2º curso / 2º cuatr.

Grado en Ing. Informática

Arquitectura de Computadores. Ejercicios

Tema 4. Arquitecturas con Paralelismo a nivel de Instrucción (ILP)

Material elaborado por los profesores responsables de la asignatura:
Julio Ortega, Mancia Anguita

1 Ejercicios

Ejercicio 1. Para el fragmento de código siguiente:

```
1.  lw  r1,0x1ac      ; r1 ← M(0x1ac)
2.  lw  r2,0xc1f      ; r2 ← M(0xc1f)
3.  add r3,r0,r0      ; r3 ← r0+r0
4.  mul r4,r2,r1      ; r4 ← r2*r1
5.  add  r3,r3,r4     ; r3 ← r3+r4
6.  add r5,r0,0x1ac   ; r5 ← r0+0x1ac
7.  add r6,r0,0xc1f   ; r6 ← r0+0xc1f
8.  sub r5,r5,#4      ; r5 ← r5 - 4
9.  sub r6,r6,#4      ; r6 ← r6 - 4
10.  sw (r5),r3       ; M(r5) ← r3
11.  sw (r6),r4       ; M(r6) ← r4
```

y suponiendo que se pueden captar, decodificar, y emitir cuatro instrucciones por ciclo, indique el orden en que se emitirán las instrucciones para cada uno de los siguientes casos:

a) Una ventana de instrucciones centralizada con emisión ordenada

b) Una ventana de instrucciones centralizada con emisión desordenada

c) Una estación de reserva de tres líneas para cada unidad funcional, con envío ordenado.

Nota: considere que hay una unidad funcional para la carga (2 ciclos), otra para el almacenamiento (1 ciclo), tres para la suma/resta (1 ciclo), y una para la multiplicación (4 ciclos). También puede considerar que, en la práctica, no hay límite para el número de instrucciones que pueden almacenarse en la ventana de instrucciones o en el buffer de instrucciones.

Ejercicio 2. Considere que el fragmento de código siguiente:

```
1.lw r3,0x10a      ; r3 ← M(0x10a)
2.addi r2,r0,#128  ; r2 ← r0+128
3.add r1,r0,0x0a   ; r1 ← r0+0x0a
4.lw r4,0(r1)      ; r4 ← M(r1)
5.lw r5,-8(r1)     ; r5 ← M(r1-8)
6.mult r6,r5,r3    ; r6 ← r5*r3
7.add r5,r6,r3     ; r5 ← r6+r3
8.add r6,r4,r3     ; r6 ← r4+r3
9.sw 0(r1),r6      ; M(r1) ← r5
10.sw -8(r1),r5    ; M(r1-8) ← r5
11.sub r2,r2,#16   ; r2 ← r2-16
```

se ejecuta en un procesador superescalar que es capaz de captar 4 instrucciones/ciclo, de decodificar 2 instrucciones/ciclo; de emitir utilizando una ventana de instrucciones centralizada 2 instrucciones/ciclo; de

escribir hasta 2 resultados/ciclo en los registros correspondientes (registros de reorden, o registros de la arquitectura según el caso), y completar (o retirar) hasta 3 instrucciones/ciclo.

Indique el número de ciclos que tardaría en ejecutarse el programa suponiendo finalización ordenada y:

a) Emisión ordenada

b) Emisión desordenada

Nota: Considere que tiene una unidad funcional de carga (2 ciclos), una de almacenamiento (1 ciclo), tres unidades de suma/resta (1 ciclo), y una de multiplicación (6 ciclos), y que no hay limitaciones para el número de líneas de la cola de instrucciones, ventana de instrucciones, buffer de reorden, puertos de lectura/escritura etc.

Ejercicio 3. En el problema anterior, (a) indique qué mejoras realizaría en el procesador para reducir el tiempo de ejecución en la mejor de las opciones sin cambiar el diseño de las unidades funcionales (multiplicador, sumador, etc.) y sin cambiar el tipo de memorias ni la interfaz entre procesador y memoria (no varía el número de instrucciones captadas por ciclo). (b) ¿Qué pasaría si además se reduce el tiempo de multiplicación a la mitad?.

Ejercicio 4. En el caso descrito en el problema 3, indique cómo evolucionaría el buffer de reorden utilizado para implementar finalización ordenada, en la mejor de las opciones.

Ejercicio 5. En un procesador superescalar con renombramiento de registros se utiliza un buffer de renombramiento para implementar el mismo. Indique como evolucionarían los registros de renombramiento al realizar el renombramiento para las instrucciones:

```
1.  mul  r2,r0,r1 ; r2 ← r0*r1
2.  add  r3,r1,r2 ; r3 ← r1+r2
3.  sub  r2,r0,r1 ; r2 ← r0-r1
4.  add  r3,r3,r2 ; r3 ← r3+r2
```

Ejercicio 6. Considere el bucle:

```
i=1;
do
{
  b[i]=a[i]*c;
  c=c+1;
  if (c>10) then goto etiqueta;
  i=i+1;
} while (i<=10);
etiqueta:.....
```

Indique cuál es la penalización efectiva debida a los saltos, en función del valor inicial de c (número entero), considerando que el procesador utiliza:

a) Predicción fija (siempre se considera que se va a producir el salto)

b) Predicción estática (si el desplazamiento es negativo se toma y si es positivo no)

c) Predicción dinámica con un bit (1= Saltar; 0 = No Saltar; Inicialmente está a 1)

Nota: La penalización por saltos incorrectamente predichos es de 4 ciclos y para los saltos correctamente predichos es 0 ciclos.

Ejercicio 7. En la situación descrita en el problema anterior. ¿Cuál de los tres esquemas es más eficaz por término medio si hay un 25% de probabilidades de que c sea menor o igual a 0, un 30% de que sea mayor o igual a 10; y un 45% de que sea cualquier número entre 1 y 9, siendo todos equiprobables?

Ejercicio 8. En un programa, una instrucción de salto condicional (a una dirección de salto anterior) dada tiene el siguiente comportamiento en una ejecución de dicho programa:

SSNNNSSNSNSNSSSSSN

donde S indica que se produce el salto y N que no. Indique la penalización efectiva que se introduce si se utiliza:

a) Predicción fija (siempre se considera que se no se va a producir el salto)

b) Predicción estática (si el desplazamiento es negativo se toma y si es positivo no)

c) Predicción dinámica con dos bits, inicialmente en el estado (11).

d) Predicción dinámica con tres bits, inicialmente en el estado (111).

Nota: La penalización por saltos incorrectamente predichos es de 5 ciclos y para los saltos correctamente predichos es 0 ciclos.

Ejercicio 9. Indique cómo transformaría la secuencia de instrucciones que se muestran en la Tabla para un core de procesamiento VLIW de forma que sólo se utilicen instrucciones de movimiento de datos condicionales (no debe haber ninguna instrucción de salto condicional).

#	OP1	OP2
1	*lw r1, x(r2)*	*add r10, r11, r12*
2		*add r13, r10, r14*
3	*beqzr3, direc*	
4	*lw r4, 0(r3)*	
5	*lw r5, 0(r4)*	

Ejercicio 10. En un procesador todas las instrucciones pueden predicarse. Para establecer los valores de los predicados se utilizan intrucciones de comparación (cmp) con el formato (p) p1, p2 cmp.cnd x,y donde cnd es la condición que se comprueba entre x e y (lt, ge, eq, ne). Si la condición es verdadera p1=1 y p2=0, y si es falsa, p1=0 y p2=1. La instrucción sólo se ejecuta si el predicado p=1 (habrá sido establecido por otra instrucción de comparación).

En estas condiciones, utilice instrucciones con predicado para escribir sin ninguna instrucción de salto el siguiente código:

```
if (A>B) then { X=1;}
else {if (C<D) then X=2; else X=3;}
```

2 Cuestiones

Cuestión 1. ¿Qué tienen en común un procesador superescalar y uno VLIW? ¿En qué se diferencian?

Cuestión 2. ¿Qué es un buffer de renombramiento? ¿Qué es un buffer de reordenamiento? ¿Existe alguna relación entre ambos?

Cuestión 3. ¿Qué es una ventana de instrucciones? ¿Y una estación de reserva? ¿Existe alguna relación entre ellas?

Cuestión 4. ¿Qué utilidad tiene la predicación de instrucciones? ¿Es exclusiva de los procesadores VLIW?

Cuestión 5. ¿En qué momento se lleva a cabo la predicción estática de saltos condiciones? ¿Se puede aprovechar la predicción estática en un procesador con predicción dinámica de saltos?

Cuestión 6. ¿Qué procesadores dependen más de la capacidad del compilador, un procesador superescalar o uno VLIW?

Cuestión 7. ¿Qué procesadores tienen microarquitecturas con mayor complejidad hardware, los superescalares o los VLIW? Indique algún recurso que esté presente en un procesador superescalar y no sea necesario en uno VLIW.

Cuestión 8. Haga una lista con 5 microprocesadores superescalares o VLIW que se hayan comercializado en los últimos 5 años indicando alguna de sus características (frecuencias, núcleos, tamaño de cache, etc.).