

### Problema 1. Tipos de máquinas

Dada la siguiente expresión donde R, A,B,C y D son variables globales:

$$R = C - (A - B) / (C - D)$$

- a) Programa una secuencia de código que la evalue en una arquitectura de tipo pila con las siguientes operaciones:

```
add/sub/mul/div #pila[top+1]=pila[top] op pila[top+1]; top=top+1
push @           #top=top-1; pila[top]=M[@]
pop @           #M[@]=pila[top]; top=top+1
```

```
push D          push C
push C          sub
sub             pop R
push B
push A
sub
div
```

- b) Programa una secuencia de código que la evalue en una arquitectura de tipo acumulador con las siguientes operaciones:

```
add/sub/mul/div @ #ACC=ACC op M[@]
load @          #ACC=M[@]
store @         #M[@]=ACC
```

```
LOAD A          LOAD A
LOAD@          DIV TMP
SUB B          STORE R
STORE R        LOAD C
LOAD C          SUB R
SUB D          STORE R
STORE TMP
```

### Problema 2. RISC-CISC

Un cachivache electrónico (llamado i-Cachivache) ejecuta repetidamente un programa que no puede tardar más de 2,5 segundos. Para ello el i-Cachivache incorpora un procesador CISC registro-memoria. Al ejecutar el programa compilado para dicho procesador hemos obtenido los siguientes datos:

- $10^9$  instrucciones dinámicas ejecutadas
- CPI 2,5 ciclos/instrucción
- el 30% de las instrucciones tienen un operando en memoria
- el 10% de las instrucciones tienen dos operandos en memoria
- el 70% de los accesos a memoria usan el modo base+desplazamiento
- el 20% de los desplazamientos se codifican con más de 16 bits
- el 20% de los accesos a memoria usan el modo registro indirecto (equivalente a base + desplazamiento cero)
- el 20% de las instrucciones tienen un operando inmediato
- el 15% de los inmediatos se codifican con más de 16 bits.

- a) ¿Cuántos accesos a datos se realizan durante la ejecución del programa?

$$\cancel{10^9(3+2 \cdot .4)} \quad 10^9 \times 3 + 2 \cdot (40\% \cdot 1) = 500 \times 10^6 \text{ Accesos}$$

- b) ¿A qué frecuencia debemos hacer funcionar el procesador para que el programa se ejecute en el tiempo previsto?

$$T_{exe} = N \cdot CPI \cdot F^{-1} \Rightarrow F = \frac{N \cdot CPI}{T_{exe}} = \frac{40 \cdot 2.5}{2.5} = 10^9 \text{ Hz}$$

En la segunda generación del cachivache (el i-Cachivache 2) deseamos sustituir el procesador CISC por un RISC que esperamos tenga un menor consumo y aumente la duración de la batería. Para ello disponemos de un traductor que traduce del lenguaje máquina CISC al RISC. Al traducir el programa hemos obtenido que:

- El 90% de las instrucciones CISC se han podido mapear sobre una instrucción RISC equivalente. El 10% restante ha necesitado 2 instrucciones RISC.
- Los accesos a memoria han necesitado una instrucción adicional de Load/Store.
- Los accesos a memoria que no se han podido implementar usando el modo base+desplazamiento han necesitado otra instrucción adicional para calcular la dirección.
- Los load/store con desplazamientos de más de 16 bits han necesitado una instrucción adicional ya que el RISC

sólo dispone de 16 bits para codificarlos.

- Los inmediatos que necesitan más de 16 bits también han necesitado una instrucción adicional ya que los inmediatos también están limitados a 16 bits.
- c) ¿Cuántas instrucciones dinámicas ejecutará el procesador RISC?

$$10^9 \cdot (0'3 \cdot 2 + 0'1 \cdot 2 * 0'4 \cdot 3 + 0'4 \cdot 0'7 + 0'2 \cdot 2 + 0'2 \cdot 0'15) = 1'606 \cdot 10^9 \text{ int. din.}$$

Al ejecutar el programa traducido en el procesador RISC hemos obtenido un CPI de 1,2 ciclos/instrucción.

- d) ¿A qué frecuencia deberíamos hacer funcionar el procesador RISC?

$$F = \frac{1'606 \cdot 10^9 \cdot 1'12}{2'5} = 0'770 \text{ GHz}$$

El procesador RISC es ligeramente más pequeño que el CISC por lo que su intensidad de fuga y su carga capacitiva son también menores, 10 A y 50 nF en el CISC por solo 8 A y 40 nF para el RISC. Ambos procesadores funcionan con una tensión de alimentación de 1 V. En ambos procesadores la potencia de cortocircuito es despreciable, por lo que solo tendremos en cuenta la potencia de fuga y la de conmutación.

- e) Calcular la energía consumida por ambos procesadores al ejecutar el programa.

$$\text{CISC: } P_{fuga} = 10A \cdot 1V = 10 \text{ W} \\ P_{con.} = 50 \times 10^{-9} F \cdot 1V^2 \cdot 1 \times 10^9 \text{ Hz} = 50 \text{ W} \quad | \quad E = (P_f + P_c) \cdot t = (50 + 10) \cdot 2'5s \\ \text{RISC: } P_{fuga} = 8A \cdot 1V = 8 \text{ W} \\ P_{con.} = 40 \times 10^{-9} F \cdot 1V^2 \cdot 0'72 \times 10^9 \text{ Hz} \approx 30'8 \text{ W} \quad | \quad E = (8W + 30'8W) \cdot 2'5 = 93J$$

- f) Calcular la ganancia en duración de batería del RISC sobre el CISC

$$G = \frac{150}{93} = 1'546 \Rightarrow 54'6\%$$

Al cabo de un tiempo se dispone de un compilador que nos permite compilar el programa directamente para el procesador RISC. Al ejecutar el programa compilado directamente para el RISC, vemos que el número de instrucciones dinámicas se ha reducido a  $1,5 \times 10^9$  instrucciones, pero el CPI ha aumentado a 1,3 ciclos/instrucción. Esto nos permitirá sacar al mercado el i-Cachivache 3 que, a parte de un nuevo diseño más "cool" de la carcasa, es idéntico al i-Cachivache 2, pero con la nueva versión del código).

- g) ¿A qué frecuencia debería funcionar la CPU RISC con el nuevo programa compilado?

$$F = (1'5 \times 10^9 \cdot 1'3) / 2'5 = 780 \text{ MHz}$$

- h) ¿Cuál será la ganancia en duración de batería con el nuevo código?

$$P_T = 80W + (40 \times 10^{-9} \cdot 1V^2 \cdot 7'6 \times 10^8) = 39'2 \text{ W}$$

$$E = P \cdot t = 39'2 \cdot 2'5 = 98J \Rightarrow G = 150 / 98 = 1'53 \Rightarrow 53\%$$

### Problema 3. Microoperaciones

Los actuales procesadores CISC de la familia x86 (tanto de Intel como AMD) traducen internamente las instrucciones de lenguaje máquina x86 a microoperaciones (que denominan uops). En una implementación de x86 tenemos las siguientes uops:

- a) Traducir la siguiente secuencia de instrucciones en x86 a la correspondiente secuencia de microoperaciones

```

        movl $0, %ecx
loop:  cmpl $1000000, %ecx
        jge fin
        movl x, %eax
        imull V(,%ecx,4), %eax
        addl %eax, suma
        incl %ecx
        jmp loop
fin:
    
```

```

        movl %ecx
loop: cmpl %ecx
        jse fin
        load %eax ← x
        add %r4 # V[%ecx+4]
        imul %r2x # r2x+r4
        load %r2
        store suma # r2
        addl %ecx
        jmp loop # %ecx+81
    
```

fin:

- b) Calcular cuantas instrucciones dinámicas y cuantas uops dinámicas se ejecutan

$$10^6 \cdot 7 + 1 = 7 \cdot 10^6 \text{ inst. dinámicas.}$$

$$2 \cdot 10^6 \cdot 10 + 1 = 20 \cdot 10^6 \text{ uops dinámicas.}$$

- c) Al ejecutar este código nuestra CPU obtiene un UPC (uops por ciclo) de 1,3

$$\frac{1}{1,3} \frac{\text{cicles}}{\text{uops}} \Rightarrow \frac{1}{1,3} \frac{\text{Cicles}}{\text{uops}} \cdot 10000 \text{ mil uops} = 2692309 \text{ cicles}$$

- d) Calcular el CPI de este código

$$CPI = \frac{2692309 \text{ cicles}}{2 \cdot 10^6 \text{ inst.}} = 1,098 \text{ ciclos/inst.}$$

Suponiendo una frecuencia de GHz

- e) Calcular el tiempo de ejecución de este código

$$T_{\text{exe}} = 2 \cdot 10^6 \cdot 1,098 \cdot (3 \times 10^{-9})^{-1} = 2,57 \times 10^{-3} \text{ s.}$$

Supongamos que las instrucciones x86 ocupan:

- 1 byte de código de operación (OpCode)
- 1 byte adicional si tiene 2 o más operandos (Modo), las de 1 sólo operando lo codifican dentro del OpCode
- 4 bytes adicionales si alguno de los operandos es un inmediato (incluidos los destinos de los saltos)
- 1 byte adicional (SIB) si accede a memoria
- 4 bytes adicionales si el modo de direccionamiento incluye desplazamiento

Las uops ocupan todas 6 bytes.

- f) Calcular el tamaño del código x86 y el que ocuparían las uops equivalentes.

$$uops = 6 \cdot 11 = 66 \text{ bytes} \Rightarrow x86 = 44 \text{ Bytes}$$

La familia *Sandy Bridge* de Intel (2011) incluye, además de una cache de instrucciones de nivel 1, una cache de micro-uops (que denominan de nivel 0) donde se guardan las uops que ya han sido traducidas para que en caso de hit no sea necesario descodificar y traducir las instrucciones x86.

- g) Calcular el numero de bytes leidos y el ancho de banda efectivo de la cache de instrucciones al ejecutar este código (sin uop cache)

$$10 \cdot 10^6 \text{ uops} \cdot 6 \text{ B} = 60 \text{ MB}$$

$$\text{Ancho banda} = 60 \text{ MB} / 2,57 \times 10^{-3} = 23346 \text{ B/s}$$

- h) Calcular el numero de bytes leidos y el ancho de banda efectivo de la cache de uops al ejecutar este código (con uop cache)

$$44 \text{ Bytes} \cdot 10^6 = 44 \text{ MB}$$

$$\text{Ancho banda} = 44 \text{ MB} / 2,57 = 171260 \text{ B/s}$$

Una de las ventajas de la cache de micro-uops es que se produce un ahorro de energía ya que no es necesario descodificar todas las instrucciones dinámicas. En nuestro procesador, un acceso, tanto a la cache de instrucciones como a la cache de uops, consume 1 nJ (nanoJulio) por byte leido. Descodificar una instrucción consume 10 nJ. El código ejemplo cabe tanto a la cache de instrucciones como en la cache de uops, que inicialmente están vacías.

- i) Calcular la energía consumida por nuestro programa en un procesador sin cache de uops (las instrucciones se leen de la cache de instrucciones y se decodifican cada vez), en uno con cache de uops (las instrucciones se leen ya decodificadas de la cache de uops, la cache de datos y el decodificador solo se usan cuando es fallo en la cache de uops) y la ganancia en consumo de energía debida a la cache de uops durante la búsqueda y descodificación de instrucciones.

$$E_{\text{sin cache}} = (1 \times 10^{-9} \text{ J} + 1 \times 10^{-9} \text{ J}) \cdot 7 \times 10^6 = 14 \times 10^{-3} \text{ J}$$

$$E_{\text{con cache}} = (1 \times 10^{-9} \text{ J} + 10 \times 10^{-9} \text{ J}) \cdot 7 \times 10^6 = 77 \times 10^{-3} \text{ J}$$

$$G = 77 / 14 = 5,5$$