

Arquitectura de Computadores



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada

se crean derivados de estos datos originales y no para fines comerciales.

Arquitectura de Computadores

Los Del DGIIM, losdeldgiim.github.io

Arturo Olivares Martos

Granada, 2023-2024

Índice general

1. Relaciones de Problemas

1.1. Arquitecturas Paralelas: Clasificación y Prestaciones

Ejercicio 1.1.1. En el código de prueba (benchmark) que ejecuta un procesador no segmentado que funciona a 300 MHz, hay un 20 % de instrucciones **LOAD** que necesitan 4 ciclos, un 10 % de instrucciones **STORE** que necesitan 3 ciclos, un 25 % de instrucciones con operaciones de enteros que necesitan 6 ciclos, un 15 % de instrucciones con operandos en coma flotante que necesitan 8 ciclos por instrucción, y un 30 % de instrucciones de salto que necesitan 3 ciclos.

1. ¿Cuál es la ganancia que se puede obtener por reducción a 3 ciclos de las instrucciones con enteros?

Resumimos los datos del enunciado en la siguiente tabla:

I_i	CPI_i	NI_i	CPI_i^p
LD	4 ciclos	0,2 NI	
ST	3 ciclos	0,1 NI	
FX	6 ciclos	0,25 NI	3 ciclos
FP	8 ciclos	0,15 NI	3 ciclos
BR	3 ciclos	0,3 NI	

Donde CPI_i^p es la reducción en el tiempo de instrucciones del tipo correspondiente.

La expresión de la ganancia es:

$$S = \frac{T_b}{T_p}$$

Donde T_p es el tiempo mejorado y T_b el base.

$$T_b = \overbrace{NI \cdot CPI}^{\text{num. ciclos}} \cdot T_c = Numciclos \cdot T_c = \left(\sum_i NI_i \cdot CPI_i \right) \cdot T_c$$

Donde T_c es el tiempo de ciclo, CPI ciclos por instrucción y NI el número de instrucciones. Calculamos estos dos para la expresión de S , en función de NI y de T_c .

$$T_b = NI(\underbrace{0,2 \cdot 4c}_{LD} + \underbrace{0,1 \cdot 3c}_{ST} + \underbrace{0,3 \cdot 3c}_{BR} + \underbrace{0,15 \cdot 8c}_{FP} + \underbrace{0,25 \cdot 6c}_{FX}) \cdot T_c$$

$$= NI(3,2 + \underbrace{1,5}_{FX}) \cdot T_c = NI \cdot (4,7) \cdot T_c$$

$$T_p = NI(3,2 + 0,25 \cdot 3c) \cdot T_c = NI \cdot (3,2 + 0,75) \cdot T_c = NI \cdot 3,95 \cdot T_c$$

$$S = \frac{NI \cdot 4,7 \cdot T_c}{NI \cdot 3,95 \cdot T_c} = 1,8987$$

2. ¿Cuál es la ganancia que se puede obtener por reducción a 3 ciclos de las instrucciones en coma flotante?

Ejercicio 1.1.2. Un circuito que implementaba una operación en un tiempo de $T_{op} = 450$ ns se ha segmentado mediante un cauce lineal con cuatro etapas de duración $T1 = 100$ ns, $T2 = 125$ ns, $T3 = 125$ ns y $T4 = 100$ ns respectivamente, separadas por un registro de acoplo que introduce un retardo de 25 ns.

1. ¿Cuál es la máxima ganancia de velocidad posible? ¿Cuál es la productividad máxima del cauce?

Tenemos que el ciclo de reloj es de $T_c = 125ns + 25ns = 150ns$. La ganancia de velocidad es:

$$S(N) = \frac{T^b(N)}{T^s(N)} = \frac{N \cdot T_R}{TLI + (N - 1)T_c} = \frac{N \cdot T_R}{4 \cdot T_c + (N - 1)T_c}$$

donde el 4 se debe a que es el número de etapas del cauce.

La ganancia máxima es:

$$S(N \gg) = \lim_{N \rightarrow \infty} S(N) = \lim_{N \rightarrow \infty} \frac{N \cdot T_R}{4 \cdot T_c + (N - 1)T_c} = \frac{T_R}{T_c}$$

2. ¿A partir de qué número de operaciones ejecutadas se consigue una productividad igual al 90 % de la productividad máxima?

Ejercicio 1.1.3. En un procesador sin segmentación de cauce, determine cuál de estas dos alternativas para realizar un salto condicional es mejor:

- ALT1: Una instrucción **COMPARE** actualiza un código de condición y es seguida por una instrucción **BRANCH** que comprueba esa condición. Se usan dos instrucciones.
- ALT2: Una sola instrucción incluye la funcionalidad de las instrucciones **COMPARE** y **BRANCH**. Se usa una única instrucción.

Hay que tener en cuenta que hay un 20 % de instrucciones **BRANCH** para ALT1 en el conjunto de programas de prueba; que las instrucciones **BRANCH** en ALT1 y **COMPARE+BRANCH** en ALT2 necesitan 4 ciclos mientras que todas las demás necesitan sólo 3; y que el ciclo de reloj de la ALT1 es un 25 % menor que el de la ALT2,

dado que en este caso la mayor funcionalidad de la instrucción **COMPARE+BRANCH** ocasiona una mayor complejidad en el procesador.

Como el tiempo de ciclo de reloj depende de la ejecución más lenta, es normal que este cambie (como se especifica en el enunciado). La relación entre estos es la siguiente:

$$T_c^1 = T_c^2 - 0,25T_c^2 = 0,75T_c^2$$

Resumimos los datos del enunciado en la siguiente tabla:

I_i^1	CPI_i^1	NI_i^1	I_i^2	CPI_i^2	NI_i^2
br	4 ciclos	$0.2 \cdot NI^1$	cmpbr	4 ciclos	$0.2 \cdot NI^1$
cmp	3 ciclos	$0.2 \cdot NI^1$			
Demás	3 ciclos	$0.6 \cdot NI^1$	Demás	3 ciclos	$0.6 \cdot NI^1$
		NI^1			$0.8 \cdot NI^1 = NI^2$

Tenemos que ver qué alternativa nos da un tiempo de ejecución menor (tengamos en cuenta que el tiempo de ciclo de cada uno no es el mismo, por lo que tenemos que pasarlo todo al mismo tiempo de ciclo):

$$T_{CPU}^1 = NI^1 \cdot (\underbrace{0,2 \cdot 4c}_{br} + \underbrace{0,8 \cdot 3c}_{cmp+Resto}) \cdot T_c^1 = NI^1 \cdot 3,2 \cdot 0,75 \cdot T_c^2 = NI^1 \cdot 2,4 \cdot T_c^2$$

$$T_{CPU}^2 = NI^1 \cdot (\underbrace{0,2 \cdot 4c}_{cmpbr} + \underbrace{0,6 \cdot 3c}_{Resto}) \cdot T_c^2 = NI^1 \cdot 2,6 \cdot T_c^2$$

Por ser $T_{CPU}^1 < T_{CPU}^2$, concluimos que la opción ALT1 es la mejor, en cuanto a tiempos de ejecución.

Ejercicio 1.1.4. ¿Qué ocurriría en el problema del ejercicio anterior (Ejercicio ??) si el ciclo de reloj fuese únicamente un 10 % mayor para la ALT2?

Ejercicio 1.1.5. Considere un procesador no segmentado con una arquitectura de tipo **LOAD/STORE** en la que las operaciones sólo utilizan como operandos registros de la CPU. Para un conjunto de programas representativos de su actividad se tiene que el 43 % de las instrucciones son operaciones con la ALU (3 CPI), el 21 % **LOADs** (4 CPI), el 12 % **STOREs** (4 CPI) y el 24 % **BRANCHs** (4 CPI). Se ha podido comprobar que un 25 % de las operaciones con la ALU utilizan operandos en registros que no se vuelven a utilizar. Compruebe si mejorarían las prestaciones si, para sustituir ese 25 % de operaciones, se añaden instrucciones con un dato en un registro y otro en memoria. Tengan en cuenta en la comprobación que para estas nuevas instrucciones el valor de CPI es 4 y que añadirlas ocasiona un incremento de un ciclo en el CPI de los **BRANCH**, pero no afectan al ciclo de reloj.

Resumimos los datos del enunciado en la siguiente tabla:

I_i^1	CPI_i^1	NI_i^1	I_i^2	CPI_i^2	NI_i^2
Instrucción ALU	3 ciclos	$0,43 \cdot NI$	ALU r,r	3 ciclos	$0,2325 \cdot NI^1$
			ALU r,m	4 ciclos	$0,1075 \cdot NI^1$
LOADs	4 ciclos	$0,21 \cdot NI^1$	LOADs	4 ciclos	$0,1225 \cdot NI^1$
STOREs	4 ciclos	$0,12 \cdot NI^1$	STOREs	4 ciclos	$0,12 \cdot NI^1$
BRANCHs	4 ciclos	$0,24 \cdot NI^1$	BRANCHs	5 ciclos	$0,24 \cdot NI^1$
					$0,8925 \cdot NI^1 = NI^2$

Donde para completar la columna NI_i^2 hemos usado que:

$$\begin{aligned}
 NI_{\text{ALU r,r}}^1 &= 0,75(0,43 \cdot NI^1) = 0,2325 \cdot NI^1 \\
 NI_{\text{ALU r,m}}^1 &= 0,25(0,43 \cdot NI^1) = 0,1075 \cdot NI^1 \\
 NI_{\text{LOADs}}^1 &= 0,21 \cdot NI^1 - 0,1075 \cdot NI^1 = 0,1225 \cdot NI^1
 \end{aligned}$$

Calculamos los tiempos en CPU:

$$\begin{aligned}
 T_{CPU}^1 &= NI^1 (\overbrace{0,43 \cdot 3c}^{\text{ALU r,r}} + (0,21 + 0,12 + 0,24) \cdot 4c) \cdot T_c \\
 &= NI^1 \cdot 3,57 \cdot T_c
 \end{aligned}$$

$$\begin{aligned}
 T_{CPU}^2 &= NI^1 (0,3235 \cdot 3c + (0,1075 + 0,1225 + 0,12) \cdot 4c + 0,24 \cdot 4c) \cdot T_c \\
 &= NI^1 \cdot 3,4875 \cdot T_c
 \end{aligned}$$

Y tenemos que $T_{CPU}^2 < T_{CPU}^1$, luego sí que mejorarían las prestaciones.

Ejercicio 1.1.6. Se ha diseñado un compilador para la máquina LOAD/STORE del problema anterior (Ejercicio ??). Ese compilador puede reducir en un 50 % el número de operaciones con la ALU, pero no reduce el número de LOADs, STOREs, y BRANCHs. Suponiendo que la frecuencia de reloj es de 50 Mhz, ¿Cuál es el número de MIPS y el tiempo de ejecución que se consigue con el código optimizado? Compárelos con los correspondientes del código no optimizado.

Ejercicio 1.1.7. En un programa que se ejecutan en un procesador no segmentado que funciona a 100 MHz, hay un 20 % de instrucciones LOAD que necesitan 4 ciclos, un 15 % de instrucciones STORE que necesitan 3 ciclos, un 40 % de instrucciones con operaciones en la ALU que necesitan 6 ciclos, y un 25 % de instrucciones de salto que necesitan 3 ciclos.

1. Si en las instrucciones que usan la ALU el tiempo en la ALU supone 4 ciclos, determine cuál es la máxima ganancia que se puede obtener si se mejora el diseño de la ALU de forma que se reduce su tiempo de ejecución a la mitad de ciclos.
2. Con qué porcentaje de instrucciones con operaciones en la ALU se podría haber obtenido en los cálculos del apartado una ganancia mayor que 2? Razone su respuesta.

Ejercicio 1.1.8. Suponga que en los programas que constituyen la carga de trabajo habitual de un procesador las instrucciones de coma flotante consumen un promedio del 13 % del tiempo del procesador.

1. Ha aparecido en el mercado una nueva versión del procesador en la que la única mejora con respecto a la versión anterior es una nueva unidad de coma flotante que permite reducir el tiempo de las instrucciones de coma flotante a tres cuartas partes del tiempo que consumían antes. ¿Cuál es la máxima ganancia de velocidad que puede esperarse en los programas que constituyen la carga de trabajo si se utiliza la nueva versión del procesador?

La p de la ley sería $3/4$ y $f = 0,87$. Para calcular la ganancia adicional:

$$S = \frac{T_b}{T_p} = \frac{\cancel{T_b}}{0,87\cancel{T_b} + 0,13\cancel{T_b} \cdot 3/4} = 1,033$$

2. ¿Cuál es la máxima ganancia de velocidad con respecto a la versión inicial del procesador que, en promedio, puede esperarse en los programas debido a mejoras en la velocidad de las operaciones en coma flotante?

Lo que esperamos en el mejor caso es llevar a 0 el tiempo que tarda en ejecutarse la parte de punto flotante.

$$S = \frac{T_b}{T_p} = \frac{\cancel{T_b}}{0,87\cancel{T_b} + 0} = 1,149$$

3. ¿Cuál debería ser el porcentaje de tiempo de cálculo con datos en coma flotante en los programas para esperar una ganancia máxima de 4 en lugar de la obtenida en el apartado ???

Buscamos la f para obtener 4:

$$\frac{\cancel{T_b}}{f\cancel{T_b} + \underbrace{(1-f)\cancel{T_b}}_p} = 4 \Rightarrow f = \frac{1}{4}$$

4. ¿Cuánto debería reducirse el tiempo de las operaciones en coma flotante con respecto a la situación inicial para que la ganancia máxima sea 2 suponiendo que en la versión inicial el porcentaje de tiempo de cálculo con coma flotante es el obtenido en el apartado ???

Se trata de buscar p , suponiendo que tenemos un 75 % de operaciones de coma flotante. Simplemente hay que despejar p .

Ejercicio 1.1.9. Suponga que, en el código siguiente, $a[]$ es un array de números de 32 bits en coma flotante y b un número de 32 bits en coma flotante y que debería ejecutarse en menos de 0,5 segundos para $N = 10^9$:

```
for (i=0; i<N; i++)
    a[i+2]=(a[i+2]+a[i+1]+a[i])*b;
```

1. ¿Cuántos GFLOPS se necesitan para poder ejecutar el código en menos de 0,5 segundos?

$$T(10) < 0,5 \text{ seg}$$

$$GFLOPS = \frac{NumFFP}{T_{CPU} \cdot 10^9} = \frac{3 \cdot N}{T_{CPU} \cdot 10^9} > \frac{3 \cdot N}{0,5 \cdot 10^9}$$

2. Suponiendo que este código en ensamblador tiene 7N instrucciones y que se ha ejecutado en un procesador de 32 bits a 2 GHz. ¿Cual es el número medio de instrucciones que el procesador tiene que poder completar por ciclo para poder ejecutar el código en menos de 0,5 segundos?

Sabemos que el tiempo en CPU lo podemos calcular de la forma:

$$T_{CPU} = NI \cdot CPI \cdot T_c = \frac{NI}{IPC \cdot F}$$

Con:

$$IPC = \frac{1}{CPS} =$$

Sabemos que $NI = 7 \cdot 10^9$, buscamos el valor de IPC y $F = 2 \cdot 10^9$ c/s:

$$\frac{7 \cdot 10^9}{IPC \cdot 2 \cdot 10^9} < 0,5$$

$$IPC > 7$$

3. Estimando que el programa pasa el 75 % de su tiempo de ejecución realizando operaciones en coma flotante, ¿cuánto disminuiría como mucho el tiempo de ejecución si se redujesen un 75 % los tiempos de las unidades de coma flotante?

Sea T_p el tiempo de mejora y T_b el tiempo base:

$$Porreduccion = \frac{T_b - T_p}{T_b} \cdot 100$$

$$T_p = 0,25T_p + 0,75T_b \cdot 0,25 = 0,4375T_b$$

$$\left(1 - \frac{T_p}{T_b}\right) \cdot 100 = \left(1 - \frac{0,4375T_b}{T_b}\right) \cdot 100 = 56,25 \%$$

$$p = 4 \quad f = 0,25$$

Ejercicio 1.1.10. Un compilador ha generado un código máquina optimizado para el siguiente programa

```

par=0; impar=0;
for (i=0;i<N;i++)
    if ((i%2) == 0)
        par=par+c*x[i];
    else
        impar=impar-c*x[i];

```

sin utilizar instrucciones de salto dentro de las iteraciones del bucle (porque se ha usado la técnica de desenrollado el bucle que veremos en el Seminario 4): el código tiene un número de iteraciones de $N/2$, 7 instrucciones fuera del bucle (2 de almacenamiento en memoria, 5 instrucciones para inicializar registros), 9 instrucciones dentro del bucle (4 instrucciones para implementar el bucle for: incremento de la variable de control i , comparación, salto condicional y un salto incondicional; 4 instrucciones coma flotante y 2 instrucciones de carga desde memoria a registro -se leen dos componentes de x). El computador donde se ejecuta dispone de:

- Un procesador superescalar de 32 bits a 2 GHz capaz de terminar dos instrucciones de coma flotante por ciclo y dos instrucciones de cualquier otro tipo por ciclo, excepto instrucciones de carga, cuyo tiempo depende de si hay o no fallo de cache (si no hay fallo de cache suponen 1 ciclo), y las instrucciones de almacenamiento que suponen 1 ciclo.
- Dos caches integradas en el chip de procesamiento (una para datos y otra para instrucciones) de 512 KBytes cada una, mapeo directo, política de actualización de postescritura, líneas de 32 bytes, y latencia de un ciclo de reloj.
- Una memoria principal con latencia de 30 ns. y ciclos burst 6-1-1-1 a través de un bus de memoria de 200 MHz con 64 bits.

Conteste a las siguientes cuestiones:

1. ¿Cuál es la velocidad pico del procesador (en GFLOPS)?
2. ¿Cuál es el tiempo mínimo que tarda en ejecutarse el programa para $N = 211$?
3. ¿Cuántos MFLOPS alcanza el programa?

Observación. Considere que el vector x se almacena en memoria en una dirección múltiplo del tamaño de una línea de cache y que ningún componente está en cache cuando se referencia; N , i estarán en registros de enteros, par , $impar$, c , y $x[]$ son números de 32 bits en coma flotante; dentro del bucle c , par e $impar$ estarán en registros.

Cuestión 1.1.1. Indique cuál es la diferencia fundamental entre una arquitectura CC-NUMA y una arquitectura SMP.

Cuestión 1.1.2. ¿Cuándo diría que un computador es un multiprocesador y cuándo que es un multicomputador?

Cuestión 1.1.3. ¿Un CC-NUMA escala más que un SMP? ¿Por qué?

Cuestión 1.1.4. Indique qué niveles de paralelismo implícito en una aplicación puede aprovechar un PC con un procesador de 4 cores, teniendo en cuenta que cada core tiene unidades funcionales SIMD (también llamadas unidades multimedia) y una microarquitectura segmentada y superscalar. Razone su respuesta.

Cuestión 1.1.5. Si le dicen que un ordenador es de 20 GIPS ¿puede estar seguro que ejecutará cualquier programa de 20000 instrucciones en un microsegundo?

Cuestión 1.1.6. ¿Aceptaría financiar/embarcarse en un proyecto en el que se plantease el diseño e implementación de un computador de propósito general con arquitectura MISD? (Justifique su respuesta).

Cuestión 1.1.7. Deduzca la expresión que se usa para representar la ley de Amdahl suponiendo que se mejora un recurso del procesador, que hay una probabilidad f de no utilizar dicho recurso y que la mejora supone un incremento en un factor de p de la velocidad de procesamiento del recurso.

Cuestión 1.1.8. ¿Es cierto que si se mejora una parte de un sistema (por ejemplo, un recurso de un procesador) se observa experimentalmente que, al aumentar el factor de mejora, llega un momento en que se satura el incremento de velocidad que se consigue? (Justifique la respuesta)

Cuestión 1.1.9. ¿Es cierto que la cota para el incremento de velocidad que establece la ley de Amdahl crece a medida que aumenta el valor del factor de mejora aplicado al recurso o parte del sistema que se mejora? (Justifique la respuesta).

Cuestión 1.1.10. ¿Qué podría ser mejor suponiendo velocidades pico, un procesador superescalar capaz de emitir cuatro instrucciones por ciclo, o un procesador vectorial cuyo repertorio permite codificar 8 operaciones por instrucción y emite una instrucción por ciclo? (Justifique su respuesta).

Cuestión 1.1.11. En la Lección 2 de AC se han presentado diferentes criterios de clasificación de computadores y en el Seminario 0 de prácticas se ha presentado atcgrid. Clasifique atcgrid, sus nodos, sus encapsulados y sus núcleos dentro de la clasificación de Flynn y dentro de la clasificación que usa como criterio el sistema de memoria. Razone su respuesta.

Cuestión 1.1.12. En la Lección 1 de AC se han presentado diferentes criterios de clasificación del paralelismo implícito en una aplicación y en el Seminario 0 de prácticas se ha presentado atcgrid. ¿Qué tipos de paralelismo aprovecha atcgrid? Razone su respuesta.

1.2. Programación paralela