AMPLIACIÓN DE ESTRUCTURA DE COMPUTADORES

Grado en Ingeniería Informática

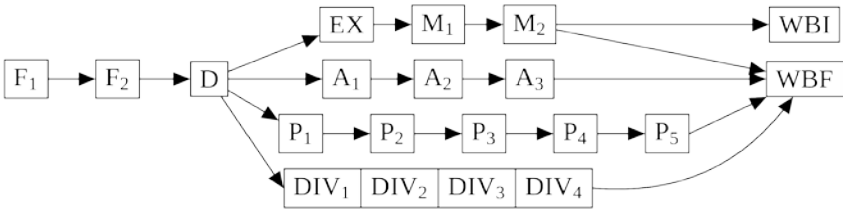
Examen Final – Problemas – 22 de junio de 2022

IMPORTANTE: Es necesario entregar esta hoja junto con los folios del examen.

- ❑ Grupo 1
- ❑ Grupo 2
- ❑ Grupo 3
- ❑ Grupo 9

Apellidos, Nombre: _____

1. (*2,5 puntos*) Tenemos un procesador con el siguiente *pipeline*:



Los accesos a las cachés de instrucciones y datos tardan dos ciclos, están segmentados y se producen en las etapas F_x y M_x. La etapa D realiza la decodificación de instrucciones y en ella se resuelven los saltos. Las instrucciones enteras se ejecutan en la etapa EX. La suma en coma flotante está segmentada, tarda 3 ciclos y se realiza en las etapas A_x. La multiplicación en coma flotante también está segmentada, tarda 5 ciclos y se realiza en las etapas P_x. La división, sin embargo, no está segmentada, tarda 4 ciclos y se realiza en las etapas DIV_x. Se utilizan bancos de registros separados para enteros y coma flotante, y la escritura se realiza en las etapas WBI y WBF, respectivamente. Se puede escribir en un registro durante la primera mitad del ciclo en una etapa WBI o WBF y leerlo durante la segunda mitad del ciclo en una etapa D. No se ha implementado ningún tipo de adelantamiento y los saltos se predicen estáticamente como no tomados. La frecuencia es de 1.5 GHz.

Hemos ejecutado un programa de prueba en el procesador anterior para analizar su rendimiento. Se han ejecutado 10.000.000 instrucciones, de las cuales el 11% son almacenamientos, el 28% son cargas, el 16% son saltos condicionales, el 25% son aritmético-lógicas de tipo entero y el 20% son operaciones de coma flotante (de las cuales el 50% son sumas, el 35% son multiplicaciones y el 15% son divisiones). Se han observado las siguientes situaciones:

1. El 75% de los saltos condicionales se toman.
2. El 50% de las instrucciones aritméticas enteras va seguida por otra instrucción aritmética que usa el dato.
3. El 10% de las instrucciones de carga van seguidas de una instrucción aritmética que consume el dato.
4. El 10% de las instrucciones de coma flotante va seguida de otra de exactamente el mismo tipo (suma, multiplicación o división) que no utiliza el resultado de la primera.

Teniendo en cuenta todo lo anterior:

a) *(1 punto)* Enumere los riesgos que pueden aparecer en cada una de las situaciones mencionadas, incluyendo para cada caso una explicación, un ejemplo de una secuencia de instrucciones que lo produce, el número de ciclos de parada necesarios para resolverlo y el número total de veces que se produciría el riesgo en el programa analizado.
b) *(1 punto)* Suponga que se modifica el *pipeline* para que realice todos los adelantamientos posibles y explique cómo afecta este cambio a cada uno de los riesgos mencionados en el apartado anterior.
c) *(0.5 puntos)* Calcule el tiempo de ejecución del programa con el *pipeline* del aparatado b).

2. (2.5 puntos) Tenemos un procesador con una jerarquía de memoria consistente en un primer nivel (L1) de cachés separado para instrucciones y datos y un segundo nivel (L2) de caché unificado. Los fallos del segundo nivel de caché buscan el bloque en memoria principal. El tiempo de acceso al TLB tanto de instrucciones como de datos es de 1 ciclo. Las cachés de primer nivel tienen un tiempo de servicio en caso de acierto de 4 ciclos, están segmentadas y son virtualmente indexadas y físicamente etiquetadas. El tiempo de acceso al segundo nivel de caché es de 20 ciclos y el tiempo de acceso a memoria principal es de 100 ciclos.

Ejecutamos un programa con un 15% de cargas y un 10% de almacenamientos. Los TLBs de instrucciones y datos muestran una tasa de fallos de un 1%. Cuando se produce un fallo en cualquiera de los TLBs, el tiempo de acceso a memoria principal también es de 100 ciclos. La tasa de fallos de la caché de instrucciones es de un 1% mientras que la de datos es de un 20%. La tasa de fallos de L2 es del 50%. Las cachés L1 y L2 son de post-escritura y ambas tienen un búfer de post-escritura que nunca se llena y al que se accede solo en caso de fallos en la caché. El 5% de los accesos al correspondiente búfer de post-escritura encuentra el dato buscado. El acceso al búfer de post-escritura tiene un coste de un ciclo.

a) (0.75 puntos) Calcula la tasa de fallos global de L1 y L2.
b) (1 punto) Calcula el tiempo medio de acceso a memoria para datos (asumiendo la jerarquía de cachés y TLB completa).
c) (0.75 puntos) Como la tasa de acierto de la L2 es baja, decidimos añadir un mecanismo de pre-búsqueda que trae líneas de caché a un búfer de pre-búsqueda. El búfer se accede ante un fallo de L2 y de búfer de post-escritura, tiene una latencia de acceso de 1 ciclo y el 50% de los accesos al búfer de pre-búsqueda encuentra la línea buscada. Calcula de nuevo el tiempo medio de acceso a memoria para datos.

SOLUCIONES

SOLUCIÓN EJERCICIO 1

a) Para cada uno de los casos, los riesgos que se pueden dar son los siguientes:

- Caso 1:
  - Se produce un **riesgo de control** que provocará una detención de 2 ciclos en el caso de los saltos tomados.
  - Ejemplo:
    BEQZ R1, label
  - Ocurriría 10.000.000 × 0,16 × 0,75 veces en el programa analizado = 1.200.0000 veces.
- Caso 2:
  - Se produce un **riesgo RAW** que provocará 3 ciclos de parada.
  - Ejemplo:
    ADD R1, R2, R3
    ADD R4, R1, R5
  - Ocurrirá 10.000.000 × 0,25 × 0,50 veces = 1.250.000 veces.
- Caso 3:
  - Se produce un **riesgo RAW** que provocará 3 ciclos de parada.
  - Ejemplo:
    LW R1, 100(R2)
    ADD R3, R1, R4
  - Ocurrirá 10.000.000 × 0,28 × 0,10 veces = 280.000 veces.
- Caso 4:
  - Se produce un **riesgo estructural** en el caso de las instrucciones de división que provoca 3 ciclos de parada. No se produce ningún riesgo en el caso de las sumas o multiplicaciones.
  - Ejemplo:
    DIVD F2, F4, F6
    DIVD F10, F8, F6
  - Ocurrirá 10.000.000 × 0,20 × 0,15 × 0,10 veces = 30.000 veces.

b) Para cada caso:

- Caso 1:
  - No cambia.
- Caso 2:
  - La penalización debida al riesgo RAW desaparece.
- Caso 3:
  - La penalización del riesgo RAW se reduce a 2 ciclos.
- Caso 4:
  - No cambia.

c) El CPI ideal del *pipeline* es 1, por lo que el número de ciclos del programa será NI × (1 + detenciones). Las detenciones las hemos detallado en el apartado anterior, por lo que en total el número de ciclos es:

10.000.000 + 10.000.000 × (0,16 × 0,75 × 2 + 0,28 × 0,10 × 2 + 0,20 × 0,15 × 0,10 × 3) = **13.050.000 ciclos**

El tiempo de ciclo es de 1/1,5 GHz = 0,6667 ns, por lo que el tiempo de ejecución es de **8,700 ms**.

SOLUCIÓN EJERCICIO 2

a) La tasa de fallos global se define como el número de fallos de caché (L2 en este caso) dividido por el número total de accesos a memoria generados por la CPU.

Una aplicación con N instrucciones tiene N accesos a instrucciones y 0.25 * N accesos a datos.

La tasa de fallos de L1 de instrucciones es 1%, por lo que se realizan 0.01 * N accesos a L2 por instrucciones.

La tasa de fallos de L1 de datos es 20%, por lo que se realizan 0.2 * 0.25 * N accesos a L2 por datos.

En total son (0.01 + 0.2 * 0.25) * N accesos a L2, es decir 0.06 * N.

La tasa de fallos de L2 es 50%, por lo que el número de fallos de L2 es 0.5 * 0.06 * N = 0.03 * N.

El número de accesos emitidos por la CPU es 1.25 * N.

Por tanto la tasa de fallos global es (0.03 * N) / (1.25 * N) = 0.024, es decir 2.4%.

b) Acceso paralelo a TLB y L1 de datos. Si falla la L1 de datos se mira el búfer de post-escritura. Si falla se mira la L2. Si falla se mira el búfer de post-escritura de la L2. Si falla se va a memoria.

Tma_d = 4 + (0.01 * 100) + (0.2 * (1 + 0.95 * (20 + 0.5 * (1 + 0.95 * 100)))) = 18.12 ciclos.

c) Tan sólo habría que añadir ante un fallo del búfer de post-escritura de la L2 el acceso al búfer de pre-busqueda.

Tma_d = 4+ (0.01 * 100)+(0.2 * (1 + 0.95 * (20 + 0.5 * (1 + 0.95 * (1 + 0.5 * 100))))=13.70 ciclos.