Dicho procesador no dispone de adelantamiento de datos. Para cierto programa, el porcentaje de uso de cada tipo de instrucción es el reflejado en la siguiente tabla:

Tipo instr.	ENTEROS	SALTOS	LOAD	STORE
% uso	70%	5%	15%	10%

Con dicho programa y procesador se ha medido un CPI promedio de 3. Se pretende añadir adelantamiento de datos para reducir el CPI.

a) El adelantamiento de datos se aplica sólo si el dato lo genera una instrucción con enteros. Suponga que hay riesgos RAW con instrucciones en la siguiente posición de memoria el 20% de las veces y con instrucciones dos posiciones de memoria después el 15% de las veces. Por simplicidad, se puede suponer que ambas circunstancias nunca suceden a la vez. Calcular el nuevo CPI.

b) El adelantamiento de datos se aplica sólo a los datos obtenidos con cargas (LOAD). Suponga que en todos los casos de adelantamiento de datos se tiene éxito en caché (no hay demoras en memoria de datos) y que hay riesgos RAW con instrucciones en la siguiente posición de memoria el 20% de las veces y con instrucciones dos posiciones de memoria después el 15% de las veces. Por simplicidad, se puede suponer que ambas circunstancias nunca suceden a la vez. Calcular el nuevo CPI.

c) Aplicar ambas mejoras simultáneamente y calcular el nuevo CPI.

4. Un procesador sin segmentar necesita 200 ns para ejecutar una instrucción. Sin considerar el periodo de llenado del procesador, calcular la aceleración en los siguientes casos:

- **a)** Un procesador ideal con 5 estados, en el que todos los estados utilizan el mismo tiempo. Debido a la segmentación, cada estado incorpora 4 ns adicionales de retardo en el tiempo de ejecución. No existen paradas debidas a problemas de saltos o de dependencia de datos.
- **b)** En un procesador más real también de 5 estados. Cada estado necesita 30 ns, 30 ns, 40 ns, 50 ns y 50 ns respectivamente. Al igual que en el caso anterior la segmentación añade un retardo de 4 ns por estado.
- **c)** Considerando e aso b), ahora se tiene que debido a los riesgos de datos y control en el 20% de las instrucciones se produce una parada de un ciclo de reloj y en el 5% de dos ciclos.

5. En la tabla adjunta se muestra la estadística de uso de un conjunto de instrucciones para un determinado procesador con estructura Harvard.

Tipo Instrucción	**LOAD/STORE**	**Saltos Condicionales**	**Otras**
Promedio de uso	15%	25%	60%

El procesador está segmentado en seis etapas. 1) Captura instrucción, 2) Decodificación y detección de riesgos, 3) Captura de operandos y cálculo de la dirección efectiva en saltos. 4) Ejecución, cálculo de la dirección efectiva en accesos a memoria y en su caso cálculo de la condición. 5) Acceso a memoria y 6) Escritura de resultados en el banco de registros. Se supone que los únicos riesgos que reducen el rendimiento ideal del sistema (CPI = 1) son lo riesgos de control. Todas las instrucciones pasan por todos los segmentos. Se pide:

- **a)** Sabiendo que el 80% de los saltos son efectivos, indicar el CPI real del sistema sabiendo que al detectar un riesgo, el procesador se detiene hasta que se soluciona. Utilizar el diagrama adjunto.
- **b)** Utilizando necesariamente las variables definidas por la ley de Amdahl calcular la mejora que se produce si, se diseña un sistema de predicción estática en donde siempre predice efectivo.

6. Un procesador segmentado con 5 etapas

IF: Lectura de instrucción

ID: Decodificación de instrucción y lectura de operandos

EX: Ejecución de operación y cálculo de la dirección efectiva.

DM: Acceso a la memoria

WB: Escritura en los registros. (Asumir que se puede escribir y leer en el mismo ciclo)

Ejecuta el siguiente código

Instrucción	Código	Función
1	LW R1,2000 (R0)	R1 ← M[R0+2000]
2	LW R2,2004 (R0)	R2 ← M[R0+2004]
3	ADD R3,R2,R1	R3 ← R2+R1
4	ADDI R1,R2,8	R1 ← R2+8
5	SUBI R4,R0,2	R4 ← R0-2
6	AND R5,R3,R2	R5 ← R3 and R2

7	SW R4,2000 (R0)	M[R0+2000] ← R4
8	SW R5,2004 (R0)	M[R0+2004] ← R5
9	SW R1,2008 (R0)	M[R0+2008] ← R1

a) Identificar, señalando el tipo, todos los posibles conflictos que pueden surgir por dependencias entre operandos.
b) Suponiendo que el procesador no tiene la posibilidad de parar el pipeline, insertar en el programa fuente el mínimo número de instrucciones NOP que se consideren necesarias para eliminar las dependencias de datos. Asumir que el procesador no dispone de mecanismos de "Forwarding" (avance de resultado en la salida de la ALU) y que no se puede cambiar el orden de la secuencia de instrucciones
c) Representar gráficamente paso a paso el estado del *pipeline*. Rellenar el cuadro adjunto de la misma manera a como se muestra para la primera instrucción.
d) ¿Cuál es la mejora producida por utilizar segmentación?

7. Se tiene un microprocesador segmentado en las siguientes 5 etapas. **IF**: captura de instrucción. **ID**: decodificación de instrucción, detección de riesgos y captura de operandos. **EX**: ejecución en ALU, cálculo de la dirección de salto y de la dirección de acceso a memoria de datos. **M**: acceso a memoria de datos y resolución de la condición de salto. **W**: escritura en registros internos. El banco de registros es tal que permite en el mismo ciclo de reloj escribir un dato y luego leerlo. Se utiliza arquitectura Harvard y no hay detenciones en memoria. La ejecución es siempre en orden. El código a ejecutar es el siguiente, en el que se sabe que el salto condicional es efectivo:

Instr./Etiq.	**Ensamblador**		*Explicación*
I1	**ADD**	**R1, R2, R3**	*; R1 = R2+R3*
I2	**LOAD**	**R4, 0(R1)**	*; R4 = M[0+R1]*
I3	**MUL**	**R5, R4, R4**	*; R5 = R4·R4*
I4	**BNE**	**R1, R4, L1**	*; Si R1 /= R4, salta a L1 (sí se salta)*
I5	**SUB**	**R5, R5, R2**	*; R5 = R5-R2*
I6	**ADD**	**R7, R4, R5**	*; R7 = R4+R5*
I7 / L1:	**ADD**	**R6, R1, R5**	*; R6 = R1+R5*
I8:	**OR**	**R1, R7, R8**	*; R1 = R7 or R8*
I9:	**ST**	**R6, 0(R1)**	*; M[0+R1] = R6*

a) Suponiendo que no hay adelantamiento de datos (*forwarding*) y que los saltos condicionales se predicen como no efectivos, rellenar el siguiente cronograma.

Instrucciones		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
ADD	R1, R2, R3	IF	ID	EX	M	W																			
LOAD	R4, 0(R1)																								
MUL	R5, R4, R4																								
BNE	R1, R4, L1																								
SUB	R5, R5, R2																								
ADD	R7, R4, R5																								
L1: ADD	R6, R1, R5																								
OR	R1, R7, R8																								
ST	R6, 0(R1)																								

b) Suponiendo que hay adelantamiento de datos implementado de la mejor forma posible y que los saltos condicionales se predicen como efectivos, rellenar el siguiente cronograma. Indique sobre el mismo los adelantamientos de datos realizados (flecha de la unidad que genera el dato a la que lo utiliza).

Instrucciones		01	02	03	04	05	06	07	08	09	10	11	1	13	14	1	16	17	1	19	20	21	22	23	24
ADD	R1, R2, R3	IF	ID	EX	M	W																			
LOAD	R4, 0(R1)																								
MUL	R5, R4, R4																								
BNE	R1, R4, L1																								
SUB	R5, R5, R2																								
ADD	R7, R4, R5																								
L1: ADD	R6, R1, R5																								
OR	R1, R7, R8																								
ST	R6, 0(R1)																								

8. Se tiene un sistema ordenador con un procesador segmentado en 4 etapas. 1) Captura instrucciones (**CI**): se lee la instrucción a ejecutar, 2) Decodificación (**DE**): se decodifica la instrucción, se leen los operandos de los registros internos, se calcula la dirección efectiva en las instrucciones con memoria y en su caso se calcula la dirección de salto. 3) Ejecución (**EJ**): se ejecuta la operación aritmética o lógica, se ejecuta la lectura en memoria y se determina si los saltos son o no efectivos. *La fase de ejecución de las instrucciones se realiza en un ciclo para todas las instrucciones excepto las de coma flotante, que necesitan 4 ciclos. En este caso la ALU queda ocupada durante los cuatro ciclos y se producen las detenciones pertinentes.* 4) Escritura (**W**): se escribe el resultado en un registro interno o, en su caso, se almacena el dato en memoria. Suponer que no existen riesgos de datos y que con respecto a los riesgos de control la estrategia es detener la CPU hasta resolver el riesgo.
Estadísticamente, el porcentaje de instrucciones empleadas en un programa se indica en la tabla adjunta.

Load/Store	Saltos condicionales	Aritmética con Enteros	Aritmética pto. flotante
20%	15%	45%	20%

NOTA: Para calcular las distintas mejoras, no tener en cuenta el tiempo de llenado del procesador y cuando corresponda, suponer que el 100% de los saltos son efectivos (el peor caso posible).
Se realizan dos mejoras consecutivas:

a) Se duplica el bus y de tener un bus común para instrucciones y datos se diseña una arquitectura tipo Harvard.
b) Se rediseña la unidad de coma flotante, de tal forma que se reduce a al mitad el tiempo de ejecución para estas instrucciones.

Se pide, aplicando necesariamente la ley de Amdahl definir la ganancia obtenida tras las dos mejoras**.**