Arquitectura e Ingeniería de Computadores - 3º Grado Ing. Informática - ETSINF - UPV

Ejercicios de la Unidad Temática 1. "Introducción a la Arquitectura de Computadores"

Ejercicio 1.1.

En la fase de diseño de un computador, hay que optar entre dos opciones A y B, que afectan a la frecuencia de reloj y al número de instrucciones de tipo *l*oad/store necesarias. Sabiendo que:

- El reloj de la opción A es 5 % más rápido que el de la B.
- Un programa compilado para A tiene un 30 % de instrucciones del tipo *load/store*, mientras que el mismo programa compilado para B permite reducir en 1/3 el número de instrucciones de este tipo.
- CPI es 1 para todas las instrucciones

decídase qué diseño es mejor.

Ejercicio 1.2.

En un computador *Load/Store*, se ha medido la frecuencia de aparición y los CPI de las siguientes categorias de instrucciones:

Operación	%	CPI
ALU	43	1
LOAD	21	2
STORE	12	2
Saltos	24	2

También se ha observado que el 25 % de las operaciones aritméticas tienen un operando que no se reutiliza, por lo que se plantea la conveniencia de añadir instrucciones aritméticas que tengan uno de sus operandos fuente en la memoria, de manera que secuencias de instrucciones como

```
ld r1,o(r10)
dadd r3,r1,r2
```

pueden ser sustituidas por:

```
daddm r3,o(r10),r2
```

Las nuevas instrucciones tienen un CPI = 2 y al emplearlas también se aumenta en uno el CPI de las instrucciones de salto. Considera que no hay ninguna variación en el coste del computador y justifica cuantitativamente la respuesta.

Ejercicio 1.3.

En un computador con el procesador *MIPS R2000* a 100 MHz y coprocesador se compila y ejecuta el programa *P* que realiza dos millones de operaciones de coma flotante. El compilador traduce cada operación de coma flotante en una instrucción del coprocesador o en una rutina con instrucciones de enteros, según las opciones de compilación indicadas. Variando estas opciones, se ha compilado *P* dos veces, generando dos códigos ejecutables: P_h (código para *MIPS R2000* con coprocesador) y P_s (código para *MIPS R2000* sin coprocesador). Se han ejecutado los dos programas y se han obtenido las siguientes medidas:

Programa	Tiempo de ejecución	CPI medio medido
P_h	92 milisegundos	3.1 ciclos
P_s	1.2 segundos	1.2 ciclos

Se pide:

1. El número de instrucciones que se ejecutan por unidad de tiempo en ambas versiones del programa *P*, expresadas en *MIPS* y el número total de instrucciones ejecutadas en cada caso.

2. El número medio de instrucciones de enteros que sustituye cada operación en coma flotante en P_s.

3. La productividad de ambos programas expresadas en *MFLOPS*.

Ejercicio 1.4.

Se ha escrito en lenguaje C un programa de compresión MP3 para comparar dos procesadores, uno antiguo A y otro más moderno B. El procesador A funciona a 200 MHz, sólo tiene instrucciones de enteros y su CPI es 1. El procesador B funciona a 900 MHz, incorpora las mismas instrucciones enteras y además instrucciones multimedia *IM* y su CPI medio depende de la aplicación, porque las instrucciones de enteros duran 1 ciclo y las instrucciones *IM* duran 3 ciclos. El código ejecutable generado por el compilador puede contener o no instrucciones *IM* según las opciones que se utilicen. Cada instrucción *IM* es equivalente a una serie de instrucciones de enteros que realizan la misma operación. Para las pruebas se preparan dos códigos ejecutables: *H* (con instrucciones *IM*) y *S* (sin ellas). Se anotan los siguientes tiempos de ejecución: el procesador A con el código *S* necesita 50 segundos para comprimir cierta canción mientras que al procesador B con el código *H* le basta con 8 segundos. Además, se ha visto que con el código *H* se ejecutan un 36 % menos de instrucciones que con el código *S*.

1. ¿Cuál es el valor medio del CPI del procesador B ejecutando el código *H*?

2. ¿Cuál es la proporción de instrucciones *M* en el programa *H*?

3. ¿Cuántas instrucciones enteras son equivalentes a una instrucción *IM* en promedio?

4. ¿Cuánto tiempo necesitará el procesador B para comprimir la misma canción con el programa *S*?

Ejercicio 1.5.

En cierta CPU, todas las instrucciones enteras se ejecutan en un ciclo de reloj, mientras que las de coma flotante necesitan 5 ciclos de reloj para completarse. La mayoría de los programas a ejecutar incluyen un 20 % de operaciones en coma flotante. Desde el punto de vista del análisis de costes y prestaciones, ¿es interesante rediseñar la parte de coma flotante del procesador para que sea 5 veces más rápida a costa de duplicar el coste total de la CPU? Justifica la respuesta.

Ejercicio 1.6.

El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2.5 minutos sin éste. Calcular el porcentaje del tiempo de ejecución que el programa realiza operaciones en coma flotante sin el coprocesador instalado.

Ejercicio 1.7.

Ejercicio 1.8.

Un computador dispone de un procesador load/store con un único nivel L1 de memoria cache (interna). Las instrucciones de acceso a memoria son el 30 % del total de ejecutadas, y el 5 % de las mismas generan un fallo que requiere un acceso a la memoria principal durante 10 ciclos de reloj y que se añade al tiempo de ejecución de la instrucción en ausencia de fallo. En ausencia de fallos de cache, el CPI del procesador es 1.

Se estudia la conveniencia de añadir un segundo nivel L2 de cache (externa). Las pruebas realizadas han demostrado que el segundo nivel de cache resuelve el 90 % de fallos de primer nivel, reduciendo la penalización a sólo 2 ciclos de reloj; el 10 % restante sigue penalizando 10 ciclos de reloj.

1. ¿Cuál es la fracción de tiempo que consume el procesador en acceder a la memoria principal con un solo nivel de cache?

2. En promedio, ¿qué penalización sufrirá el procesador a cada fallo de caché L1 si existe la caché L2?

3. ¿Cuál es la aceleración total del computador al añadir L2?

4. ¿Cuál es la fracción de tiempo que consume el procesador en acceder a la memoria principal cuando hay dos niveles de cache?

5. Si el computador está valorado en 1000 € y se hace uso del análisis de prestaciones-coste, ¿cuál sería la máxima inversión que debería hacerse en añadir la cache L2?

Ejercicio 1.9.

Se dispone de un programa P cuyo tiempo de ejecución en un computador C es de $120\ s$. Se pretende acelerar la ejecución de dicho programa realizando varias mejoras en la plataforma de ejecución:

1) Comprar una tarjeta controladora de discos RAID, que permitiría reducir el tiempo de acceso a disco a un 40 % del original, reduciendo el tiempo de ejecución total del programa P a $84\ s$. El coste de dicha tarjeta controladora es de 90 €.

2) Adquirir una tarjeta aceleradora de video, puesto que se sabe que el programa P dedica el 30 % del tiempo a procesos de representación gráfica, valorada en 125 €. La incorporación de dicha tarjeta aceleradora reduciría el tiempo dedicado a representación gráfica a $1/3$ del original.

Dadas estas características:

1. Si el computador original hubiera costado 300 €, ¿qué mejoras serían rentables por separado basándonos en el análisis de coste/prestaciones? Justifica la respuesta.

2. ¿Cuánto debería habernos costado el computador original C para que fuera interesante la incorporación simultánea de ambas mejoras desde el punto de vista de coste/prestaciones?. Justifica la respuesta.

Ejercicio 1.10.

Se ha monitorizado la ejecución de cierta aplicación en un computador, habiéndose obtenido un tiempo de ejecución de 10 minutos, así como que la mayor parte de su tiempo se consume en los procedimientos *P1* y *P2*. Con el objeto de reducir el tiempo global de ejecución se modifica el código de ambos procedimientos. Si la mejora obtenida en cada uno de los dos procedimientos *P1* y *P2* fuera infinita, el tiempo de ejecución de la aplicación descendería hasta 2 minutos. Finalmente, se consigue hacer un procedimiento *P1* que hace su función 5 veces más rápido y mejorar el *P2* en un 100 %, haciendo que el tiempo de ejecución de la aplicación sea de 4.5 minutos.

Calcula cuánto tiempo se consumía en los procedimientos *P1* y *P2* en la aplicación original.

Ejercicio 1.11.

La empresa Farmax dispone de un un supercomputador HAL para trabajos de análisis químico. El ordenador está dotado de un procesador RIX/300 con un reloj a 300 MHz. El sistema se utiliza exclusivamente para ejecutar el programa de análisis *Espectroquimix* que hace uso intensivo de instrucciones de coma flotante. Después de monitorizar el sistema, se han obtenido los datos siguientes:

- La fracción de tiempo que el procesador emplea en ejecutar instrucciones de coma flotante es 75 %.
- La frecuencia de las instrucciones de coma flotante en *Espectroquimix* es: (ver tabla).

- Los CPI de cada tipo de instrucción de CF (ver tabla).

frecuencia (%)	operación	CPI
30	suma	5
10	resta	5
40	multiplicación	10
20	división	40

La dirección de Farmax pide al ingeniero en informática al mando que aumente la velocidad de cálculo. Después de consultar con los proveedores hay que valorar la siguiente mejora: Sustituir el procesador del computador por el procesador RIX/400E, compatible binario con el anterior, que funciona a 400 MHz y tiene la parte de coma flotante muy mejorada. Los CPI de las instrucciones enteras no cambian, pero los de coma flotante quedan así:

operación	CPI
suma	3
resta	3
multiplicación	7
división	25

Calcula:

1. La mejora obtenida ejecutando instrucciones enteras.
2. El CPI medio de las instrucciones de coma flotante de los procesadores RIX/300 y RIX/400E cuando ejecutan *Espectroquimix*.
3. La productividad en MFLOPS obtenida con el procesador RIX/300 cuando ejecuta *Espectroquimix*.
4. La aceleración obtenida en la ejecución de código en coma flotante por el cambio de procesador.
5. La aceleración global obtenida en la ejecución de *Espectroquimix*.
6. La fracción del tiempo que el procesador RIX/400E emplea en ejecutar instrucciones de coma flotante.
7. La productividad en MFLOPS obtenida con el procesador RIX/400E cuando ejecuta *Espectroquimix*.

Ejercicio 1.12.

Un servidor de Internet, dedicado a servir peticiones de red, incluye una placa base con dos procesadores con cuatro núcleos cada uno basados en la microarquitectura AMD Barcelona. Cada procesador dispone de dos controladores de memoria que distribuyen equitativamente el ancho de banda entre todos los núcleos. Las aplicaciones que se ejecutan en el servidor están compuestas de muchas tareas independientes que se ejecutan concurrentemente. Cada tarea pasa el 50 % de su tiempo de ejecución procesando instrucciones, y el 30 % realizando accesos a memoria que no pueden solaparse con la ejecución de instrucciones.

Despues de años de funcionamiento la compañia necesita actualizar el servidor para proveer de servicio a más clientes. Entre las opciones disponibles se consideran dos configuraciones: a) Una placa con dos procesadores Intel Nehalem EX o b) Una placa con dos procesadores AMD Magny-Cours. Cada procesador de Intel incluye 8 nucleos y 8 controladores de memoria, mientras que cada uno de los AMD dispone de 12 nucleos y 4 controladores. Asumiendo que cada uno de los nucleos Intel y AMD son igual de rápidos, que ambos tipos de núcleos son un 20 % más rápidos que un núcleo AMD Barcelona, y que cada controlador de los nuevos procesadores es exactamente igual de rápido que los del AMD Barcelona ¿Qué configuración obtendrá la máxima aceleración?

Ejercicio 1.13.

Se plantea modificar el diseño de la arquitectura de un procesador *l*oad/store para añadir un indicador de acarreo. La modificación afecta al diseño de la UAL y al juego de instrucciones. El análisis del uso del juego de instrucciones y del *CPI* medio del procesador original, con reloj a 500 MHz, es el siguiente:

tipo	frecuencia	*CPI*
aritmética	50 %	1
carga	20 %	2
almacenamiento	10 %	1.2
bifurcación	20 %	1.2

Se ha valorado que la modificación permitirá ahorrar una de cada diez instrucciones aritméticas, pero el *CPI* de éstas subirá a 1.2 y el de las bifurcaciones a 1.5. Por otra parte, al rediseñar el decodificador, la UAL y la detección de riesgos, se ha visto que la máxima frecuencia de reloj aplicable es de 400 MHz.

Estudia la mejora siguiendo estos pasos:

1. Calcula el *CPI* del procesador original
2. Calcula el *CPI* del procesador modificado
3. Determina qué diseño es más rápido y cuantifica la respuesta

Ejercicio 1.14.

Se tiene un procesador load/store de 32 bits dotado de instrucciones que manipulan *bytes*, *halfwords* y *words*. Sobre este procesador se ha monitorizado la aplicación del programa TEX, y se ha obtenido que el 11 % de las referencias a datos se hacen a *bytes* y *halfwords* y el resto a *words*. También se sabe que el 36 % de las instrucciones del programa son de acceso a la memoria, siendo las instrucciones de *load* el doble de frecuentes que las de *store*. Finalmente, se ha medido que CPI=1.

Las instrucciones de lectura de memoria son las mismas del juego del *MIPS*: (*lb*, *lbu*, *lh*, *lhu* y *lw*). Los mismo se puede decir de las escrituras (*sb*, *sh* y *sw*)

Con el objeto de mejorar las prestaciones, se plantea realizar las siguientes modificaciones:

- Eliminar las instrucciones de acceso a *bytes* y *halfwords* del juego de instrucciones. Como consecuencia, los programas que necesiten esta funcionalidad deberán utilizar otras instrucciones del procesador:

Con acceso a *bytes*	Sin acceso a *bytes*
`lb/lbu/lh/lhu r,A`	`lw r,A'` `extract n,r`

`sb/sh r,A`	`lw r',a'`
	`insert n,r,r'`
	`sw r',a'`

  Las nuevas instrucciones de inserción (*insert*) y extracción (*extract*) disponen de las versiones adecuadas para procesar todos los tipos de datos originales *bytes* y *halfwords* con y sin signo.

- Aumentar la frecuencia de reloj, al simplificar el diseño de la interfaz del núcleo del procesador con la memoria cache.

¿En cuánto habrá que incrementar la frecuencia de reloj para que sea interesante incorporar las modificaciones propuestas?

Ejercicio 1.15.

Un equipo de arquitectos de computadores está estudiando mejorar el diseño de un computador que dispone de un juego de instrucciones semejante al *MIPS*. Se plantea la conveniencia de incluir en el juego las instrucciones de manejo de la pila, `push` y `pop` que permitirían sustituir secuencias de instrucciones como:

```
...
sw r1,0[sp]   ; apila r1 y r2
sw r2,-4[sp]
subi sp,sp,8
...
lw r4,0[sp]   ; desapila r4
addi sp,sp,4
...
```

por:

```
...
push r1  ; apila r1 y r2
push r2
...
pop r4   ; desapila r4
...
```

Se sabe que el compilador para este computador sólo utilizará la pila como soporte para las llamadas a subprograma, tanto para el paso de parámetros como para salvar o restaurar el valor de los registros destinados a variables locales, como muestran los siguientes fragmentos de código:

Programa principal:

```
...
sw r1,0[sp]     ; llamada a subprograma
sw r2,-4[sp]    ; (dos parámetros)
subi sp,sp,8
jal subprograma
addi sp,sp,8
...
```

Subprograma:

```
; Punto de entrada al subprograma
sw r1,0[sp],    ; crea espacio para tres
sw r2,-4[sp]    ;      variables locales
sw r3,-8[sp]
sub sp,sp,#12
...             ; comienza el código ...
...
lw r3,0[sp]     ; vuelta al prog. principal
lw r2,4[sp]
lw r1,8[sp]
add sp,sp,#12
jr r31
```

De las medidas de uso del computador sin estas instrucciones se tiene que el 1 % de las instrucciones son llamadas a subprograma. También se dispone de la siguiente estadística del número de variables locales y parámetros de los subprogramas:

Var. loc.	%	Parámetros	%
0	30	0	45
1	25	1	20
2	20	2	15
3	15	3	10
4	10	4	10

Para poder implementar las dos instrucciones, es necesario reducir la frecuencia del reloj, pero se sabe que no cambiará el CPI medio. Si en el diseño original la frecuencia de reloj era de 100 MHz, ¿cuál será la frecuencia mínima del diseño modificado para que interese la modificación propuesta?

Ejercicio 1.16.

Estudia la posibilidad de añadir un nuevo modo de direccionamiento indexado al *MIPS* para instrucciones de *load* y *store*. La dirección del operando en memoria se calcula mediante la suma de dos registros y un desplazamiento de 11 bits, de manera que secuencias de instrucciones como

```
dadd r1,r1,r2
ld rd,o(r1)
```

pueden ser sustituidas por:

```
ldi rd,o(r1,r2)
```

El CPI no queda afectado, pero el periodo de reloj del MIPS mejorado es 5 % más largo que el original. Las medidas tomadas sobre el MIPS original indican que el 24 % de las instrucciones ejecutadas son del tipo *load/store*, y en el 10 % de los casos pueden ser sustituidas por las nuevas instrucciones.

Deduce cuál de las dos máquinas es más rápida, cuantificando su velocidad respecto de la más lenta.

Ejercicio 1.17.

Las extensiones SIMD para MIPS64, conocidas como *MIPS64 SIMD Architecture* (MSA) añaden un nuevo banco de registros con 32 registros de 128 bits (registros `w0`, `w1`, etc.). Dependiendo de la instrucción, cada registro puede tratarse como un vector de 16 bytes (8 bits por componente del vector), un vector de 8 *halfwords* (16 bits por componente), un vector de 4 *words* (32 bits por componente), o un vector de 2 *doublewords* (64 bits por componente).

Para especificar el tipo de componente, la síntaxis de una instrucción SIMD puede incluir uno de los posibles sufijos: `.b`, `.h`, `.w`, y `.d`; que corresponden a bytes, halfwords, words, y doublewords, respectivamente. Así, la instrucción `addv.w w5,w1,w2` suma uno a uno los 4 componentes **word** de los registros vectoriales `w1` y `w2` (es decir, suma el primer componente de `w1` con el primer componente de `w2`, el segundo con el segundo, y así sucesivamente) y almacena los componentes del vector resultante en el registro `w5`.

Se pretende usar las extensiones MSA para acelerar el siguiente algoritmo, que realiza la operación vectorial $\vec{z} = A \cdot \vec{x} + \vec{y}$:

```
    addi r10,r0,N    ; N es el tamaño de los vectores
    addi r11,r0,0
    addi r12,r0,A
loop:
    lw r20,X(r11)
```

```
    lw r21,Y(r11)
    mul r20,r20,r12
    add r21,r21,r20
    sw r21,Z(r11)
    addi r11,r11,+4
    addi r10,r10,-1  ; 1 iteración procesa 1 componente de los vectores x, y, z
    bne r10,r0,loop
```

Para ello se implementan dos versiones que hacen uso de las extensiones MSA. La primera versión (MSA1) ejecuta el siguiente código (las instrucciones MSA están marcadas en negrita):

```
    addi r10,r0,N    ; N es el tamaño de los vectores
    addi r11,r0,0
    ldi.w w12,A
loop:
    ld.w w20,X(r11)
    ld.w w21,Y(r11)
    mulv.w w20,w20,w12
    addv.w w21,w21,w20
    st.w w21,Z(r11)
    addi r11,r11,+16
    addi r10,r10,-4  ; 1 iteración procesa 4 componentes de los vectores x, y, z
    bne r10,r0,loop
```

Teniendo en cuenta que cada iteración del bucle en MSA1 procesa 4 componentes (en vez de 1 componente, como el algoritmo original). Responde a las siguientes preguntas:

1. Asumiendo que N es un múltiplo de 4, ¿cuantas instrucciones ejecuta MSA1 en total?

2. Teniendo en cuenta que el CPI de todas las instrucciones es 1 y que las extensiones MSA no afectan a la frecuencia del procesador ¿cuál es la aceleración obtenida por MSA1 respecto al algoritmo original?

3. La segunda versión (MSA2) mejora MSA1 mediante el uso de la instrucción maddv, que combina las instrucciones `mulv.w` y `addv.w`. De manera que el siguiente fragmento de código:

   ```
   mulv.w w20,w20,w12
   addv.w w21,w21,w20
   ```

   se sustituye por:

   ```
   maddv.w w21,w20,w12
   ```

   Sin embargo, debido a las dependencias de datos con las instrucciones `ld.w` y al hecho de que realiza dos operaciones, el CPI de la instrucción `maddv.w` es 3. Teniendo esto en cuenta, ¿cuanto debería acelerarse la frecuencia del procesador para que la ejecución de MSA2 ofrezca mejores prestaciones que la de MSA1?