Arquitecturas Avanzadas

Solución al examen de Junio de 2001

Fernando Pardo, 6 de junio de 2001

Teoría

Las preguntas 1, 3 y 6 están contestadas en teoría y no tienen una dificultad especial.

Pregunta 2

El esquema de una memoria entrelazada de este estilo se ha visto ya en clase, por eso no lo repito aquí. Si se tienen 8 módulos en total necesitamos 3 bits para seleccionar cada uno de ellos. Si se quiere entrelazado y además tolerancia a fallos (tener al menos dos bancos) hay que utilizar selección de orden alto y bajo al mismo tiempo, es decir, de los tres bits de la dirección que podemos utilizar para seleccionar el módulo, podemos el más significativo para seleccionar el banco (dos bancos) y los dos menos significativos para seleccionar el módulo dentro del banco (4 módulos entrelazados). Otra posibilidad es coger los dos bits más significativos para los bancos (4 bancos) y el menos significativo para el entrelazado (2 módulos entrelazados).

Pregunta 4

La salida 001110 no es posible en un sistema secuencialmente consistente puesto en cualquier entrelazado de las instrucciones que se nos ocurra la última instrucción va a ser siempre un *print*, incluso suponiendo que una instrucción como *print(x,y)* se puede dividir en dos *prints* entrelazables. Como la última instrucción es un *print*, antes se han tenido que ejecutar necesariamente todas las asignaciones que ponían a 1 todas las variables.

La salida 000000 es perfectamente válida puesto que en la consistencia PRAM sólo se exige la secuencialidad de las instrucciones de un mismo procesador. Por lo tanto, todos pueden leer cero aunque las escrituras se hubieran lanzado con anterioridad (puede ocurrir que no se hayan propagado al resto de procesadores). Esto es así porque ningún procesador lee una variable que él mismo asigna sino que lee las de los demás.

Pregunta 5

La explicación del directorio encadenado está dada en teoría. La memoria adicional que utiliza es debida a los punteros que utiliza tanto en la memoria como en cada línea de caché en cada procesador. En nuestro caso particular los punteros son de 6 bits (64 procesadores). En la memoria hay 1024 líneas de caché por lo que en la parte de la memoria tendremos 1024*6 bits para los punteros. Pero además tenemos los punteros de cada línea de caché de cada procesador. Si suponemos que tienen L líneas y punteros (lista doblemente enlazada) tendremos 64*2*6*L bits que ocuparán espacio también en las cachés. Por lo tanto el coste adicional total será de 1024*6 + 64*2*6*L.

El directorio limitado con dos punteros ocupa 1024*2*6 con lo que dependiendo del valor de L (líneas de caché en cada caché) ocupará más o menos. De hecho, a poco que L valga 2, el directorio encadenado ya ocupa más espacio global.

Pregunta 7

Es como el algoritmo de primero-oeste pero en vez de ser primero-oeste es primero-este, es decir, si el destino está a la derecha debe ir a la derecha, en cualquier otro caso hay libertad:

```
Algoritmo: Mínimo Primero-Este para mallas 2-D
Inputs: (Xact, Yact) (Xdest,Ydest)   // nodo actual y destino
Output: Canal
Procedure:
     Xoffset=Xdest-Xact
     Yoffset=Ydest-Yact
     if Xoffset>0 then              // si destino derecha
          Canal=X+                  // única opción
     end if
     if Xoffset<0 and Yoffset>0 then
          Canal=selec(X-,Y+)
     end if
     if Xoffset<0 and Yoffset<0 then
          Canal=selec(X-,Y-)
     end if
     if Xoffset<0 and Yoffset=0 then
          Canal=selec(X-)
     end if
     if Xoffset=0 and Yoffset>0 then
          Canal=selec(Y+)
     end if
     if Xoffset=0 and Yoffset<0 then
          Canal=selec(Y-)
     end if
     if Xoffset=0 and Yoffset=0 then
          Canal=Interno
     end if
```

Problema

Primera parte

No hay encadenamiento ni demasiados recursos. La división en convoys y los tiempos de arranque correspondientes son lo que siguen:

12	LV	R1,V1
12	LV	R2,V2
12	ADDV LV	V2,V2,V1 R3,V3
12	SUBV LV	V2,V2,V3 R4,V4
12	ADDV SV	V4,V4,V2 R1,V2
7	MULTV	V3,V3,V4
12	SV ADDV	R3,V3 V4,V3,V1
12	SV	R2,V2
12	SV	R4,V4

Por lo tanto hay 9 convoys y 5 instrucciones en coma flotante. El tiempo de arranque será:

Tarranque=12+12+12+12+12+7+12+12+12=103

Tn=⌈n/64⌉(15+103)+9n

Tn(n>>)=10.84n+118

R∞=5*200/10.84=92.22 MFLOPS

Segunda parte

Ahora ya se permiten más cosas, pero hay restricciones, como el número de cauces de escritura a los registros, que son las que principalmente nos marcan el número de convoys:

12+6=18	LV	R1,V1	12
	LV	R2,V2	12
	ADDV	V2,V2,V1	6
	LV	R3,V3	12
12+6+7+12=37	SUBV	V2,V2,V3	6
	LV	R4,V4	12
	ADDV	V4,V4,V2	6
	SV	R1,V2	12
	MULTV	V3,V3,V4	7
	SV	R3,V3	12
6+12=18	ADDV	V4,V3,V1	6
	SV	R2,V2	12
	SV	R4,V4	12

El número de convoy es de 3 y el Tarranque=18+37+18=73, con lo que el tiempo de ejecución por elemento para n grande queda:

Tn(n>>)=4.37n+88

Y el Rendimiento cuando n tiende a infinito será:

R∞=5*200/4.37=228.57 MFLOPS

Para calcular el $N_{1/2}$ suponemos en primer lugar que $N_{1/2}$<=64, por lo que el tiempo por elemento queda como Tn(n<=64)=3n+88. El rendimiento para $N_{1/2}$ es la mitad de R∞, por lo tanto:

$R(N_{1/2})=5*200*N_{1/2}/(3*N_{1/2}+88)=228.57/2$

Despejando tenemos:

$N_{1/2} = 88*114.28/(1000-3*114.28)=15.3$

Que podemos redondear a 15 que es el entero más cercano

Tercera parte

En ambos casos se puede mejorar el código para tener menos convoys y por tanto mejorar el rendimiento.

En el **primer caso** basta con adelantar la penúltima instrucción (SV R2,V2) y ponerla justo después de la instrucción de multiplicación. Con esto convertimos estos dos convoys en uno solo:

12	LV	R1,V1
12	LV	R2,V2
12	ADDV LV	V2,V2,V1 R3,V3
12	SUBV LV	V2,V2,V3 R4,V4
12	ADDV SV	V4,V4,V2 R1,V2
12	MULTV SV	V3,V3,V4 R2,V2
12	SV ADDV	R3,V3 V4,V3,V1
12	SV	R4,V4

Se ha reducido un convoy y también se ha reducido en 7 el tiempo de arranque.

En cuanto al **segundo caso**, el número de combinaciones que reducían los convoys a sólo 2 es casi infinito. En el siguiente ejemplo simplemente se han intercambiado las instrucciones SUBV V2,V2,V3 y LV R4,V4 con lo que el código queda:

12+6=18	LV	R1,V1	12
	LV	R2,V2	12
	ADDV	V2,V2,V1	6
	LV	R3,V3	12
	LV	R4,V4	12
6+6+7+6+12=37	SUBV	V2,V2,V3	6
	ADDV	V4,V4,V2	6
	SV	R1,V2	12
	MULTV	V3,V3,V4	7
	SV	R3,V3	12
	ADDV	V4,V3,V1	6
	SV	R2,V2	12
	SV	R4,V4	12

No sólo el número de convoys es menor sino que también el tiempo de arranque se reduce sensiblemente.

Había otras opciones, como poner todas las cargas al principio y los almacenamientos al final, que también reducían el tiempo total de ejecución.