

Cognoms: ..... Nom: .....

**Examen Final d'Arquitectura de Computadors**

**Curs 2012-2013 Q1**

**Problema 1. (4 puntos)**

Disponemos del diseño de dos procesadores que son totalmente compatibles a nivel de lenguaje máquina. Estos procesadores se denominan: HP de "High Performance" y LP de "Low Power". La siguiente tabla muestra las características más relevantes: carga capacitiva equivalente (C), voltaje de alimentación (V), intensidad de fugas ( $I_f$ ), frecuencia de funcionamiento (F) y CPI de una aplicación A (obtenido con el mismo ejecutable para ambos procesadores).

Procesador	C	V	$I_f$	F	CPI
HP	10 nF	1,2 V	10 A	4 GHz	0,5 c/i
LP	2 nF	0,9 V	1 A	1 GHz	1,5 c/i

a) **Calcula** el speed-up del procesador HP respecto el LP al ejecutar la aplicación A.

tiempo =  $I \cdot \text{CPI} / F$  -> no sabemos I, pero sabemos que es el mismo valor para ambos procesadores.

$$S = t_{LP} / t_{HP} = (I \cdot 1,5 \text{ c/i} / 1 \text{ GHz}) / (I \cdot 0,5 \text{ c/i} / 4 \text{ GHz}) = 12$$

Para calcular la disipación de potencia tendremos en cuenta la potencia debida a conmutación y la debida a fugas (la potencia debida a cortocircuito es despreciable).

b) **Escribe** la fórmula para cada una de las fuentes de disipación de potencia y **calcula** la potencia disipada por dicha causa para ambos procesadores.

	Potencia conmutación (Pc)	Potencia fugas (Pf)	Potencia total (Pt)
Formula	$P_c = C \cdot V^2 \cdot F$	$P_f = I_f \cdot V$	$P_t = P_c + P_f$
procesador HP	57,6 W	12 W	69,6 W
procesador LP	1,62 W	0,9 W	2,52 W

c) **Calcula** la ganancia en energía del procesador LP respecto al procesador HP al ejecutar la aplicación A.

$E = P \cdot t$  -> No sabemos t, pero sabemos que  $t_{LP} = 12 \cdot t_{HP}$  (apartado a)

$$G = E_{HP} / E_{LP} = (69,6 \text{ W} \cdot t_{HP}) / (2,52 \text{ W} \cdot 12 \cdot t_{HP}) = 2,3$$

El TDP (Thermal Design Point) de un chip es la potencia máxima que el fabricante garantiza puede disipar sin que su funcionamiento se vea comprometido. Nuestros diseñadores de circuitos nos han asegurado que el área no va a ser un problema, pero nos han comentado que disponemos de un TDP máximo de 92W para los procesadores, por lo que en un chip podemos poner o bien 1 solo procesador HP (que denominaremos chip H-1) o bien 36 procesadores LP (que denominaremos chip L-36).

Es posible paralelizar parte de la aplicación A de forma que el trabajo se distribuya equitativamente entre los procesadores disponibles (el overhead debido a sincronización es despreciable). La parte secuencial se seguirá ejecutando en un solo procesador independientemente de los que haya disponibles en el chip. Tanto la parte secuencial como la paralela consiguen el mismo CPI que la aplicación original en los respectivos procesadores (1,5 c/i en un procesador LP y 0,5 c/i en un procesador HP).

A uno de los diseñadores se le ha ocurrido que, dado que el área no es un problema, podríamos poner 1 procesador HP y 36 LP en el mismo chip, de forma que la parte secuencial de la aplicación se ejecute en el procesador HP y la paralela en los 36 procesadores LP. Durante la ejecución de la parte secuencial los 36 procesadores LP estarán apagados, mientras que durante la ejecución de la parte paralela el procesador HP estará apagado, de forma que nunca se superará el TDP máximo 92W. A esta tercera opción la denominaremos chip HL-136.

- d) **Rellena** la siguiente tabla con el **speed-up** que se obtendría con cada uno de los chips mencionados, en función del % de la parte paralela, **respecto a 1 procesador LP** (justifica muy brevemente cada respuesta).

Chip	0% (totalmente secuencial)	100% (totalmente paralela)	60% (el 40% restante es secuencial)
<b>H-1</b> (1 procesador HP)	12 del apartado a) HP 12 veces más rápido que LP	12 del apartado a)	12 del apartado a)
<b>L-36</b> (36 procesadores LP)	1 (en secuencial usamos un solo procesador)	36 (en paralelo usamos los 36 procesadores)	$S = 1/(0,4+0,6/36) = 2,4$
<b>HL-136</b> (1 procesador HP + 36 procesadores LP)	12 del apartado a)	36 (en paralelo usamos los 36 procesadores)	$S = 1/(0,4/12+0,6/36) = 20$

En el chip HL-136 ejecutamos una **aplicación B** (que tarda 2 horas) compuesta por 3 fases:

- La **Fase 1** (30 minutos) es totalmente secuencial y se ejecuta en el procesador HP (los 36 procesadores LP están apagados).
- La **Fase 2** (30 minutos) es totalmente paralela y se ejecuta en los 36 procesadores LP (el procesador HP está apagado).
- La **Fase 3** (1 hora) es una fase de entrada/salida en que la CPU tiene una actividad muy baja, por lo que no importa la velocidad de la CPU. Para reducir el consumo, la Fase 3 se ejecuta en uno de los procesadores LP (el procesador HP y 35 de los procesadores LP están apagados).

- e) **Calcula** la potencia media disipada por el chip HL-136 al ejecutar la aplicación B

Fase 1: 25% y 69,6 W -- Fase 2: 25% y  $36 \times 2,52 \text{ W} = 90,72 \text{ W}$  -- Fase 3: 50% y 2,52 W

$P_{media} = 0,25 \times 69,6 \text{ W} + 0,25 \times 90,72 \text{ W} + 0,5 \times 2,52 \text{ W} = 41,34 \text{ W}$

La **Fase 3** de entrada/salida se corresponde exclusivamente a accesos a disco y realiza un 60% de lecturas y un 40% de escrituras. El sistema de disco está formado por un RAID 5 de 10 discos. Cada uno de estos discos tiene una capacidad de 1 Terabyte, un ancho de banda efectivo de 200 Mbytes/s. Tanto las escrituras como las lecturas a disco se corresponden a accesos aleatorios. El controlador del RAID ordena las peticiones de acceso para aprovechar al máximo la concurrencia entre discos. En ambos casos consideraremos que disponemos de suficientes peticiones de lectura/escritura de forma que el controlador del RAID siempre puede aprovechar el ancho de banda de todos los discos físicos.

- f) **Calcula** la capacidad útil y el ancho de banda efectivo del RAID 5.

Capacidad útil = 9 discos útiles = 9 Tbytes

Ancho banda lecturas = 10 discos \* 200 MB/s/disco = 2 Tbytes/s

Ancho banda escrituras = 10 discos \* 200MB/s/disco / 4 = 500 Mbytes/s

Ancho de banda fase 3 =  $0,6 \times 2 \text{ Tb/s} + 0,4 \times 0,5 \text{ Tb/s} = 1,4 \text{ Tbytes/s}$

Este RAID 5 lo sustituimos por un RAID 51 formado por 2 grupos de 10 discos.

- g) **Calcula** la capacidad útil y el ancho de banda efectivo del nuevo RAID 51.

Capacidad útil = la misma

Ancho banda lecturas = el doble

Ancho banda escrituras = el mismo

Ancho de banda fase 3 =  $0,6 \times 4 \text{ Tb/s} + 0,4 \times 0,5 \text{ Tb/s} = 2,6 \text{ Tbytes/s}$

Sabemos que el tiempo de la Fase 3 depende exclusivamente de la velocidad a la que se transfieren los datos hacia/desde el sistema de disco.

- h) **Calcula** el tiempo de la aplicación B con el RAID 51.

$\text{speedup fase 3} = 2,6 \text{ TB/s} / 1,4 \text{ TB/s} = 1,86$

$\text{tiempo} = 60 \text{ min} / 1,86 + 30 \text{ min} + 30 \text{ min} = 1 \text{ hora } 32 \text{ minutos}$

Cognoms: ..... Nom: .....

**Examen Final d'Arquitectura de Computadors**

**Curs 2012-2013 Q1**

### Problema 2. (2 puntos)

Dado el siguiente código en C:

```
int MaxMin(int a, int *b);

int Final(int a, int *b){
    int max;

    max = MaxMin(a, b);
    if (max > 0) return max;
    else return *b;
}
```

a) **Traduce** la rutina Final a ensamblador del x86.

```
Final: PUSHL %EBP
        MOVL %ESP,%EBP
        PUSHL 12(%EBP)
        PUSHL 8(%EBP)
        CALL MaxMin
        ADDL $8,%ESP
        CMPL $0,%EAX
        JG END
        MOVL 12(%EBP),%EDX
        MOVL (%EDX),%EAX
END:    MOVL %EBP,%ESP
        POPL %EBP
        RET
```

Dado el siguiente código escrito en C:

```
typedef struct {
    char c1;
    char c2;
    char *pc;
    short M[3][3]
    int i;
} X;

int Rutina(char c, short s, int V[100], X uno, X *dos, int i){
    X vx[10];
    int j;
    ...
}
```

- b) **Dibuja** como quedaría almacenada en memoria la estructura **X**, indicando los desplazamientos respecto al inicio y el tamaño de todos los campos. **Dibuja** el bloque de activación de la función **Rutina**, indicando los desplazamientos relativos al registro EBP necesarios para acceder a los parámetros y a las variables locales y el tamaño de todos los parámetros y variables locales.

X (32 bytes)			Rutina		
1	c1	+0		Reg	
1	c2	+1	320	vx	-324
2	--		4	j	-4
4	pc	+4	4	ebp	<--%ebp
18	M	+8	4	@ret	
2	--		4	c	+8
4	i	+28	4	s	+12
			4	V	+16
			32	uno	+20
			4	dos	+52
			4	i	+56

Cognoms: ..... Nom: .....

## Examen Final d'Arquitectura de Computadors

Curs 2012-2013 Q1

### Problema 3. (2 puntos)

Dado el siguiente código escrito en ensamblador del x86:

```

    movl $0, %ebx
    movl $0, %esi
for:  cmpl $256*1024, %esi
      jge end
(a)   movl (%ebx, %esi, 4), %eax
(b)   addl %eax, 4*1024(%ebx, %esi, 4)
(c)   addl 12*1024(%ebx, %esi, 4), %eax
      addl $128, %esi
      jmp for
end:
```

Suponiendo que la memoria virtual utiliza páginas de tamaño 4KB y que se dispone de un TLB de 3 entradas completamente asociativo con reemplazo LRU, responde a las siguientes preguntas:

- a) Para cada uno de las instrucciones (etiquetas (a), (b) y (c)), **indica** a qué página de la memoria virtual se accede en cada una de las 16 primeras iteraciones del bucle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
b	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
c	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4

- b) **Indica** cuales de los accesos a memoria ejecutados en las 16 primeras iteraciones son fallo de TLB (F) (deja los aciertos en blanco)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a read	F															
b read	F								F							
b write																
c read	F								F							

- c) Calcula la cantidad de aciertos de TLB en TODO el bucle

2048 iteraciones

Acceso (a): 1 fallo y 2047 aciertos

Acceso (b): 256 fallos y 1792 aciertos para leer operando en memoria y 2048 aciertos para escribir resultado

Acceso (c): 256 fallos y 1792 aciertos para leer operando en memoria

Total aciertos:  $2047 + 1792 + 2048 + 1792 = 7679$  aciertos

- d) Calcula la cantidad de fallos de TLB en TODO el bucle

Total fallos:  $1 + 256 + 0 + 256 = 513$  fallos

#### Problema 4. (2 puntos)

Disponemos de un procesador conectado a un sistema de memoria con una memoria cache de nivel 1 de datos con política de escritura copy back + write allocate.

- a) **Define** de forma clara y concisa el comportamiento de la política copy back + write allocate en caso de lectura de un dato con fallo en L1.

El procesador accede a la memoria cache y ve que el dato no está. Se ha de traer a memoria cache el bloque de memoria principal donde se encuentra el dato buscado, y reemplazar el bloque (línea) de cache a la que se ha accedido. Si la línea accedida estaba marcada como "sucia" (ha sido modificada en la cache), la línea a reemplazar se escribe en memoria principal antes de traer el nuevo bloque con el dato solicitado. Si no ha sido modificada, la línea se sobrescribe con el nuevo bloque traído de memoria y el word solicitado se sirve en paralelo al procesador.

Sabiendo que el tiempo de leer o escribir un bloque en Memoria Principal (latencia + transmisión de datos) es de 9 ciclos.

- b) **Rellena** el siguiente cronograma, indicando la ocupación de los distintos recursos del sistema para una operación de lectura de un byte que provoca un fallo en el acceso a la cache, sabiendo que el bloque a reemplazar ha sido modificado en la memoria cache por el acceso inmediatamente anterior (usa H/M para hit/miss de MC, L/E para marcar los ciclos en que se lee/escribe en MP y D para indicar en que ciclo recibe el dato la CPU) .

CLK																												
Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
MC(H/M)	M																											
MP(L/E)		E	E	E	E	E	E	E	E	E	L	L	L	L	L	L	L	L										
CPU(D)																												

El sistema tiene una memoria principal formada por DIMMs de memoria DDR con 8 chips de 1 byte por DIMM, una latencia de fila de 2 ciclos, una latencia de columna de 2 ciclos y una latencia de precarga de 1 ciclo.

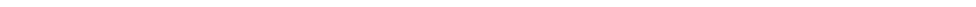
A la memoria DDR se realizan accesos de lectura/escritura en los que se lee/escribe un bloque de 64 bytes. Para indicar la ocupación de los distintos recursos utilizaremos la siguiente nomenclatura:

- AC: ciclo en que se envía el comando ACTIVE
- RD: ciclo en que se envía el comando READ
- PR: ciclo en que se envía el comando PRECHARGE
- @F: ciclo en que se envía la dirección de fila
- @C: ciclo en que se envía la dirección de columna
- D: transmisión de un paquete de datos

Una forma de mejorar el rendimiento de los acceso a MP consiste en aprovechar la localidad espacial de las filas de memoria (que los fabricantes llaman páginas). Para este apartado supondremos que el sistema tiene un controlador de memoria avanzado que no cierra la página (PRECHARGE) después de cada acceso. En caso de que un acceso se realice sobre la página abierta, no es necesario abrirla (ACTIVE). Sin embargo si el acceso se realiza sobre una página distinta, tenemos que cerrar la página anterior (PRECHARGE) y abrir (ACTIVE) la página que se desea acceder.

- c) **Rellena** el siguiente cronograma, indicando la ocupación de los distintos recursos del sistema con el controlador de memoria avanzado para la siguiente secuencia de accesos (inicialmente no hay ninguna página abierta), de forma que la secuencia se realice en el número mínimo de ciclos:

- Lectura de un bloque en la página x
- Lectura de un bloque en la página x
- Lectura de un bloque en la página y

CLK																												
Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Com	A	C				R	D							P	R	A	C			R	D							
@	@	F		@	C				@	C					@	F			@	C								
Datos						D	D	D	D	D	D	D	D	D							D	D	D	D	D	D		