pr0gramming_312823
www.wuolah.com/student/pr0gramming_312823

preguntaspruebastema-42020.pdf

TODAS Preguntas Test Tema 4 2020

2° Arquitectura de Computadores

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Preguntas Pruebas del Tema 4

El aprovechamiento eficiente del paralelismo entre instrucciones (ILP) en un procesador VLIW necesita estructuras hardware como el ROB
Precisamente se intenta reducir la complejidad hardware del procesador eliminando estructuras como el ROB y dejando que sea el compilador el principal responsable de la planificación de las instrucciones
(F)

El aprovechamiento eficiente del paralelismo entre instrucciones (ILP) en un procesador VLIW NO requiere el uso del ROB
Precisamente se intenta reducir la complejidad hardware del procesador eliminando estructuras como el ROB y dejando que sea el compilador el principal responsable de la planificación de las instrucciones
(V)

El buffer de reordenamiento (ROB) permite implementar la finalización ordenada en un procesador superescalar
Efectivamente, el ROB facilita implementar de la finalización ordenada en los superescalares.
(V)

El desenrollado de bucles necesita un hardware especial de apoyo
Se consigue escribiendo adecuadamente el bucle (menos iteraciones y cuerpo del bucle con más instrucciones)
(F)

El principal responsable del aprovechamiento eficiente del paralelismo entre instrucciones (ILP) en un procesador superescalar es el compilador.
Precisamente es el aumento de la complejidad del hardware del superescalar el que ha permitido aumentar la eficiencia con la que se aprovecha el paralelismo ILP en estos procesadores.
(F)

El procesador PROCEXA de 32 bits, puede captar TRES instrucciones por ciclo, DECODIFICAR TRES instrucciones por ciclo, y dispone de una ventana de instrucciones centralizada desde donde se produce la EMISIÓN DESORDENADA de hasta DOS instrucciones por ciclo. También dispone de un buffer de reordenamiento (ROB) donde implementa el renombramiento y la finalización ordenada, pudiendo retirarse desde el ROB, TRES instrucciones por ciclo para escribir sus resultados en la etapa WB del cauce. El procesador incluye además DOS unidades de Suma/Resta de UN ciclo; UN multiplicador de TRES ciclos, y UNA unidad de Carga/Almacenamiento de memoria de DOS ciclos.
Por tanto, las etapas del cauce del procesador son: IF (captación de instrucciones), ID (Decodificación de instrucciones), EX (ejecución en unidad funcional de la operación codificada por la instrucción), y WB (retirada de la instrucción del ROB y escritura de resultado en los registros del procesador).
No se considera etapa ROB explícita en el cauce porque se supone que en el último ciclo de ejecución de las unidades funcionales, el resultado de la operación queda almacenado en el ROB

Para este procesador y la secuencia de instrucciones siguiente

```
(1)   ld r1,0(r3)      // r1=M(r3)
(2)   ld r2,8(r3)      // r2 = M(r3+8)
(3)   add r2, r1, r2   // r2=r1+r2
(4)   mul r4, r1, r2   // r4= r1*r2
(5)   sub r4, r1, r4   // r4=r1-r4
(6)   sub r5, r1, r3   // r5=r1-r3
```

¿En qué ciclo se empieza a ejecutar la instrucción (6)? (los ciclos se numeran desde 1)

Ciclo 5. La respuesta se obtiene del diagrama temporal de procesamiento de las instrucciones siguiente

		1	2	3	4	5	6	7	8	9	10	11	12
(1)	ld r1,0(r3) ; r1=M(r3)	IF	ID	EX	EX	WB							
(2)	ld r2,8(r3) ; r2=M(r3+8)	IF	ID			EX	EX	WB					
(3)	add r2,r1,r2 ; r2=r1+r2	IF	ID					EX	WB				
(4)	mul r4,r1,r2 ; r4=r1*r2		IF	ID					EX	EX	EX	WB	
(5)	sub r4,r1,r4 ; r4=r1-r4		IF	ID								EX	WB
(6)	sub r5,r1,r3 ; r5=r1-r3		IF	ID		EX							WB

Para el procesador PROCEXA y la secuencia de instrucciones que se proporciona:

¿En qué ciclo empieza a ejecutarse la instrucción (2)? (los ciclos se numeran desde 1)

Ciclo 5. La respuesta se obtiene de la tabla de procesamiento de la secuencia de instrucciones

(5)

Para el procesador PROCEXA y la secuencia de instrucciones que se proporciona:

¿En qué ciclo empieza a ejecutarse la instrucción (4)? (los ciclos se numeran desde 1)

Ciclo 8. La respuesta se obtiene a partir de la tabla de procesamiento de la secuencia de instrucciones

(8)

Para el procesador PROCEXA y la secuencia de seis instrucciones que se proporciona:

¿Cuántos ciclos tarda en procesarse la secuencia de seis instrucciones indicada?

12 ciclos. La respuesta se obtiene a partir de la tabla de procesamiento de la secuencia de instrucciones

(12)

En la predicción dinámica de dos bits, el sentido de la predicción cambia (de saltar a no saltar o de no saltar a saltar) solo cuando el predictor falla dos veces seguidas.
No siempre. Depende del estado en que esté, el cambio de sentido puede necesitar dos fallos o puede producirse con un único fallo.
(F)

En la predicción dinámica de dos bits, el sentido de la predicción puede no cambiar (de saltar a no saltar o de no saltar a saltar) cuando el predictor falla.
Esto ocurriría si se encontrase en el estado 11 ó en el 00 Depende del estado en que esté, el cambio de sentido puede necesitar dos fallos o puede producirse con un único fallo.
(V)

En la predicción dinámica de dos bits, el sentido de la predicción siempre cambia (de saltar a no saltar o de no saltar a saltar) cuando el predictor falla
No siempre cambia el sentido de la predicción. Depende del estado en que esté puede necesitar hasta dos fallos para cambiar el sentido de la predicción
(F)

En la predicción dinámica de un bit el sentido de la predicción no cambia (de saltar a no saltar o de no saltar a saltar) mientras el predictor no falle
Efectivamente, se comporta de esa manera
(V)

En la predicción dinámica de un bit el sentido de la predicción cambia (de saltar a no saltar o de no saltar a saltar) si el predictor falla
Efectivamente, se comporta de esta manera
(V)

La segmentación software es una técnica puramente software
(V)

La segmentación software no se puede aplicar en el caso de los procesadores superescalares
En una técnica software que puede utilizarse para cualquier procesador, independientemente de su microarquitectura, aunque tiene más sentido utilizarla para ciertos procesadores
(F) Suponga que en un mismo ciclo se decodifican las cuatro instrucciones siguientes

```
(i)     add r2, r2, r1          // r2 = r2+r1
(i+1)   mul r3, r4, r1          // r3 = r4*r1
(i+2)   add r4, r4, r1          // r4 = r4+r1
(i+3)   sub r5, r2, r1          // r5 = r2- r1
```

(entre paréntesis se indica el orden en el que están en el código), y pasan a una ventana de instrucciones única desde la que se emiten a las distintas unidades funcionales. El procesador puede emitir (con emisión DESORDENADA) CUATRO instrucciones por ciclo y tiene DOS unidades de suma/resta (con un retardo de 1 ciclo) y UN multiplicador (con un retardo de 3

ciclos).

Las instrucciones (i) e (i+3) se pueden emitir en el mismo ciclo

Hay dependencia RAW entre ellas
(F)

En la misma situación de la pregunta anterior para la misma secuencia de instrucciones.
Las cuatro instrucciones se pueden emitir en un único ciclo
Hay dependencia RAW entre la primera y la última.
(F)

Un procesador de 32 bits (4 bytes) NO podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código) aunque implemente adelantamiento de LOADs a STORES ESPECULATIVO
(i) sw 0(r6), r2 // M(r6)<-- r2
(i+1) lw r4, 8(r6) // r4 <--M(r6+8)
El dato que se escribe y el que se lee no se solapan y no hay riesgo RAW. No hace falta que el procesador tenga adelantamiento especulativo.
(F)

Un procesador de 32 bits (4 bytes) que implementa adelantamiento de loads a stores ESPECULATIVO podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código)
(i) sw 0(r5), r2 // M(r5)<-- r2
(i+1) lw r4, 8(r5) // r4 <--M(r5+8)
El dato que se escribe y el que se lee no se solapan y no hay riesgo RAW. No hace falta que el procesador sea tenga adelantamiento especulativo
(V)

Un procesador de 32 bits (4 bytes) que NO implementa adelantamiento de loads a stores ESPECULATIVO podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código)
(i) sw 0(r6), r2 // M(r6)<-- r2
(i+1) lw r4, 8(r6) // r4 <--M(r6+8)
El dato que se escribe y el que se lee no se solapan y no hay riesgo RAW. No hace falta que el procesador tenga adelantamiento especulativo
(V)

Un procesador podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código) si implementa adelantamiento de LOADs a STORES ESPECULATIVO

(i) sw 0(r5), r2 // M(r5)<-- r2
(i+1) lw r4, 0(r6) // r4 <--M(r6)
No se puede saber si r5 y r6 apuntan a direcciones solapadas hasta que no se obtengan los correspondientes valores al ejecutar el código. Por eso para que se pueda producir el adelantamiento, el procesador debe implementar adelantamiento especulativo.
(V)