PRÁCTICA 6ª: ALGORITMO DE TOMASULO: EJECUCIÓN DE PROGRAMAS

Arquitectura e Ingeniería de Computadores (3º curso)
E.T.S. de Ingeniería Informática (ETSINF)
Dpto. de Informática de Sistemas y Computadores (DISCA)

Objetivos:

- Analizar la influencia de los parámetros de diseño del procesador superescalar en las prestaciones obtenidas por los programas.

Desarrollo:

Para el desarrollo de la práctica se partirá de un simulador superescalar del MIPS (MIPS/OOO), que incorpora ejecución fuera de orden y especulación basada en el algoritmo de Tomasulo. El simulador acepta un conjunto de instrucciones enteras reducido, e instrucciones de coma flotante aritméticas y de carga/almacenamiento de doble precisión.

Para invocar la ejecución del simulador se utilizará la sintaxis siguiente:

```
./mips-ooo [OPCIONES] -f <fichero.s>

----- Ejecución -----

-j
          Genera un sólo archivo .htm con todos los resultados.
-s
          Ejecuta sin imprimir los ficheros de estado y los cronogramas.

----- Configuración -----

 NOTA: Un valor 0 en cualquier parámetro indica que se mantiene el valor por
 omisión.

-e <num>:<lat>:<segm>:<er>
          Número y latencia de los operadores de enteros/saltos,
          tipo ([s]egmentado/[c]onvencional), y estaciones de reserva
-l <num>:<lat>:<segm>:<tl>:<te>
          Número y latencia de los operadores de memoria,
          tipo ([s]egmentado/[c]onvencional), y tampones de lectura y escritura
-a <num>:<lat>:<segm>:<er>
          Número y latencia de los operadores de suma/resta/comparación,
          tipo ([s]egmentado/[c]onvencional), y estaciones de reserva
-m <num>:<lat>:<segm>:<er>
          Número y latencia de los operadores de multiplicación/división,
```

```
          tipo ([s]egmentado/[c]onvencional), y estaciones de reserva
-r <num>
          Número de entradas del Reorder Buffer
-v <issue>:<buses>:<commits>
          Número de vías del procesador superescalar en ISSUE, BUSES y COMMIT
-M
          Número de operadores y estaciones de reserva en función
          del número de vías
-p {1|2h|2s|p|c}
          Tipo de predictor.
            [1] : 1 bit (opción por omisión)
            [2h]: 2 bits con histéresis
            [2s]: 2 bits con saturación
            [p] : Perfecto
            [c] : BTB con predictor perfecto
            [pnt] : Predict-not-taken
-b <num>
          Número de entradas del BTB

----- Entrada -----

-f <fichero.s>
          Nombre del fichero en ensamblador
```

Instrucciones implementadas

Enteras	Coma flotante
LD Rx, desp(Ry)	L.D Fx, desp(Ry)
SD Ry, desp(Rx)	S.D Fy, desp(Rx)
DADD Rx, Ry, Rz	ADD.D Fx, Fy, Fz
DSUB Rx, Ry, Rz	SUB.D Fx, Fy, Fz
DADDI Rx, Ry, valor	
DSUBI Rx, Ry, valor	
	MUL.D Fx, Fy, Fz
	DIV.D Fx, Fy, Fz
	C.GT.D Fx, Fy
	C.LT.D Fx, Fy
BEQZ Rx, desp	BC1F desp
BNEZ Rx, desp	BC1T desp
TRAP #N	

Ejercicios a realizar

1. Influencia del número de vías en los resultados de la ejecución:

   Utilizando el programa almacenado en el archivo "daxpy64.s", el cual realiza la operación ($\vec{Z} = a \cdot \vec{X} + \vec{Y}$, bucle DAXPY) con vectores de 64 componentes, lanzaremos el simulador modificando el número de vías del procesador. Para que el resto

de parámetros de diseño no afecte a los resultados obtenidos, los configuraremos para disponer suficientes recursos, utilizando la opción "-M" de la línea de órdenes (se dimensiona suficientemente el número de operadores y estaciones de reserva en función del número de vías del procesador superescalar) y un tamaño de 128 entradas en el *reorder buffer*.

a) Procesador segmentado. Analizaremos las prestaciones obtenidas bajo tres estrategias para las dependencias de control: *predict-not-taken*, predictor BTB de 1 bit y predictor perfecto.

Lanzaremos las simulaciones con las órdenes:

```
./mips-ooo -f daxpy64.s -s -r 128 -M -p pnt

./mips-ooo -f daxpy64.s -s -r 128 -M -p 1

./mips-ooo -f daxpy64.s -s -r 128 -M -p p
```

b) Procesador superescalar de 2 vías. Analizaremos las prestaciones obtenidas bajo tres estrategias para las dependencias de control: *predict-not-taken*, predictor BTB de 1 bit y predictor perfecto.

Lanzaremos las simulaciones con las órdenes:

```
./mips-ooo -f daxpy64.s -s -r 128 -v 2:2:2 -M -p pnt

./mips-ooo -f daxpy64.s -s -r 128 -v 2:2:2 -M -p 1

./mips-ooo -f daxpy64.s -s -r 128 -v 2:2:2 -M -p p
```

c) Procesador superescalar de 4 vías. Analizaremos las prestaciones obtenidas bajo tres estrategias para las dependencias de control: *predict-not-taken*, predictor BTB de 1 bit y predictor perfecto.

Lanzaremos las simulaciones con las órdenes:

```
./mips-ooo -f daxpy64.s -s -r 128 -v 4:4:4 -M -p pnt

./mips-ooo -f daxpy64.s -s -r 128 -v 4:4:4 -M -p 1

./mips-ooo -f daxpy64.s -s -r 128 -v 4:4:4 -M -p p
```

d) Procesador superescalar de 8 vías. Analizaremos las prestaciones obtenidas bajo tres estrategias para las dependencias de control: *predict-not-taken*, predictor BTB de 1 bit y predictor perfecto.

Lanzaremos las simulaciones con las órdenes:

```
./mips-ooo -f daxpy64.s -s -r 128 -v 8:8:8 -M -p pnt

./mips-ooo -f daxpy64.s -s -r 128 -v 8:8:8 -M -p 1

./mips-ooo -f daxpy64.s -s -r 128 -v 8:8:8 -M -p p
```

Analizar el tiempo de ejecución y los CPI (ó IPC) obtenidos. Mostrar en una gráfica dichos resultados e interprétalos.

2. Influencia del tamaño del *reorder buffer* en los resultados de la ejecución.

   Utilizando el programa almacenado en el archivo "ordena-fp.s", el cual realiza la ordenación de un vector, lanzaremos el simulador modificando el tamaño del *reorder buffer*. Utilizaremos un procesador superescalar de 4 vías con un predictor perfecto para los saltos.

   De la misma forma que hemos hecho en el apartado anterior, configuraremos el resto de parámetros de diseño para disponer suficientes recursos, utilizando la opción "-M" de la línea de órdenes.

   Seguidamente, procederemos a lanzar las simulaciones para diferentes tamaños del *reorder buffer*:

```
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -M -p p -r 64
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -M -p p -r 32
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -M -p p -r 16
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -M -p p -r 8
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -M -p p -r 4
```

   Analizar el tiempo de ejecución y los CPI (ó IPC) obtenidos. Mostrar en una gráfica dichos resultados e interprétalos.

3. Dimensionamiento de los recursos de la unidad de ejecución.

   Utilizando el mismo programa a que en el apartado anterior ("ordena-fp.s"), supóngase que se ha decidido emplear la configuración con 64 entraadas en el *reorder buffer*. En los resultados del simulador ("Ocupación de recursos") se puede obtener los valores máximos para los diferentes parámetros del procesador, que servirían para dimensionar los recursos:

```
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -M -p p -r 64
```

   Puede comprobarse que una configuración con ese número de recursos sería suficiente para alcanzar las mismas prestaciones:

```
./mips-ooo -f ordena-fp.s -s -v 4:4:4 -e 4:1:c:10 -a 4:4:c:11
                          -l 3:2:c:6:12 -p p -r 64
```

4. Influencia del predictor en los resultados de la ejecución.

   Ya hemos observado en los experimentos realizados la gran importancia de la precisión en la predicción. En este apartado, utilizando el programa almacenado en el archivo "suma-mat.s", el cual realiza la suma de dos matrices, lanzaremos el simulador modificando el tipo de predictor. Utilizaremos un procesador segmentado con la configuración por defecto.

   Lanzaremos las simulaciones con las órdenes:

```
./mips-ooo -s -f suma-mat.s -p pnt
./mips-ooo -s -f suma-mat.s -p 1
./mips-ooo -s -f suma-mat.s -p 2h
./mips-ooo -s -f suma-mat.s -p 2s
./mips-ooo -s -f suma-mat.s -p p
```

Analizar el tiempo de ejecución y los CPI (ó IPC) obtenidos. Mostrar en una gráfica dichos resultados e interprétalos.