Manifestando el aprobado: III
Examen Arq

Brais Míguez - Lucía Picos

May 2022

1. **El resultado de ejecutar el benchmark de conjuntos SPEC. . . , consiste en un total de 2,389 × 10^{12} instrucciones. Se ejecuta en 750s y el tiempo de referencia es de 9650s. Calcula:**

   a) CPI si el ciclo de reloj es 0,333 ns

   $TiempoCPU = NumInstrucciones * CPI * Período$

   $CPI = \frac{750}{2.389*10^{12}*0.333*10^{-9}} = 0.942$ ciclos

   b) Calcula el SPECRatio

   El SPECRatio no es más que el Speed UP respecto al programa de referencia:

   $Speed\ UP = TiempoReferencia\ /\ NuevoTiempo = \frac{9650}{750} = 12.86$

   c) Calcula el incremento en tiempo de CPU si se aumenta en un 10 % de las instrucciones del benchmark manteniendo el CPI calculado

   $TiempoCPU = (NumInstrucciones * 1.1) * CPI * Período$

   $TiempoCPU2 = (2.389 * 10^{12} * 1.1) * 0.942 * 0.333 * 10^{-9} = 824.33$ s

   $Incremento = Tiempo\ CPU2 - TiempoCPUOriginal =$ 824.33 - 750 = 74.33 s

2.

   i0: lw $f0, 0($r1)

   i1: sub.f $f4, $f0, $f2

   i2: add.d $f6, $f6, $f4

   i3: addi $r1, $r1, 4

   i4: sub $r3, $r3, 1

   i5: bnez $r3, bucle

   Supongamos unidad del MIPS con 5 etapas, hay adelantamiento. Se decide dirección de salto en ID y si se salta en EX. No hay especulación de salto

   a) Identificar cuántos ciclos se pierden cuando hay predicción de salto tomado y al final no se toma y cuántos ciclos se pierden si se predice que no se toma y al final se toma

   2 ciclos en ambas

   En ambas se detecta en la etapa Ex que la predicción fue incorrecta. Se borran las operaciones que se habían predicho y se carga la instrucción correcta debajo de la etapa M del salto

   ——--------

b) Salta con patrón T, T, NT, T. Predictor de 1 bit que empieza con salto no tomado. Calcular % de acierto y cuantos ciclos se pierden por predicciones erróneas

Real T T NT T

Predicción NT T T NT

2 ciclos por fallo de predicción POR QUE? Por la respuesta del apartado a

acierto ¼ = 25%

c) Lo mismo pero predictor fijo no tomado

2 ciclos por fallo de predicción POR QUE?
acierto ¼ = 25%

3. Teniendo esto en cuenta:

Operación que produce el resultado	Instrucción que los usa	Latencia
Operación ALU FP doble	Operación ALU FP doble	6
Operación ALU FP doble	Almacenar doble	3
Cargar FP doble	Operación ALU doble	2
Cargar FP Doble	Almacenar doble	0

Bucle:

i0: l.d $f0, 0($r1)

i1: l.d $f2, 0($r2)

i2: add.d $f4, $f0, $f2

i3: s.d $f4, 0($r3)
i4: daddui $r1, $r1, -8

i5: bne $r1, $r4, bucle

Array de tamaño 4000

a) Considerando latencias: cuántos ciclos tarda cada iteración y cuántos en total

i0: l.d $f0, 0($r1)

i1: l.d $f2, 0($r2)

x2 <stall> (por dependencia entre “cargar fp double” y “operación alu double”)

i2: add.d $f4, $f0, $f2

x3 <stall> (por dependencia entre “operación alu fp double” y “almacenar double”)

i3: s.d $f4, 0($r3)
i4: daddui $r1, $r1, -8

i5: bne $r1, $r4, bucle

total 6+ 2+3 = 11 ciclos por iteración

A) Si se refiere a array de 4000 bytes:

Array = 4000 bytes, en cada iteración se leen 8 bytes(double)
Iteraciones = 4000/8 = 500 elementos

Total 11 * 500 = 5500 ciclos.

B) Si se refiere a array de 4000 elementos:

4000 * 11 = 44000 ciclos

b) ¿Cómo quedaría el bucle si se planifica? Determinar ciclos para todas la iteraciones

i0: l.d $f0, 0($r1)

i1: l.d $f2, 0($r2)

i4: daddui $r1, $r1, -8
1 ciclo
i2: add.d $f4, $f0, $f2

i3: s.d $f4, 0($r3)
3 ciclos
i5: bne $r1, $r4, bucle

Total = 6+ 4 = 10

En este tendría que ir la i5: bne necesariamente al final? O se podría poner antes de la i3: s.d?
bne es un salto condicional si lo pones antes que la i3 siempre que salte ni se termina de ejecutar sd

c) Desenrollado para dos iteraciones. Total de ciclos
i0: l.d $f0, 0($r1)
i1: l.d $f2, 0($r2)
i2: l.d $f3, -8($r1)
i3: l.d $f5, 0($r2)
i4: add.d $f4, $f0, $f2 //Este no sobraría pq siempre se sobreescribe?
i5: add.d $f4, $f3, $f5
i6: s.d $f4, 0($r3)
i7: daddui $r1, $r1, -16
i8: bne $r1, $r4, bucle

SE PARTE DE ESTE:

i0: l.d $f0, 0($r1)
i1: l.d $f2, 0($r2)
i2: add.d $f4, $f0, $f2
i3: s.d $f4, 0($r3)
i4: daddui $r1, $r1, -8
i5: bne $r1, $r4, bucle

Yo creo que sería así (**sin planificar** pero desenrollando):
i0: l.d $f0, 0($r1)
i1: l.d $f2, 0($r2)
i2: l.d $f1, -8($r1)
i3: l.d $f3, -8($r2)
Parada de 1 ciclo
i4: add.d $f4, $f0, $f2
i5: add.d $f5, $f1, $f3
Parada de 2 ciclos
i6: s.d $f4, 0($r3)
i7: s.d $f5, -8($r3)

i8: daddui $r1, $r1, -16
Parada de 6 ciclos
i9: bne $r1, $r4, bucle

Total de ciclos iteración = 19. Suponiendo que en cada iteración se leen 16 bytes (2 doubles) 250 iteraciones. Total de ciclos = 19 * 250 = 4750

Y si hubiese que desenrollar el **planificado** sería así:
i0: l.d $f0, 0($r1)
i1: l.d $f2, 0($r2)
i2: l.d $f1, -8($r1)
i3: l.d $f3, -8($r2)
i8: daddui $r1, $r1, -16
i4: add.d $f4, $f0, $f2
i5: add.d $f5, $f1, $f3
Parada de 2 ciclos
i6: s.d $f4, 0($r3)
i7: s.d $f5, -8($r3)
i9: bne $r1, $r4, bucle

Total de ciclos iteración = 12.
Suponiendo que en cada iteración se leen 16 bytes (2 doubles) 250 iteraciones.
Total de ciclos = 12 * 250 = 3000

4. Partiendo de:

Procesador	A	B	C	D
Caché P0	Compartido	Modificado	Compartido	Compartido
Caché P1	Inválido	Inválido	Inválido	Compartido
Caché P2	Inválido	Inválido	Compartido	Compartido

Se tiene que llegar a:

Procesador	A	B	C	D
Caché P0	Inválido	Inválido	Inválido	Compartido
Caché P1	Inválido	Inválido	Compartido	Modificado
Caché P2	Modificado	Modificado	Inválido	Compartido

Explicar secuencias de cambios para llegar al estado final y describir tráfico del bus (Acierto, Fallo).
Nota 1 : para alcanzar estado final puede hacer falta una sola instrucción
Nota 2 : pueden existir estados finales inalcanzables

1. P2 escribe A, para ello,se apropia del bus y manda un petición PtLecEsc. El P0 le proporciona el bloque solicitado(RPBloque) e

invalida esa línea. El valor del bloque en P2 pasa a modificado. También se lo puede pasar la memoria.

2. El procesador P2 pide solicita el dato B que tiene en estado inválido a través de una solicitud PtLecEsc, el P0 le manda el bloque e invalida esa línea.
3. */*Para que el dato C acabe en esos estados podemos suponer lo siguiente. El P0 quiere escribir ese dato, lo solicita a traves de una petición PtLecEsc, el P1 se lo manda e invalida su linea. El P1 ahora tiene esa linea como modificada y el resto como inválida. Ahora el P2 quiere leer ese dato, asi que manda una petición PtLec, el P0 le manda el bloque y establece el estado de la línea a compartido, la memoria también recibe una copia del dato. Ahora P0 y P2 tienen la línea en estado compartido.*
   P2 realiza una nueva lectura de un nuevo dato y se produce reemplazo del dato C, por lo que se invalida.
4. *Solo P1 y la memoria tienen el dato-> compartido.*/*
   Creo que hubo una confusión y se realizó este apartado del supuesto estado final al inicial en lugar de al revés (del inicial al final). Creo que es así (no lo tengo 100% claro tho):
   P1 realiza una lectura del dato C, se apropia del bus y envía una petición de lectura (PtLec). P0 o P2 le proporcionan el dato, por lo que ahora los tres procesadores tienen C en estado compartido.
   A continuación, P0 y P2 realizan lecturas hasta tal punto que la línea que contiene a C se reemplaza (se invalida en sus cachés).

   Lo anterior está mal, es inviable igual que el apartado 4.
5. Esto no es posible, si el dato D esta modificado en la P1, el resto de caches lo tendrán que invalidar.

Teoría

1) Estación de reserva. Qué es, para qué sirve y en qué tipo de procesadores se encuentra.

Las estaciones de reserva son componentes hardware que se usan en la planificación dinámica de instrucciones en procesadores superescalares(permiten la emisión de múltiples instrucciones por ciclos.)

Cuando una instrucción está en la cola de instrucciones, se emite a una estación de reserva junto a sus operandos.

Si los operando están disponibles la instrucción se ejecuta en las unidades funcionales. Si no están disponibles se detiene la instrucción. Las estaciones escuchan el bus común (CDB), a la espera de que los operandos estén disponibles. Cuando por fin les llega el operando, ejecutan la instrucción que estaba detenida.

2) ¿Qué significa estado exclusivo en MESI?

El protocolo MESI es un protocolo de coherencia caché, es una mejora del protocolo MSI que incluye un cuarto estado, el EXCLUSIVO. Un bloque está en estado exclusivo si las únicas copias válidas del dato están en una caché y en memoria, es decir, el resto de cachés no tienen el dato.
Este estado supone una disminución del tráfico del bus de datos. Esto es debido a que cuando se produce una escritura de un bloque en estado Exclusivo no es necesario mandar una señal de invalidación al resto de cachés(ya que no lo tienen).

CREO que si hay que mandar una señal a la memoria para que ponga el bit de validez a 0 de ese bloque, pero esto no es fijo.

3) Procesador superescalar. ¿Es posible planificación dinámica con especulación? ¿En caso afirmativo explicar?
Un procesador superescalar permite la emisión de múltiples instrucciones en un único ciclo. La emisión de instrucciones puede ser estática(la realiza el compilador, normalmente se basa en reordenar instrucciones para evitar dependencias y crear paquetes de emision) o dinámica, la realiza el hardware→ se puede reordenar instrucciones, ejecutar instrucciones fuera de orden(aunque la post-escritura debe ser en orden del programa) y en algunos casos renombramiento de registros(TOMASULO).

En la ejecución dinámica si que es posible la especulación. La especulación puede ser de salto o de carga. En la planificación dinámica para llevar a cabo la especulación se usan buffers(ROB) que almacenan el resultado de las instrucciones que se especulan. Cuando la predicción es acertada, se escriben los valores en el banco de registros o se pasan a las estaciones de reserva a las instrucciones que necesiten los operandos. Si la predicción falla se borran los resultados en el ROB.

Es importante mencionar que los resultados se retiran del ROB en el orden de ejecución del programa, es decir, que se encarga del reordenamiento.

4) ¿Qué mide MIPS y GFLOPS? Desventajas

MIPS y GFLOPS son métricas que rendimiento que permiten contabilizar el número de instrucciones que ejecuta un procesador por unidad de tiempo, aunque estas métricas hay que cogerlas con pinzas ya que no son una buena comparación entre procesadores con distintos conjuntos de instrucciones y pueden llegar a variar entre programas.

GLFLOPS → miles de millones de operaciones en punto flotante por segundo.

MIPS→ millones de instrucciones por segundo.

5) ¿Qué es el TDP?

El TDP o Thermal Design Power es la medida máxima de calor que puede generar un chip para que su sistema de disipado mantenga el sistema funcionando de forma eficiente. Suele ser 1.5 veces menor que la potencia pico y mayor que la potencia media.

Examen Arq

May 2021

1) Se dispone de un Pc con 1 núcleo que ejecuta 1 app que dedica el 75 % del tiempo a tareas de cálculo. El 25 % restante lo dedica a esperar operaciones de E/S. Del 75 %, el 90 % lo pasa ejecutando instrucciones en punto flotante y el restante en otras instrucciones. En punto flotante requiere 15 CPI. El resto de instrucciones 5 CPI.

   Pc2 tiene 4 núcleos y una frecuencia de reloj un 50 % más alta que la máquina original en el que las instrucciones de coma flotante requieren un 20 % menos de ciclos de reloj; y el resto de instrucciones los mismos ciclos de reloj. Consideremos además que este computado B no incorpora ninguna mejora para las operaciones de E/S.

   Supón que la parte de cálculo es **totalmente paralelizable** entre los 4 núcleos de B, mientras que la E/S no admite ningún tipo de paralelización.

   a) ¿Cuál es el tiempo de ejecución dedicado a cálculo en B? esto esta bien???

Tiempo de cálculo del PC mononúcleo: Instrucciones * Período * 0.9 * 15 + Instrucciones * Período * 0.1 * 5 = 14 * Instrucciones * Período

Tiempo de cálculo del PC multinúcleo: (Instrucciones * (Período/1.5) * 0.9 * 15 * 0.8 + Instrucciones * (Período/1.5) * 0.1 * 5 = 14 * Instrucciones * Período) / 4

=>

$$(Instrucciones * Periodo * 7.2 + Instrucciones * Periodo * 0.33) / 4$$

8 =>

$$(Instrucciones * Periodo * 7.53) / 4 = 1.88\ (Instrucciones * Periodo)$$

$$Speed\ UP = \frac{Instrucciones * Periodo * 14}{Instrucciones * Periodo * 1.88} = \frac{14}{1.88} = 7.44$$

   b) ¿Cuál es la aceleración/deceleración global de la aplicación en B respecto a A?

   Para calcular el efecto sobre el tiempo total hay que usar la ley de amdahl:

$$Ley\ de\ Amdahl = \frac{1}{(1-FM) + (FM/AM)};$$

$$En\ este\ caso\ -> \frac{1}{(1-0.75) + (0.75/7.44)} = 2.88\ veces\ más\ rápido\ el\ pc\ de\ 4\ núcleos\ que\ el\ de\ 1$$

2) Cuando un programa se compila para el computador A requiere 1000 millones de instrucciones y el CPI en dicho computador es 1. En el caso del computador B, el mismo programa una vez compilado requiere que se ejecuten 800 millones de instrucciones y el CPI será 1,1. Ambos tienen una frecuencia de reloj de 4GHz.

a) Valor del MIPS en cada computador. ¿En cuál de los computadores se alcanza un mejor valor de MIPS y por qué?

PC A => 1000 millones de instrucciones -> CPI = 1

PC B => 800 millones de instrucciones -> CPI = 1.1

Frecuencia = 4 GHZ; $T = 1/F = 1/(4*10^9) = 2.5*10^{-10}$

$MIPS = N_{Instrucciones} / (Tiempo_{CPU}*10^6)$

$Tiempo_{CPU} = N * CPI * T$

$Tiempo_{CPU}A = 10^9 * 1 * 2.5*10^{-10} = 0.25$ segundos

$Tiempo_{CPU}B = 8 * 10^8 * 1.1 * 2.5*10^{-10} = 0.22$ segundos

$MIPS_A = 10^9 / (0.25*10^6) = 4000$ millones de instrucciones por segundo

$MIPS_B = 8 * 10^8 / (0.22*10^6) = 3636$ millones de instrucciones por segundo

Aún teniendo que ejecutar un mayor número de instrucciones, el MIPS de A es mayor debido a que tiene un CPI más pequeño, lo cual permite que ejecute un mayor número de instrucciones por segundo.

b) Indicar el tiempo de cada uno de los computadores ejecutando el programa.

El tiempo se calcula en el apartado anterior para poder sacar el MIPS.

3) i0: lw $f0, 0($r1)
i1: lw $f2, 0($r2)
i2: sub.f $f4, $f0, $f2
i3: mul.d $f4, $f4, $f4
i4: add.d $f6, $f6, $f4
i5: add.i $r1, $r1, 4
i6: add $r2, $r2, 4
i7: sub $r3, $r3, 1
i8: bnez $r3, bucle

Contesta a las siguientes preguntas:

a) Dependencias.

En principio no se dice nada del tipo de pipeline, por lo que se supone que es similar al del MIPS, esto significa que solo puede haber dependencias RAW, de las WAW y WAR se suda.

$f0: I0 -> I2

$f2: I1 -> I2

$f4: I2 -> I3

$f4: I3 -> I4

$r3: I7 -> I8

De todas maneras, las dependencias WAW y WAR, en caso de que se quieran calcular, serían:

WAR:

$r1: I0 -> I5

$r2: I1 ->I6

WAW:

$f4: I2 -> I3

b) Elabora un diagrama de tiempo, con pipeline de 5 etapas sin optimizaciones. - No hay forwarding.
- La arquitectura permite que una instrucción escriba en el registro y otra instrucción lea ese mismo registro sin problemas en el mismo ciclo de reloj.
- Las bifurcaciones se tratan vaciando el pipeline.

- Las referencias a memoria requieren un ciclo de reloj.

c) ¿Cuántos ciclos se necesitan para ejecutar N iteraciones del bucle? 19N+2

d) Elabora un diagrama de tiempo MIPS:
- Forwarding.
- Las bifurcaciones se tratan prediciendo todos los saltos como tomad

e) ¿Cuántos ciclos se necesitan para ejecutar N iteraciones del bucle? 11N+3

4) El mismo del boletín MSI.

5) Teoría:

a) ¿Qué es VLIW? Tipo de emisión de instrucciones y planificación que realiza.

Las arquitecturas VLIW son aquellas que operan mediante instrucciones que engloban varias operaciones (accesos a memorias, varias operaciones con enteros, etc.) en una única instrucción que se denomina palabra de instrucción muy larga (“Very Long Instruction Word”).

La emisión de instrucciones se realiza en dos etapas: una primera etapa en la que el compilador genera una secuencia de instrucciones en forma de una palabra de instrucción muy larga o VLIW, que contiene múltiple operaciones que se pueden ejecutar simultáneamente; y una segunda etapa en la cual se realiza una ejecución de la VLIW en el procesador, donde se ejecutan todas las operaciones que esta contiene simultáneamente.

El procesador VLIW no tiene unidad de planificación dinámica como en el caso de arquitecturas superescalares, sino que tiene planificación estática, llevada a cabo por el compilador. Este es el encargado de determinar las dependencias de recursos y datos entre las operaciones a ejecutar en la VLIW y programar un orden de ejecución que garantice que se proceda correctamente. Esto significa que el compilador debe ser capaz de identificar y programar solo aquellas operaciones que se puedan ejecutar simultáneamente sin causar conflictos de recursos o dependencias de datos.

b) ¿Qué es la intensidad operacional? Si tengo un procesador multinúcleo, como afecta aumentar la intensidad operacional al ancho de banda de memoria y en qué unidades se mide el ancho de banda.

La intensidad operacional es una métrica de rendimiento, la cuál expresa el número de operaciones realizadas por cada byte transferido desde o hacia la memoria. Aumentar la intensidad operacional añade una sobrecarga al ancho de banda de memoria ya que seguramente se puedan realizar muchas más operaciones a la vez de las que soporte el ancho de banda. El ancho de banda se mide en bytes por segundo.

c) ¿En qué consiste el false sharing?¿Cómo afecta al tráfico de red asociado a protocolos basados en snooping?

El false sharing o falsa compartición es un fenómeno producido en sistemas multiprocesadores con coherencia caché (CC-NUMA) cuando dos variables independientes dentro de la misma línea caché son accedidas por distintos procesadores, invalidando toda la línea cada vez que se modifica una de las variables. Pese a acceder a diferentes variables, es el bloque completo el que se transmite entre procesadores.

En los protocolos basados en snooping se utilizan mensajes de snoop para detectar y propagar los cambios en la caché de un procesador a los demás procesadores que comparten la misma línea de caché, es decir, los diferentes procesadores están constantemente espiando el bus de datos para observar las otras cachés y así mantener la coherencia.

El tráfico de red asociado a este tipo de protocolos, el false sharing puede generar mensajes de snoop innecesarios que aumentan en gran medida el tráfico de red.

d) ¿En qué consiste la planificación dinámica mediante Tomasulo?¿Cómo resuelve las WAR?

La planificación dinámica de Tomasulo se encarga de gestionar las estaciones de reserva y buffers de reordenamiento para evitar dependencias de datos o recursos. Se desenvuelve de la siguiente manera, dividida en 3 pasos:

1. Emitir:
    a. Se obtiene la instrucción de la cola.

    b. Si la estación de recurso asociada a la instrucción está disponible, emite la instrucción a esta con los valores del operando, si están disponibles.
    c. En caso contrario (los valores no están disponibles), se detiene la instrucción.
2. Ejecutar:
    a. Cuando el operando está disponible, lo almacena en cualquier estación de reserva que lo esté esperando.
    b. Cuando todos los operandos están listos, se emite la instrucción.
    c. Las cargas y el almacenamiento se mantienen en el orden del programa por la dirección efectiva.
    d. No se permite iniciar la ejecución de ninguna instrucción hasta que se hayan completado las ramas que le preceden en el orden del programa.
3. Escribir el resultado:
    a. Se escribe el resultado en las estaciones de reserva y en los buffers de almacenamiento.

NOTA:

Estación de reserva: Estructura de hardware en un procesador que se utiliza para gestionar la ejecución de instrucciones en un ordenador. Se encarga de recibir instrucciones del programador, decodificarlas y reservar los recursos necesarios para ejecutarlas, como registros y unidades funcionales. Una vez que se han reservado los recursos, la estación de reserva mantiene la instrucción en espera hasta que los recursos necesarios estén disponibles y se pueda ejecutar la instrucción.

Este algoritmo resuelve las WAR mediante el renombrado de registros.