

Índice

1 - Evaluación de prestaciones	3
2 - Arquitectura del repertorio de instrucciones	10
3 - Sistema de memoria	13
4 - Computadores paralelos	18
5 - Ejercicios de exámenes	33

BOLETÍN DE EJERCICIOS DEL TEMA 2: EVALUACIÓN DE PRESTACIONES

ARQUITECTURA DE COMPUTADORES, 2º ING. INFORMÁTICA-INGENIERÍA DEL SOFTWARE
DEPT. DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES, UNIVERSIDAD DE SEVILLA

3. (Ejercicios 1.14.4, 1.14.5 y 1.14.6 de [PH-2011])

Una falacia habitual se produce al utilizar MIPS (millones de instrucciones por segundo) o MFLOPS (millones de instrucciones de punto flotante por segundo) para comparar las prestaciones de dos procesadores y considerar que el procesador con un MIPS o MFLOPS más elevado es el que tiene las mejores prestaciones.

Consideremos los programas de la tabla siguiente (donde L/S quiere decir “load/store” o carga y almacenamiento, y PF quiere decir “punto flotante”) en un procesador con frecuencia de reloj de 3 GHz.

	Nº instr.	Instr. L/S	Instr. PF	Instr. salto	CPI (L/S)	CPI (PF)	CPI (salto)
P1	10^6	50%	40%	10%	0.75	1	1.5
P2	3×10^6	40%	40%	20%	1.25	0.7	1.25

a) Calcule los MIPS de los programas.

$$[\text{Sol.: MIPS (P1)} = 3.247 \times 10^3 \quad \text{MIPS (P2)} = 2.913 \times 10^3]$$

b) Calcule los MFLOPS de los programas.

$$[\text{Sol.: MFLOPS (P1)} = 3.000 \times 10^3 \quad \text{MFLOPS (P2)} = 4.286 \times 10^3]$$

c) Calcule las prestaciones de los programas y compárelas con los valores de MIPS y MFLOPS.

[Sol.: Rendimiento (P1) = $1/t_{\text{CPU}}$ (P1) = $3.2 \times 10^3 \text{ s}^{-1}$ Rendimiento (P2) = $1/t_{\text{CPU}}$ (P2) = $9.7 \times 10^2 \text{ s}^{-1}$
P2 tiene la mayor cifra de MFLOPS, pero P1 tiene mayores prestaciones y la mayor cifra de MIPS]

4. (Ejercicios 1.8.1, 1.8.2 y 1.8.3 de [PH-2011])

Suponga que se han desarrollado varias versiones de un procesador con las siguientes características.

	Voltaje	Frecuencia de reloj
Versión 1	5 V	0.5 GHz
Versión 2	3.3 V	1 GHz

a) ¿Cuánto se ha reducido la carga capacitiva entre versiones si la potencia dinámica se ha reducido un 10%?

$$[\text{Sol.: } C_2 / C_1 = 1.03 \Rightarrow \text{Ha aumentado un 3\%}]$$

b) ¿Cuánto se ha reducido la potencia dinámica si la carga capacitiva no ha cambiado?

$$[\text{Sol.: } P_2 / P_1 = 0.87 \Rightarrow \text{Reducción del 13\%}]$$

c) Suponiendo que la carga capacitiva de la versión 2 es el 80% de la carga capacitiva de la versión 1, determine el voltaje de la versión 2 si su potencia dinámica es tal que se ha reducido un 40% respecto a la de la versión 1.

$$[\text{Sol.: } V_2 = 3.06 \text{ V}]$$

2.

a) Podemos calcular el número de ciclos totales de la siguiente manera:

$$n_{ciclos} = \sum_{i=0}^4 n_{inst_i} \cdot CPI_i = 500 \cdot 1 + 50 \cdot 5 + 100 \cdot 5 + 50 \cdot 2 = 1350 \text{ ciclos}$$

$$\text{Por tanto, como } t_{CPU} = \frac{n_{ciclos}}{\# \text{relaj}} = \frac{1350}{2 \cdot 10^9} = 0,000000675 = 6,75 \cdot 10^{-7} \text{ s}$$

b) Podemos calcularlo de la siguiente manera:

$$CPI = \frac{n_{ciclos}}{n_{inst}} = \frac{1350}{700} = 1,928 \text{ ciclos por instrucción}$$

c) Volvemos a calcular el número de ciclos totales:

$$n_{ciclos} = \sum_{i=0}^4 n_{inst_i} \cdot CPI_i = 500 \cdot 1 + 50 \cdot 5 + 50 \cdot 5 + 50 \cdot 2 = 1100 \text{ ciclos}$$

$$\text{Por tanto, } t_{CPU} = \frac{n_{ciclos}}{\# \text{relaj}} = \frac{1100}{2 \cdot 10^9} = 0,00000055 = 5,5 \cdot 10^{-7} \text{ s}$$

La aceleración se calcula respecto al otro programa, de la siguiente manera:

$$\text{Aceleración} = \frac{t_{CPU} \text{ más lento}}{t_{CPU} \text{ más rápido}} = \frac{6,75 \cdot 10^{-7}}{5,5 \cdot 10^{-7}} = 1,227$$

El nuevo CPI del programa será:

$$CPI = \frac{n_{ciclos}}{n_{inst}} = \frac{1100}{700 - 50} = 1,69 \text{ ciclos por instrucción.}$$

$$\text{Luego, MFLOPS} = \frac{n_{\text{instr}}}{t_{\text{cpu}} \cdot 10^6} = \frac{3 \cdot 10^6 \cdot 0,4}{2,8 \cdot 10^{-4} \cdot 10^6} = 4285,71 \text{ millones de op en pf/s}$$

c) Podemos calcular las prestaciones de la siguiente manera: $\text{Prest} = \frac{1}{T_{\text{CPU}}}$

Para P1:

$$\text{Prest} = \frac{1}{3,083 \cdot 10^{-4}} = 3243,5939 \text{ programas/s}$$

Para P2:

$$\text{Prest} = \frac{1}{1,03 \cdot 10^{-3}} = 970,8737 \text{ programas/s}$$

Observamos que $\text{MIPS}_{P1} > \text{MIPS}_{P2}$, de igual manera $\text{MFLOPS}_{P1} < \text{MFLOPS}_{P2}$,

es decir, P2 tiene mayor número de millones de operaciones de punto flotante por segundo,
pero sin embargo P1 tiene mayor número de millones de instrucciones por
segundo y también mayores prestaciones.

BOLETÍN DE EJERCICIOS DEL TEMA 3: ARQUITECTURA DEL REPERTORIO DE INSTRUCCIONES

ARQUITECTURA DE COMPUTADORES, 2º ING. INFORMÁTICA-INTENIERÍA DEL SOFTWARE

DPTO. DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES, UNIVERSIDAD DE SEVILLA

Ejemplo de preguntas tipo test:

1) En las arquitecturas registro-registro:

- a) ninguna instrucción tiene operandos en memoria
- b) ninguna operación ALU tienen operandos en memoria**
- c) el código es más compacto que en las arquitecturas memoria-memoria
- d) todas las operaciones ALU se realizan mediante la pila

2) En el modo de direccionamiento indirecto con memoria:

- a) la dirección efectiva del operando se encuentra almacenada en memoria en la dirección especificada en la instrucción**
- b) el operando se encuentra en la instrucción
- c) la dirección efectiva del operando se encuentra en la instrucción
- d) la dirección efectiva del operando se encuentra en un registro

3) En el modo de direccionamiento indirecto con registro:

- a) la dirección efectiva del operando se encuentra almacenada en memoria en la dirección especificada en la instrucción
- b) la dirección efectiva del operando se encuentra en un registro**
- c) el operando se encuentra en la instrucción
- d) la dirección efectiva del operando se encuentra en la instrucción

4) Los procesadores RISC están basados en arquitecturas:

- a) memoria-memoria
- b) registro-registro**
- c) registro-memoria
- d) en cualquiera siempre y cuando el repertorio de instrucciones sea reducido

5) En un procesador en el que el tamaño de instrucción coincide con el tamaño de palabra no puede existir el modo de direccionamiento:

- a) indirecto con registro
- b) directo a registro
- c) directo a memoria o absoluto**
- d) indirecto con desplazamiento

6) Si un procesador incluye una instrucción que suma el contenido de dos registros y almacena el resultado en memoria, entonces:

- a) tiene una arquitectura reg-reg
- b) tiene una arquitectura reg-mem o mem-mem**
- c) tiene una arquitectura reg-mem
- d) tiene una arquitectura propia de los RISC

¿Tamaño de registros? Tamaño de la palabra, 8 bits.

- b) En este caso codificamos el direccionamiento en el propio *opcode*. De las 6 posibles combinaciones de modos de direccionamiento que permite el formato anterior, ahora no están permitidas operaciones con dos operandos en memoria. Tenemos, por lo tanto, 5 combinaciones de modos de direccionamiento diferentes.

Suponiendo que todas las operaciones que realiza la CPU permiten 5 combinaciones posibles de modos de direccionamiento. Tendríamos que codificar $30 \text{ operaciones} \times 5 = 150 \text{ opcodes}$. Por lo tanto necesitamos 8 bits para el opcode.

Al dedicar 8 bits para el *opcode*, tenemos 256 *opcodes* diferentes. Si cada operación admite 5 combinaciones posibles de modos de direccionamiento. Podríamos codificar hasta $256/5 = 51$ operaciones diferentes.

Op_code	Operando A	Operando B
8	4	4

- c) Se pueden añadir dos instrucciones que permitan transferir un valor inmediato de 4 bits a la parte alta o a la parte baja del destino. Por ejemplo: MOVH para transferir un valor a la parte alta del destino y MOVL para transferirlo a la parte baja. Para transferir el valor 254 (0xFE) al registro R1 realizaríamos la siguiente secuencia:
 MOVH R1, 0xF
 MOVL R1, 0xE
- d) ¿Cuántos registros tiene el procesador? $2^4 = 16$ registros.

Sea un sistema con un procesador cuya capacidad de direccionamiento es de 4 GBytes, que utiliza memoria virtual paginada con un tamaño de página de 4 KBytes y que puede acceder a una memoria principal de hasta 256 MBytes.

El sistema dispone de un sistema de memoria con caches separados. La cache de datos tiene 4 líneas y es de correspondencia directa. El tamaño de un bloque es de 16 bytes. Usa la política de escritura C B -WA.

Realice las siguientes tareas:

- a) Dibuje la estructura del sistema, indicando los diferentes campos en que se divide cada dirección (dirección virtual y dirección física); así como los diferentes elementos necesarios para realizar el mecanismo de traducción. Indique el tamaño de todos los campos.
- b) Dibuje la estructura de la caché de datos, teniendo en cuenta la configuración de la caché dada en el enunciado. Especifique también los campos en los que se divide la dirección física para esta configuración de caché. Indique el tamaño de todos los campos.
- c) Se quiere ejecutar el siguiente programa escrito en C, en el que se inicializa un vector de caracteres con las 26 letras del alfabeto en mayúsculas.

```
int i;
char letras[26];
i=0;

while (i<26)
{
    letras[i]='A'+i;
    i++;
}
```

Realice una traza de ejecución y represente el estado final de la caché, tras ejecutar el código, teniendo en cuenta solo los accesos a datos, suponiendo que la tabla de páginas y la caché inicialmente está vacía. Considere que el vector letras tiene asociado la dirección virtual 0x00001100 y que los marcos de páginas físicas se asignan a partir del marco 256 de memoria principal. La variable i se almacena en un registro interno del procesador. Tenga en cuenta que el tipo de dato *char* ocupa 1 byte en memoria.

Para realizar la traza ayúdese de una tabla con la siguiente información:

Dato accedido	Dirección física	Etiqueta	Línea en caché	Fallo y tipo / Acierto

- d) Calcule la frecuencia de fallos de accesos a caché para dicha traza de accesos.
- e) Explique qué ocurriría si la política de escritura fuera WT-NWA.

b) Sabiendo que tenemos 28 bits de direcciones físicas y que ésta se divide en etiqueta, índice y desplazamiento de bloque;

$$\text{Desplazamiento de bloque} = \log_2(16) = 4 \text{ bits}$$

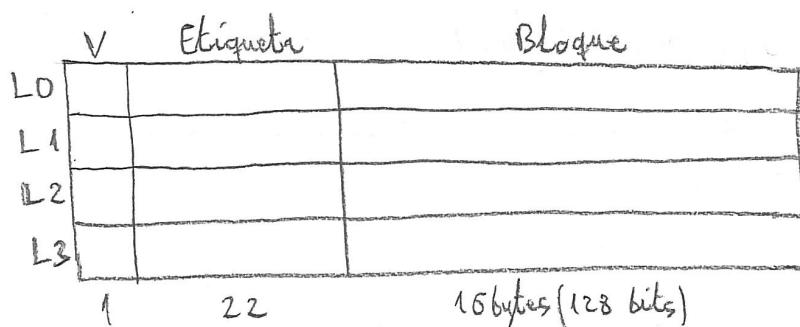
$$\text{Índice} = \log_2(4) = 2 \text{ bits}$$

$$\text{Etiqueta} = \text{Dir. Físico} - (\text{Desplazamiento de bloque} + \text{Índice}) = 28 - (4 + 2) = 22 \text{ bits.}$$

Por tanto, la dirección física queda dividida de la siguiente manera:

DF	Etiqueta	Índice	Desp de Bloq.	28
22	2	4		

La estructura de la caché será la siguiente:



c) 0x00001100 corresponde al marco o número de página física 256 \rightarrow 0x0100 luego la dirección física se concatena con el desplazamiento de página \rightarrow 0x01001100

Dato accedido	Dirección física	Etiqueta	Línea en caché	Fallo / Acierto
Letras [0]	0x01001100	0x4004	0	F. Forzoso
Letras [1]	0x01001101	0x4004	0	Acierto
-----	-----	---	---	-----
Letras [15]	0x0100110F	0x4004	0	Acierto
Letras [16]	0x01001110	0x4004	1	F. Forzoso
Letras [17]	0x01001111	0x4004	1	Acierto
-----	-----	---	---	-----
Letras [25]	0x0100111F	0x4004	1	Acierto

1. Indique qué bucle es paralelizable usando Open MP.

- | | |
|----|--|
| A. | <pre>for (i=0; i<10; i++) { a[i] = a[i] * 3; res += a[i]; }</pre> |
| B. | <pre>for (i=0; i<10; i++) { a[i] = a[i-1] * 3; }</pre> <p><i>El bucle del apartado B no puede ser paralelizable debido a que se puede estar produciendo una condición de carrera al acceder al elemento "i-1" del vector a.</i></p> |

- C. Ninguno.
D. Los dos son paralelizables.

2. Al ejecutar el siguiente código, el valor final de la variable res es incorrecto. ¿Qué directiva de OpenMP debemos utilizar para solucionar el problema?

```
#pragma parallel for private(aux)
for (i=0; i<100; i++) {
    aux = funcion(x[i]);
    res = sqrt(res + aux);
}
```

- A. reduction
B. atomic
C. critical
D. Ninguna

reduction o atomic no pueden ser, debido a que no aceptan la operación de raíz cuadrada (sqrt)

3. Indique qué afirmación es correcta:

- A. Un comunicador es cualquier función MPI que se utiliza para la comunicación entre los procesos.
B. Un comunicador define o identifica a un grupo de procesos.
C. Un comunicador es el parámetro que se utiliza para identificar al proceso al que se envía el mensaje o al proceso del que se recibe el mensaje en las funciones MPI Send o MPI Recv, respectivamente.
D. Ninguna es correcta.

4. En un DLX con todos los desvíos, ¿cuantos ciclos de bloqueo se producen al ejecutar la siguiente secuencia de instrucciones?

```
lw r2, 0(r1)
addi r2, r2, 4
sw 0(r2), r1
j etiqueta
```

- A. 1 ciclo
B. 2 ciclos
C. 3 ciclos
D. 4 ciclos

lw r2, 0(r1)	IF	ID	EX	MEM	WB			
addi r2, r2, 4		IF	ID	-	EX	MEM	WB	
sw 0(r2), r1			IF	-	ID	EX	MEM	WB
j etiqueta				-	IF	ID	EX	MEM

Hay 1 bloqueo de datos. Además, se produce un bloqueo de control debido al salto de la instrucción "j". Por tanto hay 2 ciclos de bloqueos.

5. ¿Cuántos ciclos se pueden eliminar reordenando el código de la pregunta anterior?

- A. El código no se puede reordenar.
B. El código se puede reordenar, pero no se eliminan bloqueos.
C. 1 ciclo.
D. 2 ciclos.

lw r2, 0(r1)	IF	ID	EX	MEM	WB			
sw 4(r2), r1		IF	ID	-	EX	MEM	WB	
addi r2, r2, 4			IF	-	ID	EX	MEM	WB
j etiqueta				-	IF	ID	EX	MEM

Aunque se ha podido reordenar el código, el número de bloqueos sigue siendo el mismo.

TRAZA de ejecución:

Para hacer la traza de ejecución, hay que tener en cuenta que en el código hay 2 instrucciones de salto (una correspondiente al bucle for y otra correspondiente a la estructura condicional). En cada iteración del bucle debemos analizar estas 2 instrucciones de salto para saber si el salto es o no tomado. El enunciado nos dice que vamos a trabajar con una secuencia de número enteros (positivos y negativos). Del código se deduce que en la primera iteración se accede a la posición de memoria 1020 (=996+24), en la que se encuentra el valor negativo -100, y en las 5 siguientes iteraciones se accederán a los valores 400, -10, 1241, -15, 300 (en este orden decreciente).

En el caso de ser un número negativo (-100, -10, -15), la condición de la estructura condicional es verdadera, por lo tanto el salto NO SE TOMA; y en el caso de ser un número positivo (400, 1241, 300), la instrucción de salto correspondiente a la estructura condicional SI SE TOMA.

A continuación se muestra la traza de ejecución para la primera iteración, que se trabaja con un número negativo. Para las otras iteraciones correspondientes a números negativos, sería equivalente, excepto que las dos primeras instrucciones no se ejecutarían (no pertenecen al bucle for).

1ª iteración: Nº NEGATIVO (*) SALTO NO TOMADO; (**) SALTO TOMADO																
add \$t4,\$t0,\$t0	IF	ID	EX	MEM	WB											
addi \$t2,\$t2,24		IF	ID	EX	MEM											
lw \$t1,996(\$t2)		IF	ID	EX	MEM	WB										
sgei \$t3,\$t1,0		IF	ID	-	EX	MEM	WB									
bnez \$t3,POS (*)			IF	-	-	ID	EX	MEM	WB							
sub \$t1,\$t0,\$t1						IF	ID	EX	MEM	WB						
addi \$t4,\$t4,1							IF	ID	EX	MEM	WB					
sw \$t1, 996(\$t2)								IF	ID	EX	MEM	WB				
subi \$t2,\$t2,4									IF	ID	EX	MEM	WB			
bnez \$t2,SIG(**)									IF	-	ID	EX	MEM	WB		
sw \$t4,2000(\$t0)										IF _{sw}				"aborted"		

A continuación se muestra la traza de ejecución para la segunda iteración, que se trabaja con un número positivo. Para las otras iteraciones correspondientes a números positivos, sería equivalente, excepto para la última iteración en la que la última instrucción (la que está fuera del bucle) si se ejecutaría, es decir no se aborta en la etapa IF.

2ª iteración: Nº POSITIVO (*) SALTO TOMADO; (**) SALTO TOMADO																
lw \$t1,996(\$t2)			IF	ID	EX	MEM	WB									
sgei \$t3,\$t1,0			IF	ID	-	EX	MEM	WB								
bnez \$t3,POS (*)			IF	-	-	ID	EX	MEM	WB							
sub \$t1,\$t0,\$t1							IF _{sub}	"aborted"								
subi \$t2,\$t2,4								IF	ID	EX	MEM	WB				
bnez \$t2,SIG(**)									IF	-	ID	EX	MEM	WB		
sw \$t4,2000(\$t0)										IF _{sw}				"aborted"		

EJERCICIO 2. Considere un procesador MIPS que ejecuta el siguiente trozo de código:

```
addi $t2,$t0,20
addi $t3,$t0,$t0
etiq1:lw $t1,0($t2)
beqz $t1,etiq2
add $t3,$t3,1
etiq2:subi $t2,$t2,4
bnez $t2,etiq1
sw $t3, 1000($t0)
```

Considere que el procesador MIPS es un procesador segmentado que usa la técnica de anticipación de resultados o bypass y predicción de salto no tomado. Inicialmente la memoria contiene a partir de la dirección 4 los siguientes valores de 32 bits: 1,0,3,4,0, ...

Se pide :

- a) Escriba un trozo de código usando un lenguaje de alto nivel (p.e. Lenguaje C) que realice la misma acción que el trozo de código escrito en el enunciado.

```
cuenta = 0 ;
for(i=5 ; i>0 ; i--)
    if (x[i] != 0)
        cuenta ++ ;
```

TRAZA de ejecución:

Para hacer la traza de ejecución, hay que tener en cuenta que en el código hay 2 instrucciones de salto (una correspondiente al bucle for y otra correspondiente a la estructura condicional). En cada iteración del bucle debemos analizar estas 2 instrucciones de salto para saber si el salto es o no tomado. El enunciado nos dice que vamos a trabajar con una secuencia de números enteros positivos, algunos de ellos de valor 0. Del código se deduce que en la primera iteración se accede a la posición de memoria 20, en la que se encuentra el valor 0, y en las 4 siguientes iteraciones se accederán a los valores 4,3,0,1 (en este orden decreciente).

En el caso de ser un número con valor distinto de 0, la condición de la estructura condicional es verdadera, por lo tanto el salto NO SE TOMA; y en el caso de ser un número 0, la instrucción de salto correspondiente a la estructura condicional SI SE TOMA.

- c) Indique el número total de instrucciones que se ejecutan.

Instrucciones que se ejecutan una vez, porque están fuera del bucle, ya sea al principio o al final = 2 + 1 = 3

Instrucciones que se ejecutan en cada iteración: Hay que distinguir entre el número de instrucciones que se ejecutan en el caso de ser un 0 (4 instrucciones) o distinto de 0 (5 instrucciones) = $4*2 + 5*3 = 23$

Nº de instrucciones = 3 + 23 = 26 instrucciones

- d) Calcule el número medio de ciclos por instrucción (CPI) empleados en la ejecución completa del programa dado.

Para calcular el CPI, debemos calcular el número de ciclos de bloqueo de datos y de control:

- Todas las iteraciones tienen el mismo número de ciclos de bloqueos de datos = 3 ciclos x 5 iteraciones
- En cuanto a los ciclos de bloqueo de control, hay 6 ciclos de bloqueo de control:
 - o el salto correspondiente al bucle for (instrucción 7) se toma 4 veces, provocando 4 ciclos de bloqueo de control
 - o el salto correspondiente al if (instrucción 4) se toma cuando el número es 0, en 2 ocasiones, provocando 2 ciclos de bloqueo de control

$$\text{CPI} = (4 + 26 + 3 \times 5 + 6) / 26 = 1,961$$

EJERCICIO 3. Considere un procesador MIPS que ejecuta el siguiente trozo de código:

```
addi $t2,$t0,100
INC: lw $t1,0($t2)
      addi $t1,$t1,1
      sw $t1, 0($t2)
      addi $t2,$t2,4
      seqi $t3,$t2,140
      beqz $t3,INC
      sw $t0, 0($t2)
```

Se pide :

- a) Escriba un trozo de código usando un lenguaje de alto nivel (p.e. Lenguaje C) que realice la misma acción que el trozo de código escrito en el enunciado.

```
for(i=0 ; i<10 ; i++)
    x[i]++ ;
```

TRAZA de ejecución:

Para hacer la traza de ejecución, hay que tener en cuenta que en el código hay 1 instrucción de salto (correspondiente al bucle for). Este salto ES TOMADO en todas las iteraciones del bucle, excepto en la última.

- c) Indique el número total de instrucciones que se ejecutan.

Instrucciones que se ejecutan una vez, porque están fuera del bucle, ya sea al principio o al final = 1 + 1 = 2

Instrucciones que se ejecutan en cada iteración: 6 instrucciones x 10 iteraciones

Nº de instrucciones = 2 + 60 = 62 instrucciones

- d) Calcule el número medio de ciclos por instrucción (CPI) empleados en la ejecución completa del programa dado.

Para calcular el CPI, debemos calcular el número de ciclos de bloqueo de datos y de control:

- Todas las iteraciones tienen el mismo número de ciclos de bloqueos de datos = 2 ciclos x 10 iteraciones
- En cuanto a los ciclos de bloqueo de control, hay 9 ciclos de bloqueo de control (10 iteraciones -1):

$$\text{CPI} = (4 + 62 + 2 \times 10 + 9) / 62 = 1,532$$

EJERCICIO 4. Considere un procesador MIPS que ejecuta el siguiente trozo de código:

```
1      addu $t0, $t0, $zero
2      etq:   lw $t1, 1000($t0)
3      lw $t2, 1040($t0)
4      addu $t3, $t1, $t2
5      sw $t3, 1000($t0)
6      addiu $t0, $t0, 4
7      seqi $t4, $t0, 40
8      beq $t4, $zero, etq
```

Se pide :

- a) Escriba un trozo de código usando un lenguaje de alto nivel (p.e. Lenguaje C) que realice la misma acción que el trozo de código escrito en el enunciado.

```
for(i=0 ; i<10 ; i++)
    x[i] = x[i] + y[i] ;
```

TODAS	3	MEM	4	EX
TODAS	4	MEM	5	MEM
TODAS	6	EX	7	EX
TODAS	7	EX	8	ID

- c) Indique el número total de instrucciones que se ejecutan.

Instrucciones que se ejecutan una vez, porque están fuera del bucle, ya sea al principio o al final = 1

Instrucciones que se ejecutan en cada iteración: 7 instrucciones x 10 iteraciones

$$\text{Nº de instrucciones} = 1 + 70 = 71 \text{ instrucciones}$$

- d) Calcule el número medio de ciclos por instrucción (CPI) empleados en la ejecución completa del programa dado.

Para calcular el CPI, debemos calcular el número de ciclos de bloqueo de datos y de control:

- Todas las iteraciones tienen el mismo número de ciclos de bloqueos de datos = 2 ciclos x 10 iteraciones
- En cuanto a los ciclos de bloqueo de control, hay 9 ciclos de bloqueo de control (10 iteraciones -1):

$$\text{CPI} = (4 + 71 + 2 \times 10 + 9) / 71 = 1,464$$

- e) Reordene las instrucciones del programa dado para reducir los ciclos de bloqueos de datos y calcule cuánto se mejora el CPI del programa respecto al CPI calculado en el apartado d); para ello calcule la aceleración.

Antes de reordenar el código observamos en la traza de ejecución cuando se producen ciclos de bloqueos de datos, con el fin de insertar alguna instrucción entre las instrucciones que tienen dependencia de datos y que provocan esos ciclos de bloqueos de datos. Los ciclos de bloqueos de datos se producen entre las instrucciones 3-4 y 7-8.

Una posible solución es adelantar la instrucción 6 para que se ejecute entre las instrucciones 3 y 4. Para que la ejecución siga siendo igual, será necesario modificar el valor de la dirección de la instrucción de almacenamiento (instrucción 5), porque al adelantar incrementamos el valor de \$t0 en 4, antes de que se realice el almacenamiento (*****); por tanto en lugar de 1000, será 1000-4 = 996. Con esto conseguimos reducir 1 ciclo de bloqueo

Y si además, retrasamos la ejecución de la instrucción 5 de almacenamiento, para que se ejecute entre las instrucciones 7 y 8, conseguimos reducir el 2º ciclo de bloqueo de datos.

```

1      addu $t0, $t0, $zero
2  etq:  lw $t1, 1000($t0)
3      lw $t2, 1040($t0)
6      addiu $t0, $t0, 4 (adelantada)
4      addu $t3, $t1, $t2
7      seqi $t4, $t0, 40
5      sw $t3, 996($t0) (cambio la constante*****)
8      beq $t4, $zero, etq

```

BOLETÍN DE EJERCICIOS DE LA SEGUNDA PARTE DEL TEMA 6: COMPUTADORES PARALELOS

ARQUITECTURA DE COMPUTADORES, 2º ING. INFORMÁTICA-INGENIERÍA DEL SOFTWARE

DEPT. DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES, UNIVERSIDAD DE SEVILLA

Para simplificar el análisis, en los tres primeros problemas suponga que el programa opera únicamente en dos modos:

1. Modo paralelo, usando completamente todos los procesadores.
2. Modo secuencial, usando únicamente un procesador.

1. El estudio de la ejecución de un programa secuencial revela que el 95 por ciento del tiempo de ejecución se invierte dentro de funciones que son paralelizables. ¿Cuál es la aceleración máxima que podríamos esperar al ejecutar una versión paralela de este programa en 10 procesadores?

[Sol.: A = 6.90]

2. Para un tamaño dado del problema, el 6 por ciento de las operaciones de un programa paralelo están dentro de funciones de E/S que deben ejecutarse necesariamente en un solo procesador. ¿Cuál es el número mínimo de procesadores necesarios para que el programa paralelo muestre una aceleración de 10?

[Sol.: Al menos 24 procesadores.]

3. Suponga que quiere obtener una aceleración de 80 en un computador paralelo con 100 procesadores. Calcule cuál es la fracción máxima del programa que puede ser secuencial para ello.

[Sol.: $f_{\max}(\text{secuencial}) = 0.25 \%$]

4. Suponga que tiene una aplicación corriendo sobre un multiprocesador de memoria distribuida con 32 procesadores, con una frecuencia de reloj de 3.3 GHz, para el cual una referencia a una zona de memoria no local (que reside en un procesador remoto y por tanto incluye comunicación entre distintos procesadores) tarda 200 ns en completarse.

Para simplificar, suponga que para esta aplicación todas las referencias a una zona local de memoria provocan un acierto en la jerarquía local de memoria (aunque esta suposición es ligeramente optimista).

Si el CPI base para cada procesador (suponiendo que todas las referencias a memoria aciertan en caché) es de 0.5, calcular cuánto más rápido es el multiprocesador en caso de que no haya comunicación (todas las instrucciones provocan referencias a zonas de memoria local) frente al caso en que un 0.2% de las instrucciones provocan una referencia a una zona de memoria no local (en la que hay comunicación).

[Sol.: 3.64 veces más rápido]

4.

Si el CPI base es 0,5 para el caso donde todas las instrucciones provocan referencias a zonas de memoria Local, entonces, para el caso del 0,2% de las instrucciones que provocan referencias a zonas de memoria no local:

$$\begin{aligned} \text{CPI}_{nl} &= \text{CPI}_{base} + 0,002 \cdot f_{nl} \cdot t_{cyclo_{nl}} = 0,5 + 0,002 \cdot 3,3 \cdot 10^9 \cdot 200 \cdot 10^{-9} = \\ &= 1,82 \text{ ciclos por instrucción} \end{aligned}$$

Por tanto la aceleración será:

$$a = \frac{T_{local}}{T_{local}} = \frac{\text{CPI}_{nl} \cdot n_{inst} \cdot t_{cyclo}}{\text{CPI}_{ll} \cdot n_{inst} \cdot t_{cyclo}} = \frac{1,82}{0,5} = \boxed{3,64 \text{ veces más rápido}}$$

Apellidos y nombre:

Cuestión 3 (1,5 puntos). Indique qué opción es la correcta en el margen izquierdo de cada pregunta. Sólo se puede seleccionar una opción. Cada respuesta incorrecta resta un tercio de una correcta.

- 1) Las arquitecturas basadas en registros de propósito general:
 - a) no permiten instrucciones ALU con operandos en memoria
 - b) incluyen las arquitecturas Registro-Registro, Registro-Memoria y Memoria-Memoria
 - c) no permiten instrucciones para manipular una pila
 - d) incluyen sólo las arquitecturas Registro-Registro y Registro-Memoria
- 2) En el modo de direccionamiento absoluto:
 - a) el operando se encuentra en la instrucción
 - b) la dirección efectiva del operando se encuentra en un registro
 - c) la dirección efectiva del operando se encuentra en la instrucción
 - d) la dirección efectiva del operando se encuentra almacenada en memoria en la dirección especificada en la instrucción
- 3) En las cachés de mapeado directo:
 - a) no existen fallos de capacidad
 - b) no existen fallos de conflicto
 - c) el índice representa el conjunto en el que debe encontrarse el bloque de memoria
 - d) no se requieren políticas de sustitución de bloque
- 4) Para direccionar un dispositivo de entrada/salida:
 - a) el procesador debe utilizar alguna instrucción especial
 - b) el procesador debe realizar el acceso como si se tratase de un acceso a memoria
 - c) el procesador genera direcciones físicas de forma excepcional
 - d) existen dos métodos distintos que pueden coexistir en el mismo computador
- 5) En el procesador MIPS estudiado:
 - a) el número de ciclos de bloqueo debidos a un riesgo de datos es a lo sumo 2
 - b) el número de ciclos de bloqueo es siempre 0 si se usa anticipación de resultados
 - c) el número de bloqueos de control es 1 si el salto es no tomado
 - d) se apuesta por salto tomado
- 6) Según la taxonomía de Flynn, las GPU son:
 - a) SISD
 - b) SIMD
 - c) MISD
 - d) MIMD
- 7) ¿Cuál es mínimo número de procesadores que se requiere para conseguir una aceleración de al menos 2,5 al parallelizar un programa en el que la parte secuencial del programa supone un 20%?
 - a) 2
 - b) 8
 - c) 4
 - d) 10

$$A = \frac{1}{1 - f + \frac{f}{p}} \rightarrow 2,5 \geq \frac{1}{0,2 + \frac{1 - 0,2}{p}} \rightarrow p \geq 4$$

Examen oficial, convocatoria de junio de 2015: Problema 1:

a) Las direcciones físicas usa 32 bits de direcciones.

A partir del tamaño de la línea, podemos obtener los bits de direcciones que se usan para el desplazamiento del bloque = 16 bytes $\rightarrow \log_2(16) = 4$ bits de desplazamiento.

Podemos obtener el número de conjuntos en total que tiene la caché de la siguiente manera:

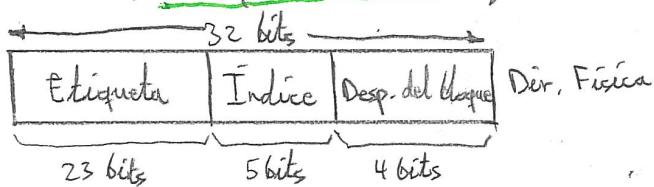
$$N^{\circ} \text{ conjuntos} = \frac{\text{Tam de la caché}}{\text{Tam del bloque} \cdot N^{\circ} \text{ vías}} = \frac{1 \text{ KB}}{16 \cdot 2} = \frac{2^{10}}{16 \cdot 2} = 32 \text{ conjuntos en total}$$

Por tanto el número de bits necesarios para el índice será:

$$\text{Índice} = \log_2(N^{\circ} \text{ conjuntos}) = \log_2(32) = 5 \text{ bits de direcciones}$$

Entonces la etiqueta tendrá: $32 - 4 - 5 = 23$ bits de direcciones.

Por tanto, el esquema sería el siguiente:



b)

Dir. de Mem	Etiqueta	Conjunto	Línea	Acíto/Fallo	Bloque accedido
0x10010000	0x100100	0	0	Fallo Forzoso	a [0]
0x10020000	0x100200	0	1	Fallo Forzoso	b [0]
0x10030000	0x100300	0	0	Fallo Forzoso	c [0]
0x10010004	0x100100	0	1	Fallo conflicto	a [1]
0x10020004	0x100200	0	0	Fallo conflicto	b [1]
0x10030004	0x100300	0	1	Fallo conflicto	c [1]
---	---	---	---	---	---
0x10010010	0x100100	1	0	Fallo Forzoso	a [4]
0x10020010	0x100200	1	1	Fallo Forzoso	b [4]
0x10030010	0x100300	1	0	Fallo Forzoso	c [4]
0x10010014	0x100100	1	1	Fallo conflicto	a [5]
---	---	---	---	---	---
0x1003001C	0x100300	1	1	Fallo conflicto	c [7]

Se producen 6 fallos
Forzoso y $(3 \cdot 3) + (3 \cdot 3) = 18$
Fallos por conflictos.
No hay fallos por capacidad

Problema de caché, prueba de evaluación continua, 2013.

Un computador tiene una unidad de memoria principal de 1 KB y una frecuencia de reloj de 2,0 GHz. La memoria caché, de un único nivel, es 4-asociativa (asociativa por conjuntos), tiene una capacidad total de 128 B (contando sólo los bloques de datos), un tamaño de línea de 16 B y utiliza un algoritmo de reemplazo LRU ("Least Recently Used").

Se pide:

a) Dibuje un esquema indicando en cuántos campos debe particionarse una dirección de memoria para calcular su ubicación en la caché, cómo se llama cada uno y cuántos bits tiene cada campo, incluyendo el cálculo necesario para calcular dicho número de bits.

b) Suponemos que inicialmente la memoria caché está vacía y que en la ejecución de un programa se leen sucesivamente las direcciones de memoria principal siguientes, expresadas en hexadecimal:

155, 015, 0F8, 051, 150, 380, 393, 01F.

Construya una tabla en la que indique, para cada uno de los accesos a memoria:

- El valor que toman los diferentes campos en que deba dividirse una dirección de memoria en función de la organización de memoria caché.
- Si la lectura de cada dirección de memoria produce un acierto o un fallo de caché. En caso de que sea un fallo, indique de qué tipo es dicho fallo.
- El contenido completo de los datos contenidos en cada línea de la caché después de leer cada una de estas direcciones de memoria. Para ello identifique cada bloque de memoria principal con la notación $M[X]$ donde X es la dirección de bloque (expresada en binario). No es necesario que indique los contenidos de la memoria de etiquetas ni de los bits de información (validez, permisos, política de reemplazo, etc).

c) Calcule la tasa de fallos producida en la ejecución del programa.

4

$$\text{Tasa de fallos} = \frac{\text{Nº fallos}}{\text{Nº accesos}} = \frac{7}{8} = 0,875$$

Es decir, ha habido un 87,5% de fallos respecto al número de accesos en total.

Prueba 1 de evaluación continua, 2013/14. Grupo 1: Problema de memoria virtual y caché:

a)

La memoria virtual es de $4\text{ GB} = 2^2 \cdot 2^{30} = 2^{32}$ bytes, por tanto, las direcciones virtuales son 32 bits.

La memoria principal es de $256\text{ MB} = 2^8 \cdot 2^{20} = 2^{28}$ bytes, por tanto, las direcciones físicas son de 28 bits.

Podemos calcular cuántos bits de desplazamiento de página tenemos tanto para la dirección física como la virtual:

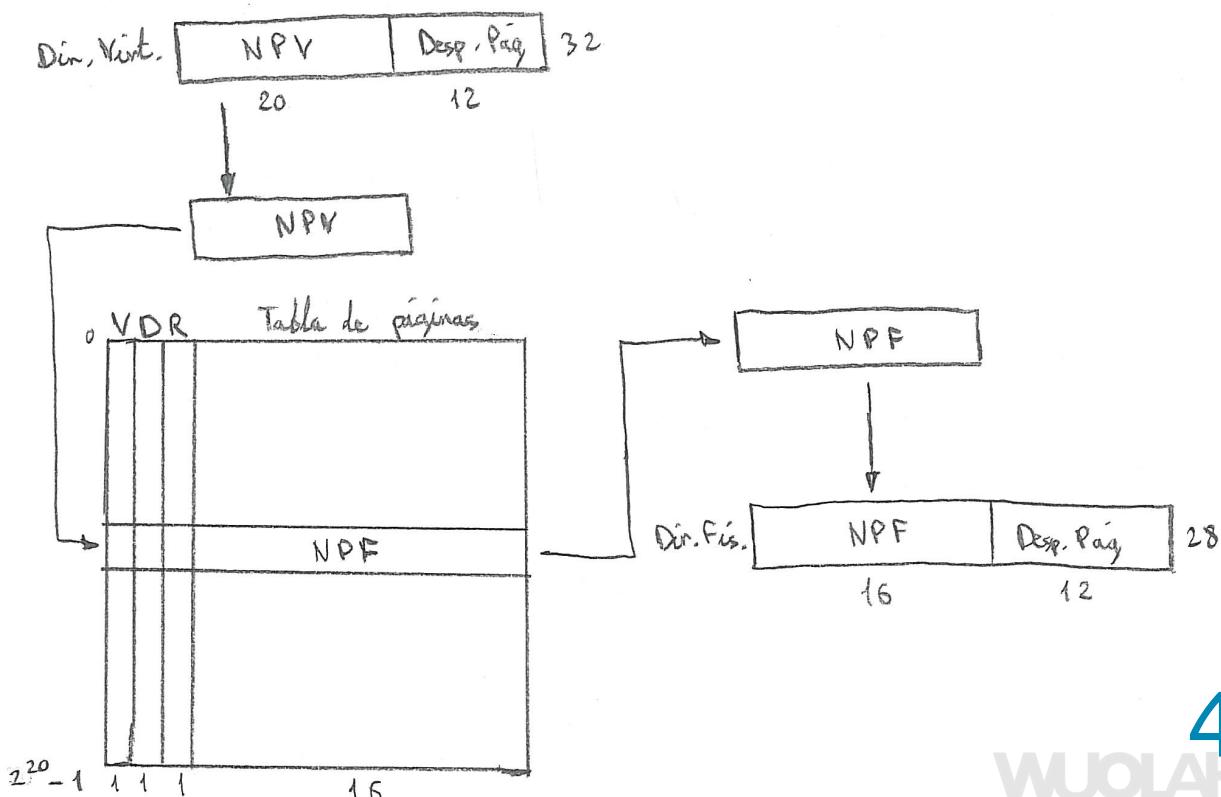
Desp. de pag = $\log_2(4\text{ KB}) = \log_2(2^2 \cdot 2^{10}) = 12$ bits de desplazamiento de página.

Sabiendo esto podemos calcular el número de página virtual y físico:

$$\text{Nº de pag. virtual} = 32 - 12 = 20 \text{ bits}$$

$$\text{Nº de pag. físico} = 28 - 12 = 16 \text{ bits}$$

Por tanto, la estructura del sistema será la siguiente:



El estado final de la caché será el mismo que el inicial, ya que en ningún momento se ha llevado un dato a la caché. Es decir, la caché estará vacía.

d)

$$\text{Frecuencia de fallos} = \frac{\text{Nº de fallos}}{\text{Nº de accesos}} = \frac{26}{22} = 1$$

Es decir, el 100% de los accesos son fallos.

e)

Si se cambia la política de escritura de la memoria caché a CB-WA, entonces en caso de acierto se llevará el dato sólo a memoria caché, y en caso de fallo, se llevará a memoria caché y memoria principal.

Entonces, el miss rate va a disminuir, ya que de letras [1] a letras [15] y de letras [17] a letras [25] se produciría esta vez un acierto.

Es decir, el miss rate sería:

$$\text{Frecuencia de fallos} = \frac{\text{Nº de fallos}}{\text{Nº de accesos}} = \frac{2}{26} = 0,076$$

Habrá un 7,6% de fallos respecto al total de accesos.

Prueba 1 de evaluación continua, 2013/14. Grupo 2: Problema de memoria virtual y caché.

a)

La memoria virtual es de $4\text{GB} = 2^2 \cdot 2^{30} = 2^{32}$ bytes, por tanto, las direcciones virtuales son 32 bits.

La memoria principal es de $1\text{GB} = 2^{30}$ bytes, por tanto, las direcciones físicas son de 30 bits.

Para calcular cuántos bits de desplazamiento de página tenemos tanto para la dirección física como la virtual:

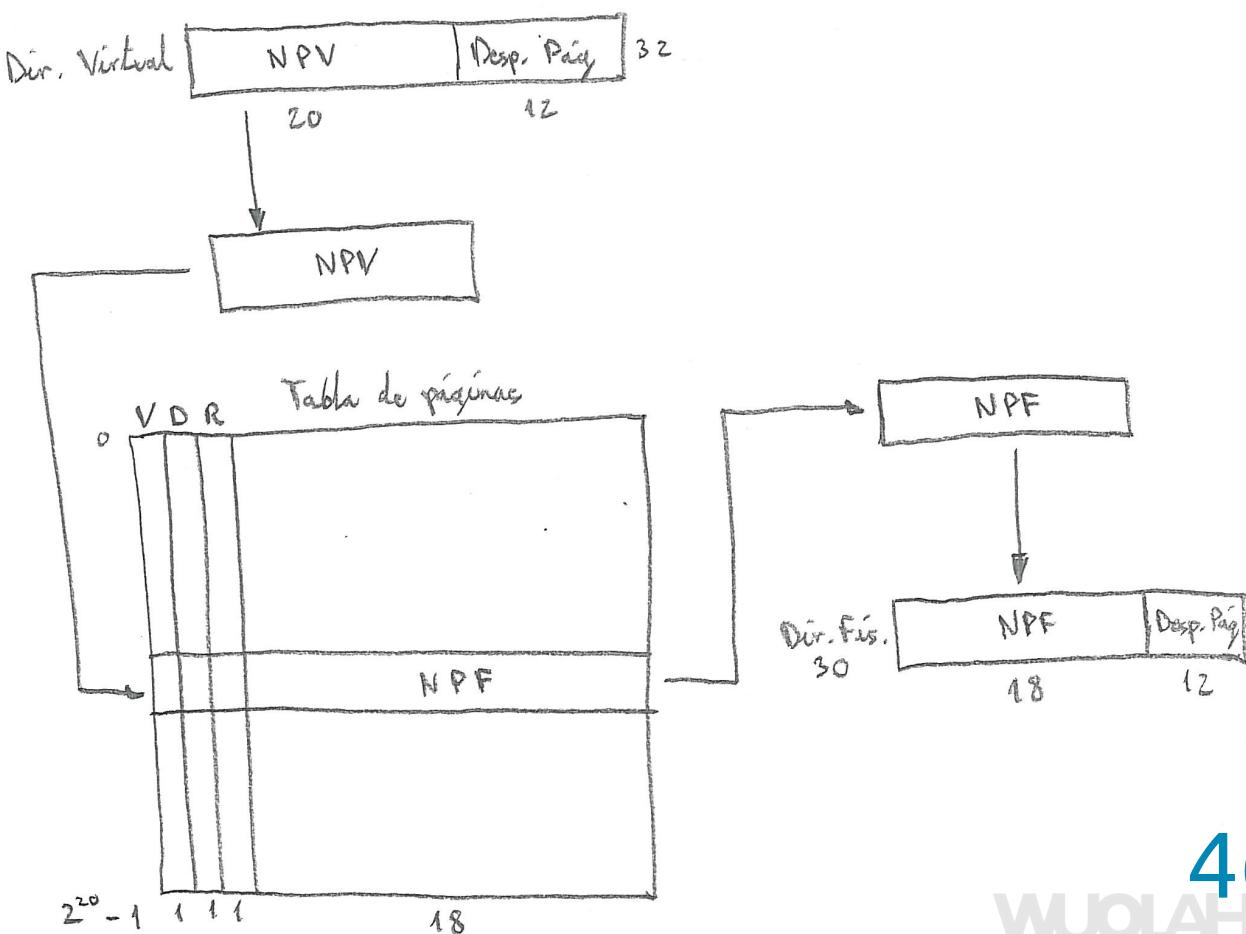
$$\text{Desplazamiento de página} = \log_2(4\text{KB}) = \log_2(2^2 \cdot 2^{10}) = 12 \text{ bits de desplazamiento de pági...}$$

Sabiendo esto podemos calcular el número de página virtual y físico:

$$\text{Nº de pág. virtual} = 32 - 12 = 20 \text{ bits}$$

$$\text{Nº de pág. físico} = 30 - 12 = 18 \text{ bits}$$

Por tanto, la estructura del sistema sería la siguiente:



Variable / tipo de acceso	Dir. física	Etiqueta	Línea en caché	Fallo / Acierto
producto [0] / write	0x0007100	0x000710	0	Fallo Forzoso
producto [1] / Write	0x0007104	0x000710	0	Fallo Forzoso
producto [2] / Write	0x0007108	0x000710	0	Fallo Forzoso
producto [3] / write	0x000711C	0x000710	0	Fallo Forzoso
producto [4] / Write	0x0007110	0x000711	1	Fallo Forzoso
producto [5] / write	0x0007114	0x000711	1	Fallo Forzoso
producto [6] / write	0x0007118	0x000711	1	Fallo Forzoso
producto [7] / Write	0x000711C	0x000711	1	Fallo Forzoso
producto [8] / Write	0x0007120	0x000712	2	Fallo Forzoso
producto [9] / write	0x0007124	0x000712	2	Fallo Forzoso

El estado final de la caché será el mismo que el inicial, ya que en ningún momento se ha llevado un dato a la caché. Es decir, la caché estaría vacía.

d) Frecuencia de fallos = $\frac{\text{Nº de Fallos}}{\text{Nº de accesos}} = \frac{10}{10} = 1$

Es decir, el 100% de los accesos producen fallos.

- e) Si se cambia la política de escritura de la memoria caché a CB-WA, entonces en caso de acierto se llevará el dato sólo a memoria caché, y en caso de fallo, se llevará a memoria caché y memoria principal. Entonces, el miss rate va a disminuir, ya que de producto [1] a producto [3], de producto [5] a producto [7] y producto [9] se produciría esta vez un acierto. Es decir, el miss rate sería:

Frecuencia de fallos = $\frac{\text{Nº de Fallos}}{\text{Nº de accesos}} = \frac{3}{10} = 0,3$. Habría un 30% de fallos respondiendo al total de aciertos.

Prueba 1 de evaluación continua, 2013/14. Grupo 3: Problema de memoria virtual y caché:

a)

La memoria virtual es de $4\text{ GB} = 2^2 \cdot 2^{30} = 2^{32}$ bytes, por tanto, las direcciones virtuales son 32 bits.

La memoria principal es de $1\text{ GB} = 2^{30}$ bytes, por tanto, las direcciones físicas son de 30 bits.

Para calcular cuántos bits de desplazamiento de página tenemos tanto para la dirección física como la virtual:

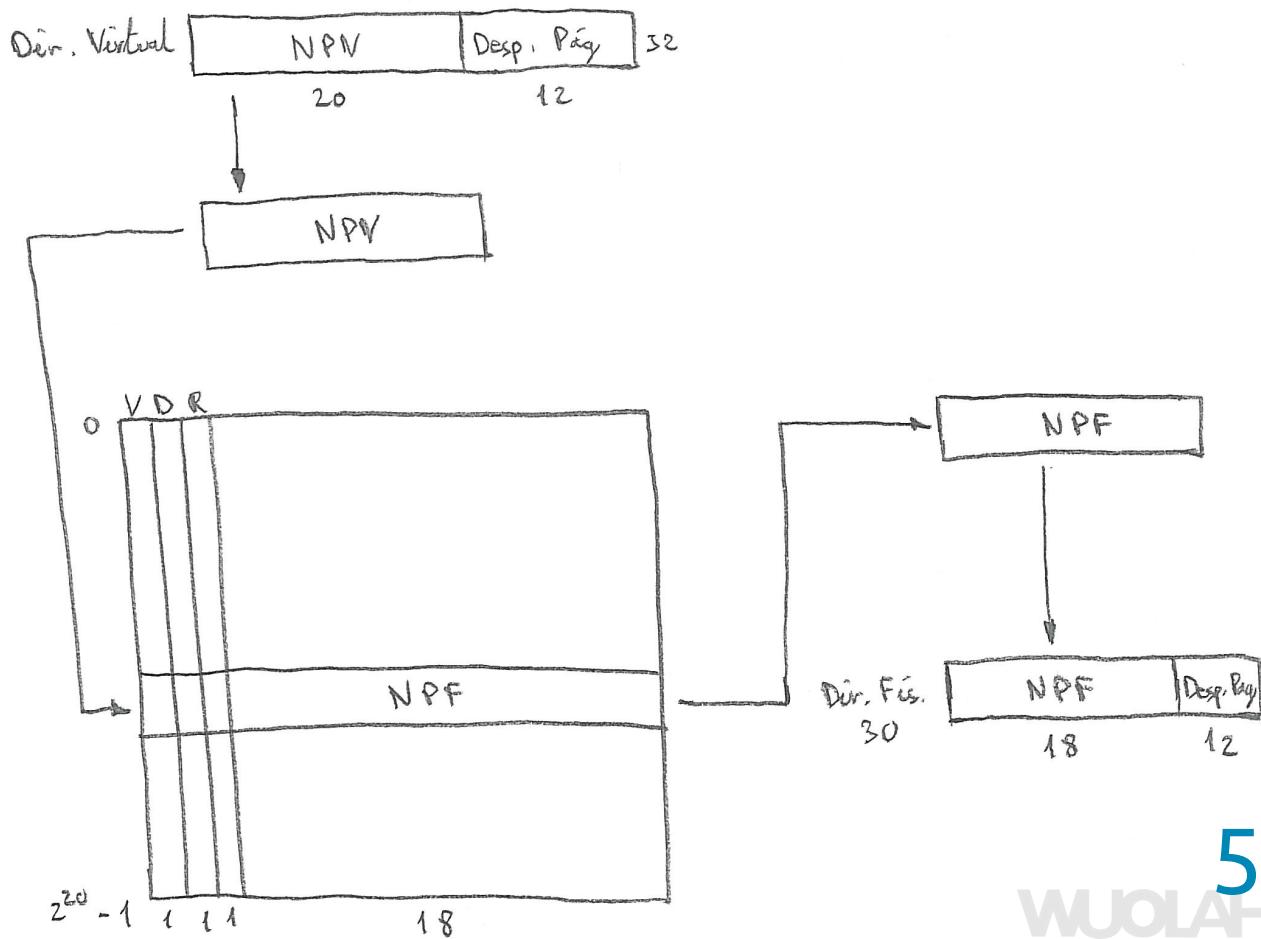
$$\text{Desplazamiento de página} = \log_2(4\text{ KB}) = \log_2(2^2 \cdot 2^{10}) = 12 \text{ bits de desplazamiento de página.}$$

Sabiendo esto podemos calcular el número de página virtual y físico:

$$\text{Nº de pág. virtual} = 32 - 12 = 20 \text{ bits}$$

$$\text{Nº de pág. físico} = 30 - 12 = 18 \text{ bits}$$

Por tanto, la estructura del sistema sería la siguiente



Variable / Tipo de acceso	Dir. física	Etiqueta	Línea en caché	Fallo / Acierto
factorial[0]/Lectura	0x001050 0	0x00105	Conj 0/Via 0	Fallo Forzoso
factorial[0]/Write	0x001050 4	0x00105	Conj 0/Via 0	Acierto
factorial[1]/Lectura	0x001050 8	0x00105	Conj 0/Via 0	Acierto
-----	-----	-----	-----	-----
factorial[4]/Lectura	0x001051 0	0x00105	Conj 1/Via 0	Fallo Forzoso
factorial[4]/Write	0x001051 4	0x00105	Conj 1/Via 0	Acierto
-----	-----	-----	-----	-----
factorial[8]/Lectura	0x001052 0	0x00105	Conj 2/Via 0	Fallo Forzoso
factorial[8]/Write	0x001052 4	0x00105	Conj 2/Via 0	Acierto
-----	-----	-----	-----	-----
36 veces				

El estado final de la caché sería el siguiente:

	V	Etiqueta	Blq		V	Etiqueta	Blq
C0	1	0x00105	factorial[0 ... 3]	---			
C1	1	0x00105	factorial[4 ... 7]	---			
C2	1	0x00105	factorial[8 ... 9]	---			
---	---	---	---	---	---	---	---
---	---	---	---	---	---	---	---
C15	0			---			

					Vía 0		Vía 3

d) Frecuencia de fallos = $\frac{\text{Nº de Fallos}}{\text{Nº de Accesos}} = \frac{3}{3+17+2+50+2+36} = 0,027$

Es decir, el 2,7% de los accesos producen fallos.

ARQUITECTURA DE COMPUTADORES (GRADO EN INGENIERÍA DEL SOFTWARE)
EXAMEN DE SEGUNDA CONVOCATORIA (CURSO 2015-2016)
SEPTIEMBRE, 2016

Apellidos y nombre:

P2 (2 puntos):

Suponga que el código que se muestra a continuación se ejecuta en un MIPS a 1GHz con segmentación de cauce, con el camino de datos optimizado para saltos, predicción de salto no tomado y toda la lógica de anticipación de resultados (desvíos) necesaria para evitar bloqueos. Suponga que a partir de la posición 1000 de se encuentran los siguientes bytes: 65, 70, 69, 77, 66, 66, 66, 0, 70, 71, 70, ...

- a) Realice la traza de ejecución de la primera iteración indicando los bloqueos que se producen y los desvíos que se activan.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
add \$t2,\$zero,\$zero	IF	ID	EX	MEM	WB												
e1: lb \$t1,1000(\$t2)		IF	ID	EX	MEM	WB											
beq \$t1,\$zero,e2			IF	-	-	ID	EX	MEM	WB								
addi \$t2,\$t2,1				-	-	IF	ID	EX	MEM	WB							
j e1				-	-		IF	ID	EX	MEM	WB						
e2: sw \$t2, 2000(\$zero)				-	-			IF _{sw}				ABORTO					

- b) Calcule el número total de instrucciones que se ejecutan

El salto condicional "beq \$t1,\$zero,e2" se produce cuando \$t1=0, y en la dirección de memoria 1007 hay un 0 como dato, por tanto en la 8^a iteración saltaría.

- c) Calcule el número total de bloqueos de datos que se producen.

Se producen 2 bloques de datos en la primera iteración, luego, en 8 iteraciones: $8 \cdot 2 = 16$ bloques de datos

$$N^{\circ} \text{ inst} = 1 + 4 \cdot 7 + 2 + 1 = 32 \text{ instrucciones}$$

add iteraciones instrucciones
 sw completas de la iteración incompleta

- d) Calcule el número total de bloques de control que se producen.

Se produce un bloqueo de control por cada salto tomado; por tanto 7 veces la instrucción "j" + 1 vez la instrucción beq = 8 bloques de control

- e) Calcule el tiempo total que consume el programa.

$$t_{CPU} = \frac{n^{\circ} \text{ inst}}{f_{relaj}} = \frac{\text{Latencia inicial} + n^{\circ} \text{ instrucciones} + n^{\circ} \text{ bloques}}{f_{relaj}} = \frac{4 + 32 + 16 + 8}{1 \cdot 10^9} = 60 \cdot 10^{-9} \text{ s}$$

- f) Calcule la aceleración que se obtiene con un procesador a 2 GHz en relación al MIPS suponiendo que el programa equivalente requiere la ejecución de 30 instrucciones con un CPI medio de 3 ciclos/instrucción.

El tiempo total para este nuevo procesador será: $t_{CPU'} = \frac{n^{\circ} \text{ inst} \cdot CPI \text{ medio}}{f_{relaj}} = \frac{30 \cdot 3}{2 \cdot 10^9} = 45 \cdot 10^{-9}$

Por tanto, la aceleración será:

$$a = \frac{t_{CPU'}}{t_{CPU}} = \frac{45 \cdot 10^{-9}}{60 \cdot 10^{-9}} = 0,75, \text{ es decir, es un } 75\% \text{ más rápido.}$$

Suponemos MIPS con camino de datos optimizado para saltos, predicción de salto, Arquitectura de Computadores Rubén Bueno Menéndez y toda la lógica de anticipación de resultados necesaria para evitar bloqueos. $f_{relaj} = 1 \text{ GHz}$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
addu \$t4, \$t4, \$zero	IF	ID	EX	MEM	WB													
lw \$t1, 1000(\$t4)		IF	ID	EX	MEM	WB												
addiu \$t4, \$t4, 4			IF	ID	EX	MEM	WB											
et1: lw \$t3, 1000(\$t4)				IF	ID	EX	MEM	WB										
slt \$t2, \$t3, \$t1					IF	ID	-	EX	MEM	WB								
beq \$t2, \$zero, et2						IF	-	-	ID	EX	MEM	WB						
addu \$t1, \$t3, \$zero.							-	-	IF _{full}	ABORTO								
et2: addiu \$t4, \$t4, 4							-	-		IF	ID	EX	MEM	WB				
seq \$t2, \$t4, 24							-	-			IF	ID	EX	MEM	WB			
beq \$t2, \$zero, et1							-	-				IF	-	ID	EX	MEM	WB	
sw \$t1, 2000(\$t4)							-	-					-	IF	ABORTO			

Considere que la memoria a partir de la posición 1000 contiene los siguientes valores en base 10: 16, 17, 20, 14, 22, 8.

Para el programa completo tenemos:

Nº de instrucciones:

$$3 + (3+3) \cdot 2 + 4 + 3 + (3+3) + 4 + 4 = 36 \text{ instrucciones}$$

3 primeras instrucciones
2 iteraciones de et1 y et2 con las 3 primeras instrucciones de cada etiqueta
1 iteración con et1 y et2 con las 3 primeras instrucciones de cada etiqueta
1 iteración con et1 y et2 completas, las 3 primeras instrucciones de cada etiqueta

Nº Total de bloques de datos:

Hay 3 bloques de datos en la primera iteración, por tanto, en el programa completo:

$$3 \cdot 5 = 15 \text{ bloques de datos!}$$

Nº total de bloques de control:

Se produce un bloqueo de control por cada salto tomado, por tanto:

$$(1+1) \cdot 2 + 1 + (1+1) = 7 \text{ bloques de control!}$$

Tiempo total que consume el programa

$$t_{CPU} = \frac{\text{Latencia inicial} + \text{nº instrucciones} + \text{nº bloques}}{f_{relaj}} = \frac{4 + 36 + 15 + 7}{1 \cdot 10^9} = 62 \cdot 10^{-9} \text{ s}$$

CPI medio:

$$CPI_m = \frac{\text{Latencia inicial} + \text{nº instrucciones} + \text{nº bloques}}{\text{nº instrucciones}} = \frac{4 + 36 + 15 + 7}{36} = 1,72 \text{ ciclos/instrucción}$$