Arquitectura e Ingeniería de Computadores
Problemas (hoja 2). Curso 2009-10

1. El siguiente fragmento de código se ejecuta en un DLX con segmentación:

```
SUB R1,R2,R3
ADD R4,R5,R6
SUB R5,R4,R8
ADD R7,R2,R3
ADD R9,R7,R3
LW  R1,10(R6)
ADD R3,R1,R4
SUB R6,R7,R8
```

Suponiendo que un dato se puede escribir en un banco de registros y leer su nuevo valor en el mismo ciclo:

(a) Calcular el número de ciclos necesarios para ejecutar este código si no existe la posibilidad de anticipar operandos o reordenar el código.
(b) Calcular el número de ciclos si existe anticipación de operandos, pero no reordenación de código.
(c) Tratar de reordenar el código para conseguir que el número de ciclos sea mínimo, si no hay anticipación de operandos. ¿Cuántos ciclos son necesarios en este caso?.

2. Se tiene el siguiente fragmento de código del MIPS.

```
OR R4,R8,R9
(100 instrucciones con dependencias internas de datos, pero no detención de pipeline)
ADD R5,R6,R7
OR  R4,R1,R6
(100 instrucciones con dependencias internas de datos, pero no detención de pipeline)
LW  R10,20(R4)
ADD R8,R9,R10
SUB R6,R8,R1
OR  R1,R3,R5
BEQ R1,R2,1000
ADD R4,R5,R6
```

Nota: El destino del salto es la instrucción "`OR R4,R1,R6`", y el salto se repite 100 veces.

(a) Cuánto tardaría en ejecutarse, si cuando hay un conflicto de control se espera a que se solucione y se supone que el destino de salto se conoce en la fase ID y la condición del salto también. Suponer que no hay anticipación de operandos.
(b) Si se tuviesen saltos retardados, ¿cómo se podría rellenar el hueco de salto para disminuir el tiempo de ejecución? ¿Cuál sería este tiempo?
(c) ¿y si además de salto retardados tuviésemos anticipación de operandos?

3. Supongamos que el siguiente fragmento de código está listo para ejecutar en un computador similar al DLX, pero con memoria común de instrucciones y datos y *branch delay-slot* de una instrucción:

```
      ...
loop: LW   R1,A(R4)
      LW   R2,B(R4)
      ADD  R1,R1,R2
      SUB  R1,R1,R3
      SUB  R4,R4,#4
      SW   C(R4),R1
      BNEZ R4,loop
```

```
ADD   R3,R3,#4
ADD   R1,R1,R5
...
```

El valor inicial del registro R4 es 1000. Se pide:

(a) ¿Cuál es el valor de los CPI (ciclos por instrucción) obtenidos al ejecutar el programa?
(b) Si la frecuencia de funcionamiento es de 20 MHz, ¿Cuál es el valor de los MIPS? ¿Y el tiempo de ejecución?

4. Un computador tiene el ciclo de instrucción segmentado en 7 etapas:

- **IF, IS**: Búsqueda de la instrucción.
- **RF**: Lectura de operandos y decodificación de la instrucción.
- **EX**: Cálculo de la dirección efectiva, operación en la ALU y condición de saltos.
- **DF, DS**: Acceso a memoria.
- **WB**: Escritura del resultado en el registro destino.

La máquina aplica la técnica del cortocircuito para resolver los riesgos de datos.

(a) ¿Cuánto vale el *branch delay slo*t?
(b) Considera la secuencia de instrucciones siguiente:

```
ADD R1,R2,R3
SUB R4,R1,R5
ADD R6,R1,R7
SUB R8,R1,R9
ADD R10,R1,R11
SUB R12,R1,R13
ADD R14,R1,R15
```

Indica los cortocircuitos necesarios para resolver todos los riesgos de datos que se produzcan.

5. Supongamos que la carga de trabajo del DLX segmentado viene caracterizada por tres aplicaciones A, B y C. Supongamos que el porcentaje de instrucciones que sufren un ciclo de reloj de penalización en cada aplicación es el siguiente:
Aplicación A: 9%; aplicación B: 14%; aplicación C: 11%.
Determinar:

(a) El número promedio de ciclos de reloj por instrucción.
(b) El valor del Speedup, frente a la máquina no segmentada.
(c) La Eficiencia.

6. El siguiente bucle en código ensamblador del DLX ejecuta el cálculo de la operación vectorial Y= a*X + Y, donde a es un escalar y X e Y son vectores:

```
loop: LD    F2,0(R1)          ; cargar X(i)
      MULTD F4,F2,F0          ; a*X(i)
      LD    F6,0(R2)          ; cargar Y(i)
      ADDD  F6,F4,F6          ; aX(i)+Y(i)
      SD    0(R2),F6          ; salvar Y(i)
      ADDI  R1,R1,8           ; incr. índice X
      ADDI  R2,R2,8           ; incr. índice Y
      SGTI  R3,R1,long        ; final ?
      BEQZ  R3,loop
```

Supongamos que los tiempos de cálculo de las operaciones en punto flotantes son: suma, 2 ciclos; multiplicación, 5 ciclos. Determinar el ciclo de reloj en que se lanza a ejecución cada una de las instrucciones en la primera iteración del bucle. Se supone que la máquina no tiene ningún mecanismo de planificación dinámica de instrucciones.

7. Consideremos la organización básica de un DLX segmentado en 5 etapas, en donde cada etapa opera en un ciclo de reloj. Supongamos que su carga de trabajo es tal que:

- En el 20% de los casos la instrucción i+1 presenta una dependencia LDE respecto de la i. En la quinta parte de estas situaciones la instrucción i es de carga.
- En otro 10% de los casos la instrucción i+2, pero no la i+1, presenta una dependencia LDE respecto de la i.

- En otro 5% de los casos la instrucción i+3, pero no la i+2, ni la i+1, presenta una dependencia LDE respecto de la i.
- El 15% de las instrucciones son de salto.
- El 60% de los saltos se toman.

Determinar en promedio el nº de ciclos de reloj que se precisan para ejecutar cada instrucción, en cada uno de los siguientes supuestos:

(a) Con la organización básica del DLX (sin anticipación de operandos, ni aceleración de saltos).
(b) Si se modifica la estructura básica para implementar la anticipación de operandos (forwarding).
(c) Si además se añade el HW necesario para resolver los saltos en la etapa de decodificación y se asume la predicción de que los saltos no se toman (predict-not-taken).
(d) Si con la estructura HW del apartado anterior, se implementa una política de saltos retardados, donde en el 60% de los casos el hueco de retardo (delay slot) se rellena con una instrucción previa al salto, en el 25% con una instrucción del destino del salto y en el resto con NOP.

8. Supongamos que la versión básica del DLX para instrucciones enteras segmentado en cinco etapas (con cortocircuitos y resolución de saltos en la segunda etapa) se modifica de la siguiente forma:

- Las etapas EX y MEM se funden en una sola.
- Como consecuencia la duración del ciclo de instrucciones debe incrementarse en un 50%.

Hallar la relación de tiempos de ejecución entre la versión original del DLX y la modificada. Supóngase que en el 4% de las instrucciones son instrucciones de carga seguidas de un uso inmediato del operando cargado, y que el 5% de las instrucciones son saltos condicionales donde la política de tratamiento de saltos no permite realizar trabajo útil en el "delay slot".

9. Se dispone de un procesador DLX con los siguientes operadores multiciclo:

- Sumador/Restador segmentado lineal. Latencia 2 ciclos.
- Multiplicador segmentado lineal. Latencia 3 ciclos.
- Divisor convencional. No segmentado. Latencia 4 ciclos.

Para la resolución de los riesgos estructurales y de datos se plantean las dos alternativas siguientes:

A. Riesgos estructurales: detección y espera en la fase ID. Riesgos RAW: detección y espera en fase ID y cortocircuito. Riesgos WAW: detección y espera en la fase ID.
B. Riesgos estructurales: detección y espera en la última fase del operador. Riesgos RAW: detección y espera en la primera fase del operador y cortocircuito. Riesgos WAW: inhibición de la escritura.

En ambos casos, se previenen los riesgos WAR leyendo los operandos disponibles en la fase ID. Mostrar, para ambas alternativas, el diagrama de ejecución del siguiente fragmento de código, representando las etapas que atraviesan cada una de las instrucciones, utilizando la siguiente notación: IF fase de búsqueda, ID fase de decodificación, EX fase de ejecución monociclo, A1, A2 fases de ejecución del sumador/restador, M1, M2, M3 fases de ejecución del multiplicador, D1, D2, D3, D4 fases de ejecución del divisor, ME fase de acceso a memoria y WB fase de escritura en registros, o IW en caso de que la escritura esté inhibida.

```
LD    F1,0(R1)
DIVD  F4,F0,F1
ADDD  F2,F3,F4
LD    F4,4(R1)
MULTD F3,F4,F2
LD    F5,8(R1)
```

(a) Propuesta A.

Instruc	1	2	3	4	5	6	7	8	9	10	11	…	….	…
LD F1, 0(R1)														
DIVD F4,F0,F1														
ADDD F2,F3,F4														
LD F4,4 (R1)														
MULD F3,F4,F2														
LD F5,8(R1)														

Número de ciclos de ejecución:

(b) Propuesta B.

Instruc	1	2	3	4	5	6	7	8	9	10	11	…	….	…
LD F1, 0(R1)														
DIVD F4,F0,F1														
ADDD F2,F3,F4														
LD F4,4 (R1)														
MULD F3,F4,F2														
LD F5,8(R1)														

Número de ciclos de ejecución:

10. Al diseñar el DLX segmentado, los arquitectos han de decidir en qué fase escriben el PC las instrucciones de bifurcación: ID o EX. Se utiliza la técnica del salto retardado. La frecuencia de reloj es 10 % más rápida si el PC se escribe en la etapa EX, pero el *branch delay slot* es mayor. Para la generación de código se va a adaptar un viejo compilador para DLX no segmentado que crea programas con un 10 % de instrucciones de bifurcación. A este compilador se le añade un módulo de optimización que intenta reordenar el código para colocar instrucciones útiles en el *branch delay slo*t, y si no lo consigue, inserta tantas instrucciones NOP como haga falta. La tasa de éxito del optimizador es:

- 30 % de los casos no encuentra ninguna instrucción, insertando las instrucciones NOP necesarias.
- 50 % de los casos encuentra una sola instrucción capaz de ocupar el *branch delay slo*t.
- 20 % de los casos encuentra dos instrucciones capaces de ocupar el *branch delay slo*t.

Suponiendo que el CPI es el mismo en ambas alternativas, evalúa qué opción es más interesante.

11. Considérese la sentencia de PASCAL:

```
var v: array[0..dim-1] of integer;
...
      for i:= 0 to n-1 do v[i]:=v[i]+5;
```

que un compilador no optimizante ha traducido al siguiente código para DLX:

```
      LW  Ri,n
      SHL Ri,Ri,#2
bu:   SUB Ri,Ri,#4
      LW  Rv,v[Ri]
      ADD Rv,Rv,#5
      SW  v[Ri],Rv
      BNEZ Ri,bu
      NOP
```

Se dispone de un DLX , con cortocircuito y detección de riesgos y reloj a 100 MHz. Se pide:

(a) El CPI en régimen estacionario cuando DLX está ejecutando este bucle.

(b) Modificar el programa para aprovechar el *branch-slot* con la mejor de las estrategias. Calcular los CPI y MIPS en régimen estacionario. Expresar, en tanto por cien, la mejora en prestaciones respecto del programa no optimizado.
(c) Evaluar el siguiente proyecto de mejora en el diseño del DLX: eliminar los cortocircuitos y la lógica de detección de *cualquier* riesgo de datos para así incrementar en un 50 % la frecuencia de reloj. Naturalmente, el compilador deberá tenerlo en cuenta e insertar instrucciones `NOP` en donde corresponda. Lectura y escritura de registros se lleva a cabo en distintos semiciclos: primero escrituras y luego lecturas. Evaluar este proyecto utilizando el código del apartado (a)

APÉNDICE (Ejemplo de introducción al WinDLX).- Utiliza el siguiente código para probar el funcionamiento del simulador WinDLX.
a) Ejecútalo ciclo a ciclo tal como está. Explica las paradas que se produzcan.
b) Modifica el código para que el salto condicional se tome y observa su efecto. ¿Qué tratamiento se da a los saltos condicionales?
c) Prueba a eliminar el cortocircuito (menú de configuración) y observa ahora el comportamiento ciclo a ciclo. Explica el comportamiento en cada situación de riesgo.

```
                .data 0
                ;comienzo de los datos
                .word 6,6
                ;
                .text 0x100
                ;comienzo del código
                .global start
start:          addi        r6,r0,#23       ; inicializa r6 a 23
                addi        r2,r0,#0        ; inicializa r2 a 0
                lw          r1,4(r2)
                addi        r2,r2,#3
                sub         r3,r4,r5
                beqz        r6,sal
                multf       f8,f9,f10
                add         r10,r11,r12
                addf        f8,f9,f10
sal:            add         r10,r11,r12
fin:            trap 0                      ; finalizar el programa
```