Arquitectura e Ingeniería de Computadores
Problemas (hoja 3). Curso 2006-07

1. Sea la siguiente secuencia de código de instrucciones en punto flotante para un computador similar al DLX que aplica gestión dinámica de instrucciones basada en el algoritmo de Tomasulo:

```
LD    F6,x(R1)
LD    F2,y(R1)
MULTD F0,F2,F4
SUBD  F8,F6,F2
DIVD  F6,F0,F6
ADDD  F1O,F0,F6
ADDD  F6,F8,F2
SD    z(R1),F6
```

Indica el estado de la unidad de ejecución en punto flotante en el momento en que todas las instrucciones han sido lanzadas a ejecución en la misma, si la latencia de la memoria es tal que sólo la primera se ha ejecutado completamente, habiéndose cargado en F6 el dato x. Suponer que el resto de registros Fi, poseen un valor dado por la expresión [Fi].

2. Aplica el método de Tomasulo a la siguiente versión del bucle AXPY para el procesador DLX funcionando a 80MHz. Las etapas del ciclo de instrucción para las instrucciones multiciclo son:

- **IF** (búsqueda),
- **ID** (decodificación),
- **Issue** (lanzamiento a ejecución, aplicando Tomasulo),
- **EX** (ejecución en el operador multiciclo) y
- **WB** (escritura en el bus común de datos).

Todos los operadores multiciclo están segmentados y las latencias son las siguientes: 2 ciclos para load/store, 7 ciclos para suma/resta y 10 ciclos para multiplicación/división.

```
bucle:      LD          F2,x(R1)
            MULTD       F4,F2,F0
            LD          F6,y(R1)
            ADDD        F6,F4,F6
            SD          y(R1),F6
            BNEZ        R1,bucle
            SUBI        R1,R1,#8
```

Suponiendo que en el mismo ciclo se puede escribir un dato en el bus común de datos e iniciar una operación que estuviera en espera de ese dato, dibuja el diagrama instrucción/tiempo correspondiente a las dos primeras iteraciones e indica el estado en que se encuentran las estaciones de reserva, los load buffers, los store buffers y los registros, en el ciclo 16.
¿Cuál es la velocidad de ejecución sostenida del bucle en MFLOPS?

3. Cierto programa consume la mayor parte de su tiempo de ejecución en el siguiente fragmento de código:

Y = Y + X

Dicho programa se pretende ejecutar sobre dos computadores DLX y DLX/T. Las dos máquinas poseen operaciones de punto flotante en su juego de instrucciones, funcionan a una frecuencia de 100 MHz, el CPI de las instrucciones enteras es 1 y el *delay slot* es de 1 instrucción. En el DLX, las etapas que atraviesa una instrucción de coma flotante son: **IF** (búsqueda de la instrucción), **ID** (decodificación de la instrucción, lectura de registros fuente y detección de riesgos), **EX** (ejecución en el operador multiciclo correspondiente) y **WB** (escritura del resultado en el registro destino). Los riesgos de datos se detectan en ID, esperando la

instrucción implicada en dicha fase. El DLX/T posee gestión dinámica de instrucciones, aplicando el algoritmo de Tomasulo en una nueva etapa **I** (*Issue*) ubicada a continuación de ID, y escribiendo en el bus común de datos en WB (1 ciclo).
Las características de las unidades funcionales **no segmentadas** de cada máquina son las siguientes:

	FU de carga	ADDD	MULT	FU de Store
DLX	1 , 2 ciclos	1, 3 ciclos	1, 4 ciclos	1, 2 ciclos
DLXT	1, 2 ciclos, 3 buffers	1, 3 ciclos, 2 buffers	1, 4 ciclos, 2 buffers	1, 2 ciclos, 3 buffers

El código generado por el compilador es el siguiente:

```
loop: LD    F2,X(R1)
      LD    F4,Y(R1)
      ADDD  F4,F2,F4
      SD    Y(R1),F4
      BNEZ  R1,loop
      SUB   R1,R1,#8
```

Calcula el tiempo necesario, expresado en segundos, para procesar el fragmento de código en cada una de las máquinas, así como la velocidad de ejecución del mismo en MFLOPS para vectores de tamaño muy grande.

4. Un computador DLX/T funciona a una frecuencia de 100 MHz, aplica gestión dinámica de instrucciones basada en el algoritmo de Tomasulo. El *delay slot* es de 1 instrucción y todas las unidades funcionales multiciclo son segmentadas lineales, con las siguientes etapas:

	Fu de carga	ADDD	Fu de stores
DLX/T	3 etapas	4 etapas	3 etapas

Se pretende ejecutar el siguiente fragmento de código sobre dicha máquina:

```
      ; R1 = 256
loop: SUB   R1, R1, #4
      LD    F0, y(R1)
      LD    F2, z(R1)
      LD    F4, t(R1)
      ADDD  F6, F4, F0
      ADDD  F8, F2, F0
      ADDD  F6, F6, F8
      BNEZ  R1, loop
      SD    x(R1),F6
```

¿Cuántas estaciones de reserva, buffers de lectura y buffers de escritura hacen falta para ejecutarlo lanzando una instrucción por ciclo? Calcula los MFLOPS obtenidos al ejecutar el programa.

5. Es esencial que el Scoreboard sea capaz de distinguir entre riesgos RAW y WAR, puesto que los riesgos WAR exigen que se congele la instrucción que va a hacer la escritura del resultado hasta que la instrucción en conflicto lea sus operandos, mientras que un riesgo RAW requiere retardar la lectura de operandos hasta que la instrucción en conflicto haga la escritura del resultado (justo lo contrario). Por ejemplo, consideremos la secuencia:

```
MULD  F0,F6, F4
SUBD  F8,F0,F2
ADDD  F2, F10, F2
```

donde aparecen ambos tipos de riesgos. La instrucción MULD debe completar su fase de WB para que SUBD pueda hacer la lectura de operandos; sin embargo, ADDD no puede completar

su WB hasta que SUBD haya completado su lectura de operandos. Demostrar que con las estructuras de información del Scoreboard y las condiciones que se exigen para que una instrucción pase de una fase a la siguiente, los riesgos WAR y RAW se controlan correctamente.

6. Como práctica de sus asignaturas de programación, un estudiante de informática ha escrito el siguiente programa de ordenación:

```
program ordenar;
const n= ...; { tamaño del vector }
var A: array [0..n-1] of word;
i,j,temp:word;
begin
for i:=0 to n-2 do { bucle fori }
for j:= i+1 to n-1 do { bucle for2 }
      if A[i]>A[j] then begin
            temp:=A[i];
            A[i]:=A[j];
            A[j]:=temp;
      end;
end.
```

El estudiante sabe que, en este algoritmo, el bucle exterior *for* se ejecuta n - 1 ≈ n veces y que el interior *for* se ejecuta 1/ 2 *n(n – 1)≈1/2 n^2 veces, siendo n la dimensión del vector. También ha realizado algunas medidas y sabe que la condición de la sentencia *if* se satisface el 50 % de las veces.

El compilador genera código para DLX, cuyas características más notables son:

- Ciclo de instrucción segmentado en cinco etapas:
  - **IF**: Búsqueda de la instrucción.
  - **ID**: Decodificación de la instrucción. Lectura de registros fuente (2º semiciclo). Cálculo de la dirección, condición y escritura del PC (saltos).
  - **EX**: Operación (ALU). Cálculo de la dirección efectiva (carga/almacenamiento).
  - **ME**: Acceso a memoria (carga/almacenamiento).
  - **WB**: Escritura del resultado en el registro destino (1º semiciclo).
- Inserta un ciclo de parada por dependencia LDE de datos después de una instrucción de carga. El resto de dependencias LDE se resuelve mediante cortocircuito.
- Salto retardado con cancelación. Incluye cuatro instruciones nuevas:
  - beqz*T y bnez*T cancelan la instrucción que ocupa el *slot* si no se cumple la condición de salto. Son apropiadas cuando el programador (o compilador) supone que probablemente se cumplirá la condición.
  - beqz*F y bnez*F cancelan la instrucción que ocupa el *slot* si se cumple la condición de salto. Son apropiadas cuando el programador (o compilador) supone que probablemente no se cumplirá la condición.
- En ausencia de conflictos, CPI = 1.
- Reloj a 200 MHz.

El resultado de compilar el programa *ordenar* es el siguiente:

```
            ADDI R1,R0,R0     ; variable i ubicada en r1
fori:       SGTI R5,R1,4*(n-2); condición de salto bucle fori
            BNEZ*F R5,endfori ;
            ADDI R2,R1,4*1    ; variable j ubicada en r2
for2:       SGTI R5,R2,4*(n-1); condición de salto bucle for2
            BEZ*F R5,endfor2  ;
            LW   R3,A(r1)     ;
            LW   R4,A(r2)     ;


if:         SGT  R5,R3,R4     ;
```

```
            BEQZ*F R5,endif   ;
            SW   A(R2),R3     ; la optimización ha eliminado
            SW   A(R1),R4     ; la variable temp
endif: ;
            ADDI R2,R2,4*1    ; j++
            BEQZ*T R0,for2+4  ;
            SGTI R5,R2,4*(n-1); slot
endfor2: ;
            ADDI R1,R1,4*1    ; i++
            BEQZ*T R0,fori+4  ;
            SGTI R5,R1,4*(n-2); slot
endfori: ...
```

Se pide:

(a) El número de ciclos de parada por cancelación que se producirán durante la ejecución del program en función de n.
(b) El número de ciclos de parada que se producirán durante la ejecución del programa por conflictos RAW en función de n.
(c) El número de instrucciones que se ejecutarán, sin contar las que se cancelen.
(d) El valor medio del CPI.
(e) El tiempo de ejecución del programa para n = 1000.

7. Se dispone de un procesador segmentado con un juego de instrucciones de tipo entero basadas en el procesador DLX. Las fases que atraviesa una instrucción son:

- **IF:** Búsqueda de la instrucción.
- **ID:** Decodificación de la instrucción. Cálculo de la condición de salto y dirección de destino en caso de instrucción de bifurcación.
- **EX:** Fase de ejecución aritmética, y cálculo de la dirección de acceso a memoria, en su caso.
- **ME:** Acceso a memoria de datos, en su caso.
- **WB:** Escritura en registros.

Dicho procesador posee un predictor del tipo **Branch Target Buffer** de **2 bits**, que se accede durante la fase **IF**, obteniendo la respuesta al final de dicha fase.

Dada la siguiente secuencia de código, y suponiendo que el *buffer* del predictor contiene el estado **"predicción no-salta-fuerte"** para la instrucción "`BNEZ R1,loop`", mostrar las fases de las dos instrucciones que se ejecutan inmediatamente después de este salto, en cada una de las cuatro primeras iteraciones. En caso de que alguna instrucción se cancele, marcar con una '**X**' las fases que no ejecuta.

```
      ...
      ADD   R1,R0,#4
loop: SUB   R5,R5,R2
      ADD   R2,R2,#1
      SUB   R1,R1,#1
      BNEZ  R1,loop
      AND   R6,R5,#63
      OR    R6,R6,#128
      ...
```

8. Se quiere diseñar un procesador de propósito especial con predicción dinámica de saltos por medio de una tabla de historia de saltos. La mayoría de los programas que el procesador ejecutará están compuestos por bucles anidados a tres niveles:

```
FOR i = 1 ... n
     FOR j = 1 ... n
           FOR k = 1 ... n
                 ....
                 ....
           END
     END
END
```

¿Cuantos fallos se producirán con las dos alternativas estudiadas en clase, usando 1 bit (2 estados) y 2 bits (4 estados), si en ambos casos se parte de no saltar (en el caso de 2 bits se parte del "no saltar fuerte")?

9. Supongamos un procesador que implementa planificación dinámica de instrucciones utilizando el método del Scoreboard. Existen dos UFs no segmentadas de suma en punto flotante (latencia 3) y otras dos de multiplicación (latencia 6) y existe además una UF de carga/alamacenamiento (latencia 1). Se pide mostrar la evolución de la tabla de estado de las instrucciones para el código que sigue, indicando en qué ciclo finalizan cada etapa las distintas instrucciones.

```
MUL   F2, F5, F4
ADD   F6, F2, F4
ADD   F4, F5, F7
MUL   F3, F4, F6
LD    F4, 0(R15)
MUL   F6, F4, F2
```

10. Un procesador tiene una unidad de punto flotante que emplea 1 ciclo en la emisión de instrucciones, 1 ciclo en la lectura/escritura de registros, 2 ciclos para suma, 10 para multiplicación, y 30 para la división. Dispone de 2 unidades de multiplicación/división, y 3 unidades de suma/resta. Mostrar el diagrama instrucciones-tiempo para el código que se muestra a continuación, usando:

a) Scoreboard
b) Tomasulo

Considerar igual número de estaciones de reserva (Tomasulo) que de unidades funcionales (Scoreboard). Usar números para indicar en las tablas, el ciclo en el que se realizan las distintas fases de ejecución de cada instrucción (existen dos buses para la lectura del banco de registros, y uno para la escritura).

```
DIV F10, F11, F5
ADD F6, F10,F1
MUL F7, F6, F4
ADD F1, F13, F14
ADD F4, F1, F17
ADD F15, F16, F17
```

11. Cierto programa consume la mayor parte de su tiempo de ejecución en el siguiente fragmento de código:

T = a +Y +Z, donde T, Y, Z son vectores de n componentes y a un escalar.

Dicho programa se pretende ejecutar sobre un computador DLX cuyas características son:

(a) Frecuencia de reloj: 100 MHz.
(b) Gestión dinámica de instrucciones basada en el algoritmo de Tomasulo (tiempo de transferencia por el bus común de datos: 1 ciclo)
(c) Unidades de coma flotante **no segmentadas**: 1 de carga (4 ciclos, 2 buffer), 1 operador multifunción (3 ciclos, 3 estaciones de reserva)y 1 de almacenamiento (4 ciclos, 2 buffers).
(d) CPI para las instrucciones enteras = 1, y *delay-slot* =1.

El código generado por el compilador disponible es el siguiente:

```
; F0 contiene a
loop: LD    F4,Y(R1)
      LD    F2,Z(R1)
      ADDD  F4,F0,F4
      ADDD  F4,F2,F4
      SD    T(R1),F4
      BNEZ  R1,loop
      SUB   R1,R1,#8
```

Calcula la expresión del tiempo de ejecucióon del bucle escalar en función del tamaño de los vectores n.