

Problema 1. (1) Texe, CPI, F, Tc, Num instrucciones dinámicas

Dados dos procesadores con las siguientes características:

	CPI	Frecuencia
Procesador A	1,2 ciclos / instrucción	2 GHz
Procesador B	1,5 ciclos / instrucción	3 GHz

- a) Calcula el tiempo de ciclo de cada procesador
b) Suponemos que un programa P ejecuta $2 \cdot 10^6$ instrucciones dinámicas en ambos procesadores, ¿cuál es el tiempo de ejecución del programa P en cada procesador?
c) Si un programa X se compila y ejecuta en el procesador B en 1s, ¿cuántas instrucciones ejecuta? $N = \frac{1}{1,5 \cdot 3 \cdot 10^9} = 2 \cdot 10^9$
d) Si sabemos que el procesador A ejecuta el programa X un 25% más rápido que el procesador B, ¿cuántas instrucciones dinámicas son necesarias para ejecutar el programa X en el procesador A?
e) Suponiendo que todas las instrucciones (tanto de A como de B) se codifican en 4 bytes, ¿cuánto ocupa el programa X compilado para el procesador A?

$$\begin{aligned} a) T_a &= \frac{1}{2 \cdot 10^9 \text{ Hz}} = 5 \cdot 10^{-10} \text{ s} & T_b &= \frac{1}{3 \cdot 10^9 \text{ Hz}} = 3,33 \cdot 10^{-10} \text{ s} \\ b) T_{exeA} &= 2 \cdot 10^6 \cdot 1,2 \cdot 5 \cdot 10^{-10} \text{ s} = 1,2 \cdot 10^{-3} \text{ s} & T_{exeB} &= 2 \cdot 10^6 \cdot 1,5 \cdot 3,33 \cdot 10^{-10} \text{ s} = 10^{-3} \text{ s} \\ d) \frac{1}{T_{exeA}} &= 1,25 \rightarrow T_{exeA} = 0,8 \text{ s} & N &= \frac{0,8}{1,2 \cdot 5 \cdot 10^{-10}} = 1,33 \cdot 10^9 \text{ instrucciones} \\ e) 4 \cdot 1,33 \cdot 10^9 &= 5,32 \cdot 10^9 \text{ bytes} \end{aligned}$$

Problema 2. (2) Texe, Ley de Amdahl, SpeedUp, Capacidad de abstracción

Dado un programa que se ejecuta durante tres fases bien diferenciadas en un procesador Load/Store parecido al MIPS que funciona a 1 GHz, se mide su rendimiento en las distintas fases con los siguientes resultados:

	Instrucciones Dinámicas	Instrucciones Estáticas	CPI	Instrucciones Dinámicas de acceso a Memoria
Fase 1	10^6 instrucciones	10^6 instrucciones	2 ciclos / instrucción	10^6 instrucciones
Fase 2	10^9 instrucciones	10^6 instrucciones	3 ciclos / instrucción	10^7 instrucciones
Fase 3	10^9 instrucciones	10^6 instrucciones	4 ciclos / instrucción	10^7 instrucciones

- a) ¿Cuanto tiempo tarda en ejecutarse el programa en cuestión?
b) ¿Es un programa intensivo en memoria o en cálculo?
c) Si hacemos cambios en la arquitectura de forma que las instrucciones de la Fase 3 se ejecuten un 25% más rápido (y suponemos que no afectamos al resto de instrucciones), ¿cuál es la ganancia para el conjunto del programa?
d) ¿Cuál es el CPI de las instrucciones de acceso a memoria (Load/Store) en la Fase 1? y ¿Cuántas veces más rápidas deberían ser las instrucciones de memoria para que la Fase 1 tardase la mitad en ejecutarse?
e) Suponiendo que el CPI de las instrucciones de acceso a memoria es el mismo que en la fase 1, ¿Cuántas veces más rápidas deberían ser las instrucciones de memoria para que el programa tardase la mitad en ejecutarse?

$$a) T_{exe} = \frac{(10^6 \cdot 2 + 10^9 \cdot 3 + 10^9 \cdot 4)}{10^9} = 7 \text{ s}$$

b) En cálculo ya que de $2 \cdot 10^6 \cdot 10^9$ instrucciones dinámicas, $1,98 \cdot 10^9$ son de cálculo y $2,1 \cdot 10^7$ son de memoria

$$c) T_{exe3} = \frac{10^9 \cdot 4}{10^9} = 4 \text{ s} \rightarrow T_{exenou} = \frac{4}{1,25} = 3,2 \text{ s} \quad T_{exe} = \frac{(10^6 \cdot 2 + 10^9 \cdot 3)}{10^9} + 3,2 \text{ s} = 6,2 \text{ s}$$

$$\text{Speedup} = \frac{7}{6,2} = 1,129 \rightarrow 12,9\%$$

d) 2 ciclos/instrucción. Para que tardara la mitad tendría que tener la mitad del CPI por tanto, 1 ciclo/instrucción.

$$e) \frac{(10^6 \cdot 2 + 10^9 \cdot 3 + 10^7 \cdot x)}{10^9} = 3,5 \text{ s} \quad x = -\frac{122}{10}, \text{ per tant impossible}$$

Problema 6. (2) CPI, MIPS, MFLOPS, Amdahl

Hemos realizado un estudio sobre el comportamiento de un conjunto de programas representativo de un entorno de trabajo departamental en el procesador X. Este estudio nos dice el porcentaje de instrucciones de cada tipo que se ejecutan y su coste en ciclos de reloj. Esta información se resume en la siguiente tabla:

Tipo de instrucciones	% de uso	Coste en ciclos
Aritméticas de enteros	30%	2 ciclos
Acceso a Memoria	30%	5 ciclos
Coma Flotante	15%	7 ciclos
Saltos	15%	3 ciclos
Otras	10%	4 ciclos

Además, sabemos que en media se realizan 2 operaciones de coma flotante por cada instrucción de coma flotante ejecutada.

- Calcula el CPI medio para el procesador X. $CPI = 2 \cdot 0'3 + 5 \cdot 0'3 + 7 \cdot 0'15 + 3 \cdot 0'15 + 4 \cdot 0'1 = 4 \text{ CPI}$
- Suponiendo que el procesador X funciona a una frecuencia de 2 GHz calcula los MIPS y MFLOPS que obtendríamos en dicho conjunto de programas. $MIPS = \frac{1}{106 \cdot 4 \cdot 2 \cdot 10^9} = 500$ $MFLOPS = (500 \cdot 0'15 \cdot 2) = 150$

Un estudio más específico nos indica que una de las tareas que más tiempo consume es la gestión de procedimientos y funciones (subrutinas en general). Uno de los ingenieros de la compañía ha diseñado una modificación del procesador X (que llamaremos X1) con el objetivo de reducir el coste de la gestión de subrutinas. Los experimentos realizados con X1 revelan lo siguiente:

- La duración del ciclo de reloj del procesador X1 aumenta un 5%.
- El procesador X1 ejecuta un 25% menos de instrucciones de acceso a memoria que el procesador X.
- El procesador X1 ejecuta un 15% menos de instrucciones aritméticas con enteros que el procesador X.
- Para el resto de instrucciones el recuento queda inalterado.
- ¿Qué procesador es más rápido? Justificad cuantitativamente la decisión.
- Calcula los MIPS y MFLOPS para X1. $MIPS = \frac{1}{106 \cdot 4 \cdot 2 \cdot 0'525 \cdot 10^9} = 473'82$ $MFLOPS = (473'82 \cdot 0'15) \cdot 2 = 142'15$

Problema 9. (2) Coste, Energía, Consumo, Sostenibilidad

Un procesador tiene una superficie de 200 mm² y se fabrica en una oblea de silicio con una superficie útil de 63.200 mm². El coste de una oblea y el proceso de impresión y verificación de los datos (dies) es de 23.700 €. Durante este proceso el factor de yield es del 75%.

- ¿Cuál es el coste de un dado (die)? $Coste = \frac{23700}{0'75 \cdot 63200} = 100'6$

El coste empaquetado y testeado final es de 20€ por dado y el yield final de los circuitos integrados después del testeado final es del 92%. El fabricante quiere obtener un 50% de beneficio sobre el coste de fabricación.

- En media, cuantos circuitos integrados funcionales se producen por oblea? $\frac{63200}{200} \cdot 0'75 \cdot 0'92 = 218$ dices
- ¿Cuál será el precio de venta de los procesadores? $\frac{100'6}{0'92} \cdot 1'5 = 195'65€$

Se denomina "embodied energy" al coste energético de producir un producto. Este coste no solo incluye la energía consumida durante la fabricación sino también la energía consumida en producir los componentes, transporte, embalaje y comercialización del producto. Ignorar este coste energético puede conducir a mucha gente a pensar que sustituir un dispositivo más antiguo por uno más moderno con un menor consumo va a redundar en un ahorro energético y por consiguiente es beneficioso desde el punto de vista medioambiental. Sin embargo debemos tener en cuenta la "embodied energy" para determinar si dicha sustitución resulta beneficiosa.

Deseamos sustituir nuestros antiguos procesadores (que cumplen perfectamente con su tarea) por el procesador mencionado anteriormente ya que creemos que será positivo desde el punto de vista ambiental. Se ha estimado que la "embodied energy" del nuevo procesador es de 200 MJoules. En la siguiente tabla se muestra el uso medio y los consumos de ambos procesadores usados en entorno de sobremesa y servidor.

Estado	uso diario sobremesa	uso diario servidor	consumo procesador viejo	consumo procesador nuevo
Pleno rendimiento	2 horas	10 horas	50W	40W
Inactivo/bajo rendimiento	7 horas	14 horas	10W	5W
suspendido/apagado	15 horas	0 horas	0W	0W

Para un uso en entorno sobremesa:

- Calcular el consumo anual de ambos procesadores (en MJoules/año) $Nº \text{días} = 7200 \cdot 50 + 25200 \cdot 10 = 612 \cdot 10^3 \text{ J/día}$ $= 223'38 \text{ MJ/año}$
- ¿Durante cuanto tiempo deberemos tener el nuevo procesador para amortizar la "embodied energy"? $\frac{200}{223'38 - 15'11} = 2'77 \text{ años}$ $25200 \cdot 5 = 151'11 \text{ MJ/año}$

Para un uso en entorno servidor:

- Calcular el consumo anual de ambos procesadores (en MJoules/año)
- ¿Durante cuanto tiempo debemos tener el nuevo procesador para amortizar la "embodied energy"?
- ¿Cuál crees que sería la mejor decisión en un entorno en donde tenemos computadores de los dos tipos (sobremesa y servidor)? USAR LOS NUEVOS

Los fabricantes de procesadores suelen cambiar el "socket" cada poco tiempo (2-3 años) e incluso tienen varias líneas distintas simultáneamente con lo que si nuestro procesador antiguo tiene 3 años o más es muy probable que para usar el nuevo tengamos que cambiar también la placa base y posiblemente la memoria. Se ha estimado que la embodied energy de CPU+placa+memoria es de unos 2000 MJoules para entorno sobremesa y 3000 MJoules para entorno servidor. La siguiente tabla muestra el consumo del conjunto para ambos entornos.

Estado	uso diario sobremesa	consumo conjunto viejo	consumo conjunto nuevo	uso diario servidor	consumo conjunto viejo	consumo conjunto nuevo
Pleno rendimiento	2 horas	100W	80W	10 horas	120W	100W
Inactivo/bajo rendimiento	7 horas	30W	20W	14 horas	40W	30W
suspendido/apagado	15 horas	10W	5W	0 horas	10W	5W

- ¿Porque crees que el entorno servidor tiene un mayor coste y mayores consumos a pesar de usar la misma CPU? Porque consume mas memoria
- ¿Durante cuanto tiempo debemos tener el nuevo conjunto (CPU+placa+memoria) para amortizar la "embodied energy"? (para ambos sistemas)
- Razona bajo que circunstancias se podria considerar ético o no, desde un punto de vista sostenible, un cambio de socket por parte de un fabricante.

j) Viejo = $7200 \cdot 100 + 25200 \cdot 30 + 54000 \cdot 10 = 735'84 \frac{\text{MJ}}{\text{año}}$
 Nuevo = $7200 \cdot 80 + 25200 \cdot 20 + 54000 \cdot 5 = 492'75 \frac{\text{MJ}}{\text{año}}$
 $\frac{2000}{735'84 - 492'75} = 8'23 \text{ años}$

Viejo = $36000 \cdot 120 + 50400 \cdot 40 = 2'131 \cdot 10^3 \frac{\text{MJ}}{\text{año}}$
 Nuevo = $36000 \cdot 100 + 50400 \cdot 30 = 1'87 \cdot 10^3 \frac{\text{MJ}}{\text{año}}$
 $\frac{3000}{2'131 \cdot 10^3 - 1'87 \cdot 10^3} = 6'82 \text{ años}$

No sería el caso
 ya que no
 hablamos
 consumiendo toda
 la embodied energy
 (8'23 o 6'82 años).

Problema 11.
 A pleno rendimiento, una CPU funciona a una frecuencia de 3 GHz y está alimentada a 1,6 V. En modo bajo consumo la CPU funciona a una frecuencia de 1 GHz y está alimentada a 1 V. Hemos medido que el consumo de la CPU en alto rendimiento es de 120W y en modo bajo consumo es de 27,5 W. En estos datos solo se considera la potencia debida a conmutación y la debida a fugas. Tanto la corriente de fugas (I) como la carga capacitiva equivalente (C) son las mismas en ambos modos.

- Calcula la corriente de fugas (I) y la carga capacitiva equivalente (C) de la CPU (usar prefijo más adecuado del SI)

$$C = \frac{120 \text{ W}}{1.6 \cdot 3 \cdot 10^9} = 1'5625 \cdot 10^{-8} = 15'625 \text{ nF}$$

$$I = \frac{27.5 \text{ W}}{1 \text{ V}} = 27.5 \text{ A}$$

Problema 12. (1) fiabilidad, disponibilidad

Tenemos un sistema compuesto por los elementos mostrados en la tabla siguiente. La tabla también muestra el numero de componentes de cada tipo y el tiempo medio hasta fallo (MTTF) de cada componente.

Componente	Fuente alimentación	CPU	Placa base	DIMM	GPU	Disco duro
Nº	1	1	1	4	1	8
MTTF (horas)	125.000	1.000.000	200.000	1.000.000	500.000	100.000

- a) Calcula el tiempo medio hasta fallos del sistema
 b) El tiempo medio para reemplazar un componente que ha fallado (MTTR) es de 20 horas. Calcula el tiempo medio entre fallos (MTBF).
 c) ¿Cual es la disponibilidad del sistema?

$$a) \frac{1}{MTTF_{sistema}} = \frac{1}{125 * 10^3} + \frac{1}{10^6} + \frac{1}{200 * 10^3} + \frac{4}{10^6} + \frac{1}{500 * 10^3} + \frac{8}{100 * 10^3}$$

$$MTTF_{sistema} = 10000 \text{ horas}$$

$$b) MTBF = 10020 \text{ horas}$$

$$c) \text{Disponibilidad} = \frac{10000}{10020} = 0.998$$

Problema 1. (1) Operadores lógicos

Suponed que x e y, variables de tamaño 1 byte, tienen los valores 0x66 y 0x93 respectivamente. Rellenad la siguiente tabla, indicando los valores resultantes de aplicar las siguientes expresiones en C:

Expresión	valor binario	valor hex	Expresión	valor binario	valor hex
x & y	0000 0010	0x02	x && y	0000 0001	0x01
x y	1111 0111	0xF7	x y	0000 0001	0x01
~x ~y	1111 1101	0xFD	!x !y	0000 0000	0x00
x & !y	0000 0000	0x00	x && ~y	0000 0001	0x01

Problema 2. (1) Desplazamientos

Rellenad la tabla que se muestra a continuación. El ejercicio consiste en aplicar desplazamientos lógicos y aritméticos sobre un conjunto de variables de tamaño byte.

x		x << 4		x >> 3 (lógico)		x >> 3 (aritmético)	
hex	binario	hex	binario	hex	binario	hex	binario
0xE0	1111 0000	0x00	0000 0000	0x1E	0001 1110	0xFE	1111 1110
0x0F	0000 1111	0xF0	1111 0000	0x01	0000 0001	0x01	0000 0001
0xCC	1100 1100	0xC0	1100 0000	0x19	0001 1001	0xF9	1111 1001
0x55	0101 0101	0x50	0101 0000	0x0A	0000 1010	0x0A	0000 1010
0x80	1000 0000	0x00	0000 0000	0x10	0001 0000	0xF0	1111 0000
0x02	0000 0010	0x20	0010 0000	0x00	0000 0000	0x00	0000 0000

Problema 5. (2) Traducción

Dada la siguiente definición de variables globales:

```
char A[256];
char tabla[256];
```

Traducid a ensamblador de IA32 el siguiente fragmento de código:

```
for (i=0; i<256; i++)
    A[i] = tabla[A[i]];
```

```
movl $A, %eax
movl $tabla, %ecx
movl $0, %ebx
for: cmpl $256, %ebx
      jge fifor
      movsbl (%eax, %ebx), %edx
      movb (%ecx, %edx), %dl
      movb %dl, (%eax, %ebx)
      incl %ebx
      jmp for
fifor:
```

Problema 6. (2) Traducción

Dado el siguiente código escrito en C:

```
int *sorpresa (int i, int *x)
{
    if (i>-10 && i<10)
        *x = i;
    else
        x = &i;
    return x;
}
```

Teniendo en cuenta que los parámetros de la función son accesibles en las siguientes direcciones: i en 8(%ebp), dirección de x en 12(%ebp), traducid a ensamblador del IA32 el cuerpo de la subrutina.

sorpresa: pushl %ebp

movl %esp, %ebp

movl 8(%ebp), %ebx

movl 12(%ebp), %ecx

cmpl \$-10, %ebx

jle else

cmpl \$10, %ebx

jge else

movl %ebx, (%ecx)

jmp fi

else: leal 8(%ebp), %ebx

movl %ebx, 12(%ebp)

fi: movl 12(%ebp), %eax

popl %ebp

ret

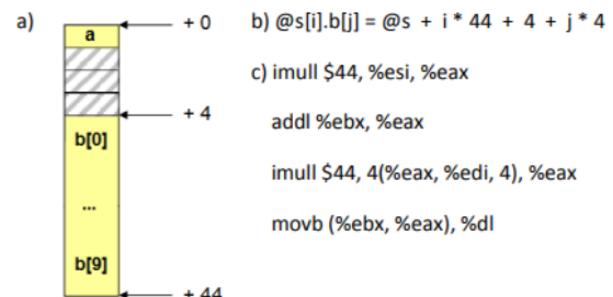
Problema 9. (1) Structs

Dada la siguiente declaración de variables:

```
typedef struct{
    char a;
    int b[10];
} elem;
elem s[100];
```

Sabemos que la dirección de `s` se encuentra en `%ebx` y las variables `i`, `j` y `x` se encuentran respectivamente en los registros `%esi`, `%edi` y `%dl`

- Dibujad el struct `elem`.
 - Determinad cómo se calcula la dirección de memoria donde se almacena el dato `s[i].b[j]`
 - Escribid la secuencia de instrucciones necesaria para codificar la siguiente sentencia
- ```
x = s[s[i].b[j]].a
```

**Problema 10. (1) Subrutinas**

Dada la siguiente función escrita en C:

```
int calcula(int M[10][10], int m, int n)
{ int i, suma, fila;
 suma = 0;
 fila = 0;
 for (i = m; i<n; i++)
 suma = suma + Normaliza(M[fila][i], &fila);
 return (suma + 1);
}
```

- Dibujad el bloque de activación de la función.
- Traducid la función a ensamblador del IA32. Suponed que la función `Normaliza` ya está programada.

|          |                            |     |                     |
|----------|----------------------------|-----|---------------------|
| calcula: | pushl %ebp                 |     | call Normaliza      |
|          | movl %esp, %ebp            | -12 |                     |
|          | subl \$12, %esp            | -8  | addl \$8, %esp      |
|          | pushl %ebp                 | -4  | addl %eax, -8(%ebp) |
|          | @RET                       | +8  | incl %ebx           |
|          | @M                         | +12 | jmp for             |
| for:     | int i, suma, fila;         | +16 |                     |
|          | int m, n;                  | +20 | fifor:              |
|          | movl \$0, -8(%ebp)         |     | movl -8(%ebp), %eax |
|          | movl \$0, -4(%ebp)         |     | incl %eax           |
|          | movl 12(%ebp), %ebx        |     | popl %ebx           |
|          | cmpl 16(%ebp), %ebx        |     | movl %ebp, %esp     |
|          | jge fifor                  |     | popl %ebp           |
|          | leal -4(%ebp), %eax        |     | ret                 |
|          | pushl %eax                 |     |                     |
|          | movl -4(%ebp), %edx        |     |                     |
|          | imull \$10, %edx           |     |                     |
|          | addl %ebx, %edx            |     |                     |
|          | movl 8(%ebp), %ecx         |     |                     |
|          | movl (%ecx, %edx, 4), %edx |     |                     |
|          | pushl %edx                 |     |                     |

**Problema 14.**

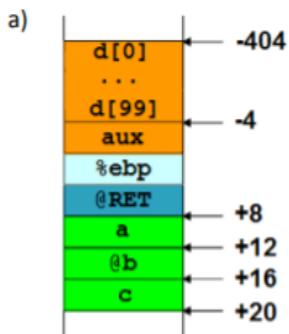
Dada la siguiente función escrita en C:

```
void examen (int a, int b[100], int *c)
{ int d[100];
 int aux;
 ...
}
```

- a) Dibujad el bloque de activación de la función.

Traducid a ensamblador del IA32 las siguientes sentencias suponiendo que se encuentran en el cuerpo de la rutina **examen**:

- b) Traducid a ensamblador IA32 la siguiente sentencia suponiendo que se encuentra en el cuerpo de la rutina  
`examen(0, d, &aux);`
- c) Traducid a ensamblador IA32 la siguiente sentencia suponiendo que se encuentra en el cuerpo de la rutina  
`for (aux = 0; aux < 100; aux++)
 b[aux] = d[aux];`
- d) Traducid a ensamblador IA32 la siguiente sentencia suponiendo que se encuentra en el cuerpo de la rutina  
`examen(a, b, c);`



b) leal -4(%ebp), %eax  
leal -404(%ebp), %ecx  
pushl %eax  
pushl %ecx  
pushl \$0  
call examen

c) movl \$0, %ecx

for: cmpl \$100, %ecx  
jge fifor  
leal -404(%ebp), %eax  
movl (%eax, %ecx, 4), %eax  
movl 12(%ebp), %edx  
movl %eax, (%edx, %ecx, 4)  
incl %ecx  
jmp for

fifor:

d) pushl 16(%ebp)

pushl 12(%ebp)

pushl 8(%ebp)

call examen

**Problema 18.**

Considerad el siguiente código escrito en C, suponiendo que las constantes N y M han sido declaradas previamente en un #define.

```
int mat1[M][N];
int mat2[N][M]

int SumaElemento(int i, int j)
{
 return mat1[i][j] + mat2[i][j];
}
```

Para esta subrutina el compilador genera el siguiente código en ensamblador:

```
SumaElemento:
 pushl %ebp
 movl %esp, %ebp

 movl 8(%ebp), %eax ← %eax = i
 movl 12(%ebp), %ecx ← %ecx = j
 sall $2, %ecx ← %ecx = 4j
 leal (,%eax,8), %edx ← %edx = 8i
 subl %eax, %edx ← %edx = 8i - i
 leal (%eax, %eax, 4), %eax ← %eax = i + 4i
 movl mat2(%ecx, %eax, 4), %eax ← %eax = M[@mat2 + 4j + 4(i + 4i)] = mat2[i][j]
 addl mat1(%ecx, %edx, 4), %eax ← %eax = mat2[i][j] + M[@mat1 + 4j + 4(8i - i)]
 = mat2[i][j] + mat1[i][j]
 movl %ebp, %esp
 popl %ebp
 ret

@mat1[i][j] = @mat1 + 4(i * N + j) = @mat1 + 4iN + 4j
@mat2[i][j] = @mat2 + 4(i * M + j) = @mat2 + 4iM + 4j
```

- ¿Cuánto valen las constantes M y N?
- ¿Cuántas instrucciones estáticas tiene el código?
- ¿Cuántas instrucciones dinámicas tiene el código?
- ¿Cuántos accesos memoria se producen al ejecutar este código?
- Suponiendo que cada ciclo se ejecutan 0,8 instrucciones si éstas no acceden a la memoria de datos y 0,5 si acceden a la memoria de datos, ¿cuántos ciclos tarda en ejecutarse el programa anterior?
- Si cambiamos la memoria del procesador de forma que los accesos a las instrucciones fuesen más rápidos y se ejecutaran 0,1 instrucciones más por ciclo ¿cuál sería la ganancia para este programa?

$$a) 4iM = 4i + 16i = 20i \implies M = \frac{20i}{4i} = 5$$

$$4iN = 32i - 4i = 28i \implies N = \frac{28i}{4i} = 7$$

- 13
- 13
- 9
- Si acceden a memoria: 0.5 i/c (2 c/i); si no acceden a memoria: 0.8 i/c (1.25 c/i)

$$9 * 2 + 4 * 1.25 = 23 \text{ ciclos}$$

**Problema 19.**

Dado el siguiente código escrito en C:

```

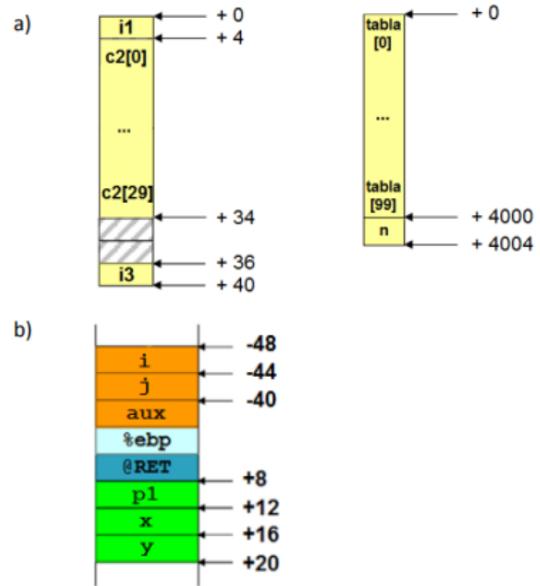
typedef struct {
 int i1;
 char c2[30];
 int i3;
} sx;

typedef struct {
 sx tabla[100];
 int n;
} s2;

int F(sx *p2, int y);

int examen(s2 *p1, int *x, int y)
{ int i, j;
 sx aux;
 . . .
}

```



a) Dibujad como quedarían almacenadas en memoria las estructuras sx y s2, indicando claramente los desplazamientos respecto el inicio y el tamaño de todos los campos.

b) Dibujad el bloque de activación de la función examen, indicando claramente los desplazamientos relativos al EBP necesarios para acceder a los parámetros y las variables locales.

c) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
return (*x+aux.i3);
```

d) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
aux.i1 = F(&(*p1).tabla[j], y);
```

e) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
i = j * y;
```

f) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
aux.c2[i] = aux.c2[23];
```

g) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
for (i=0; (i<y) && (i<(*p1).n) ; i=i+5)
 (*p1).tabla[i].i1 = (*p1).tabla[i].i3 + i;
```

h) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
if (aux.i1 != y)
 aux.i3 = i;
else
 aux.i3 = j;
```

i) Traducid la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen:

```
i = 0;
while (aux.c2[i] != '.') {
 aux.c2[i] = '#';
 i++;
}
```

c) movl 12(%ebp), %eax

movl (%eax), %eax

addl -4(%ebp), %eax

d) movl 8(%ebp), %eax

movl -44(%ebp), %ecx

imull \$40, %ecx

addl %ecx, %eax

movl 16(%ebp), %ecx

pushl %ecx

pushl %eax

call F

addl \$8, %esp

movl %eax, -40(%ebp)

g) pushl %esi

movl \$0, %eax

movl 8(%ebp), %ecx

cmpl 16(%ebp), %eax

jge fifor

cmpl 4000(%ecx), %eax

jge fifor

imull \$40, %eax, %edx

addl %ecx, %edx

movl %edx, %esi

movl 36(%esi), %esi

addl %eax, %esi

movl %esi, (%edx)

addl \$5, %eax

jmp for

fifor: popl %esi

h) movl -40(%ebp), %eax

cmpl 16(%ebp), %eax

je else

movl -48(%ebp), %ecx

jmp end

else: movl -44(%ebp), %ecx

end: movl %ecx, -4(%ebp)

i) movl \$0, %eax

leal -40(%ebp), %ecx

while: cmpb '\$.', 4(%ecx, %eax)

je fiwhile

movb '\$#', 4(%ecx, %eax)

incl %eax

jmp while

fiwhile:

#### Problema 4. Repaso Cache

Se quiere diseñar la memoria cache para un determinado procesador. Se barajan dos alternativas:

- (1) Con escritura inmediata (write through) y sin carga en caso de fallo de escritura.
- (2) Con escritura cuando reemplazo (copy back) y carga en caso de fallo de escritura

Se han obtenido por simulación las siguientes medidas:

- porcentaje de escrituras: 20%
- porcentaje de bloques modificados: 33.33%
- tasa de aciertos caso (1): 0.9
- tasa de aciertos caso (2): 0.85

El tiempo de acceso a memoria cache es de 10 ns y el tiempo de memoria principal para escribir una palabra es de 80 ns. Para leer o escribir un bloque en la memoria principal se emplean 100 ns.

Se pide:

- a) **Calculad** el tiempo invertido en ejecutar 1000 accesos para las dos alternativas. Detallad el número de accesos de cada tipo y el tiempo empleado para cada uno de ellos.

Alternativa 2 (escritura retardada con asignación)

a) Alternativa 1 (escritura inmediata sin asignación)

$$tma = 0.8 * 10 + 0.1(0.3333(2 * 100 + 2 * 10) + 0.6666(100 + 2 * 10)) = 31.5 \text{ ns}$$

$$tma = 0.8(0.9 * 10 + 0.1(10 + 100 + 10)) + 0.2 * 80 = 32.8 \text{ ns} \quad \xrightarrow[1000 \text{ accesos}]{\quad} 32800 \text{ ns}$$

$$\xrightarrow[1000 \text{ accesos}]{\quad} 31500 \text{ ns}$$

- b) **Indicad** qué alternativa sería la más rápida para un programa que sólo realizará lecturas.

b) La alternativa 1, ya que tiene mejor/mayor tasa de aciertos.

c)

**Indicad** qué motivos pueden existir para que la escritura de una palabra tarde ligeramente menos que la escritura de un bloque.

c) tamaño de un bloque > tamaño de una palabra.

#### Problema 5. Repaso Cache

Tenim una CPU amb les següents característiques:

- CPI ideal: 1.5 cicles/instrucció
- Temps de cicle (Tc): 10 ns
- Nombre de referències per instrucció (nr): 1.6 referències/instrucció
- Cache d'instruccions i dades separades
- Cache de dades amb **copy back** i **write allocate**.

Les característiques de les dues caches son les següents:

| Característica                                       | Instruccions | Dades        |
|------------------------------------------------------|--------------|--------------|
| Numero de referències a memòria per instrucció (nr)  | 1 ref/inst   | 0.6 ref/inst |
| Taxa de fallades (m)                                 | 4 %          | 10 %         |
| Penalització (Tp) al reemplaçar un bloc no modificat | 10 cicles    | 15 cicles    |
| Penalització (Tp) al reemplaçar un bloc modificat    | ---          | 20 cicles    |
| Temps de servei en cas d'encert (Tsa)                | 1 cicles     | 1 cicles     |
| Percentatge de blocs modificats (pm)                 | 0 %          | 20 %         |

- a) **Calculeu** el temps mig d'accés a memòria en cicles (TmaI) pels accessos a instruccions?

- b) **Calculeu** el temps mig d'accés a memòria en cicles (TmaD) pels accessos a dades?

- c) **Calculeu** el temps mig d'accés a memòria en cicles (Tma) per tots els accessos?

- d) **Calculeu** el temps d'execució en ns. (Texec) d'una instrucció?

$$c) Tma = \frac{1.4 \times 1 + 2.6 \times 0.6}{1.6} = 1.85 \text{ cicles}$$

$$d) CPI = CPI_{ideal} + CPI_{mem}$$

$$a) TmaI = t_{hit} + t_{miss} \times t_{penalització}$$

$$= 1 + 0.04 \times 10$$

$$= 1.4 \text{ cicles}$$

$$= CPI_{ideal} + n_{ref} (Tma - t_{hit})$$

$$= 1.5 + 1.6 (1.85 - 1)$$

$$= 2.86 \text{ cicles}$$

$$b) TmaD = 1 + 0.1 (0.2 \times 20 + 0.8 \times 15) = 2.6 \text{ cicles}$$

$$Texec = N \times CPI \times Tc$$

$$= 1 \times 2.86 \times 10$$

$$= 28.6 \text{ ns}$$

### Problema 7. Repaso Memoria Virtual

Dado el siguiente código escrito en ensamblador x86:

```

 movl $0, %ebx
 movl $0, %esi
for: cmpl $512*1000, %esi
 jge end
(a) movl (%ebx, %esi, 4), %eax
(b) addl %eax, 8*1024(%ebx, %esi, 4)
(c) movl %eax, 16*1024(%ebx, %esi, 4)
 addl $512, %esi
 jmp for
end:

```

Suponiendo que la memoria utiliza páginas de tamaño 8KB y que utilizamos un TLB de 4 entradas (reemplazo LRU), responde a las siguientes preguntas:

- a) Para cada uno de los accesos (etiquetas a, b, c), indica a qué página de la memoria virtual se accede en cada una de las 17 primeras iteraciones.

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| a | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2  | 2  | 3  | 3  | 3  | 3  | 4  |
| b | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3  | 3  | 4  | 4  | 4  | 4  | 5  |
| c | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4  | 4  | 5  | 5  | 5  | 5  | 6  |

- b) Calcula la cantidad de aciertos de TLB, en todo el bucle: ... 3748 .....
- c) Calcula la cantidad de fallos de TLB, en todo el bucle: ... 252 .....

Suponiendo que la memoria utiliza páginas de tamaño 4KB y que utilizamos un TLB de 4 entradas (reemplazo LRU), responde a las siguientes preguntas:

- d) Para cada uno de los accesos (etiquetas a, b, c), indica a qué página de la memoria virtual se accede en cada una de las 17 primeras iteraciones.

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| a | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5  | 5  | 6  | 6  | 7  | 7  | 8  |
| b | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | 7  | 7  | 8  | 8  | 9  | 9  | 10 |
| c | 4 | 4 | 5 | 5 | 6 | 6 | 7 | 7 | 8 | 8 | 9  | 9  | 10 | 10 | 11 | 11 | 12 |

- e) Calcula la cantidad de aciertos de TLB, en todo el bucle: ... 2500 .....
- f) Calcula la cantidad de fallos de TLB, en todo el bucle: ... 1500 .....

1 página = 8192 bytes



| Iteración    | Acceso                            | TLB        |    |   | Observación                                                                                                                                                                   |
|--------------|-----------------------------------|------------|----|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |                                   | 0          | 1  | 2 |                                                                                                                                                                               |
| 0 (i = 0)    | (a) M[0], página 0, MISS          | 0          |    |   |                                                                                                                                                                               |
|              | (b) M[8192], página 1, MISS + HIT | 1, 0       |    |   | • for (int i = 0; i < 512000; i += 512) { ... } $\Rightarrow$ 1000 iteraciones, en cada una hacemos 4 accesos a memoria (1 movl + 2 addl + 1 movl) $\Rightarrow$ 4000 accesos |
|              | (c) M[16384], página 2, MISS      | 2, 1, 0    |    |   |                                                                                                                                                                               |
| 1 (i = 512)  | (a) M[2048], página 0, HIT        | 0, 2, 1    |    |   |                                                                                                                                                                               |
|              | (b) M[10240], página 1, HIT + HIT | 1, 0, 2    |    |   | • iteración 0: 3 MISS y 1 HIT ; de la iteración 1 a la 4: 1 MISS y 15 HIT (cada 4 iteraciones se repetirá el patrón)                                                          |
|              | (c) M[18432], página 2, HIT       | 2, 1, 0    |    |   |                                                                                                                                                                               |
| 2 (i = 1024) | (a) M[4096], página 0, HIT        | 0, 2, 1    |    |   | • Fallos de TLB = $3 + \lfloor 999 / 4 \rfloor * 1 = 252 \Rightarrow$ Aciertos de TLB = 3748                                                                                  |
|              | (b) M[12288], página 1, HIT + HIT | 1, 0, 2    |    |   |                                                                                                                                                                               |
|              | (c) M[20480], página 2, HIT       | 2, 1, 0    | d) |   |                                                                                                                                                                               |
| 3 (i = 1536) | (a) M[6144], página 0, HIT        | 0, 2, 1    |    |   |                                                                                                                                                                               |
|              | (b) M[14336], página 1, HIT + HIT | 1, 0, 2    |    |   |                                                                                                                                                                               |
|              | (c) M[22528], página 2, HIT       | 2, 1, 0    |    |   |                                                                                                                                                                               |
| 4 (i = 2048) | (a) M[8192], página 1, HIT        | 1, 2, 0    |    |   | • Fallos de TLB = $500 * 3 = 1500$ ; Aciertos de TLB = $500 * 4 + 500 = 2500$                                                                                                 |
|              | (b) M[16384], página 2, HIT + HIT | 2, 1, 0    |    |   |                                                                                                                                                                               |
|              | (c) M[24576], página 3, MISS      | 3, 2, 1, 0 |    |   |                                                                                                                                                                               |

1000 iteraciones

En 500, 3 MISS y 1 HIT en cada una de ellas

En 500, 4 HIT en cada una de ellas

### Problema 2. Repaso Cache

Disponemos de un procesador de 16 bits con un bus de direcciones de 16 bits. Este procesador tiene una memoria cache de datos con las siguientes características:

- Tamaño de bloque = 16 bytes
- Asociatividad = 2 (reemplazo = LRU)
- Número de líneas = 8
- Política de escritura: write through + write no allocate

Suponiendo que el contenido de la cache es el siguiente:

| conjunto 0 |   | conjunto 1 |   | conjunto 2 |   | conjunto 3 |   |
|------------|---|------------|---|------------|---|------------|---|
| EC8        | 1 | EC5        | 1 | EC6        | 0 | EC7        | 1 |
| AB4        | 0 | libre      | 0 | AB2        | 1 | libre      | 0 |

Teniendo en cuenta que:

- En el contenido de la MC para simplificar hemos dejado el número de bloque de memoria en vez del tag.
- El bit a 1, en el contenido de la cache, indica que es la línea más recientemente referenciada.
- R-byte (lectura de 1 byte), R-word (lectura de 2 bytes), W-byte (escritura de 1 byte), W-word (escritura de 2 bytes).
- El tamaño de las lecturas (y escrituras) se ha de indicar en bytes.

Rellenad la siguiente tabla:

| Tipo   | @ en hex | Bloq de memoria | Conjunto de MC | Acierto / Fallo | Lectura de MP |      |        | Escritura en MP |      |        |
|--------|----------|-----------------|----------------|-----------------|---------------|------|--------|-----------------|------|--------|
|        |          |                 |                |                 | si / no       | @    | tamaño | si / no         | @    | tamaño |
| R byte | 8890     | 889             | 1              | Falb            | Si            | 8890 | 16     | No              |      |        |
| W word | EC51     | EC5             | 1              | Acerb           | No            |      |        | Si              | EC51 | 2      |
| W byte | EC62     | EC6             | 2              | Acerb           | No            |      |        | Si              | EC62 | 1      |
| W word | 23D3     | 23D             | 1              | Falb            | No            |      |        | Si              | 23D3 | 2      |
| W byte | ABA4     | ABA             | 2              | Falb            | Nb            |      |        | Si              | ABA4 | 1      |
| R word | ABA5     | ABA             | 2              | Falb            | Si            | ABA5 | 16     | Nb              |      |        |
| R byte | 23D6     | 23D             | 1              | Falb            | Si            | 23D6 | 16     | Nb              |      |        |
| W word | EC57     | EC5             | 1              | Acerb           | No            |      |        | Si              | EC57 | 2      |
| R byte | EC68     | EC6             | 2              | Acerb           | Nb            |      |        | Nb              |      |        |
| R word | 8899     | 889             | 1              | Falb            | Si            | 8890 | 16     | Nb              |      |        |

Indicad también cómo queda la cache después de realizar los 10 accesos a memoria:

| conjunto 0 |   | conjunto 1 |   | conjunto 2 |   | conjunto 3 |   |
|------------|---|------------|---|------------|---|------------|---|
| EC8        | 1 | EC5        | 0 | EC6        | 1 | EC7        | 1 |
| AB4        | 0 | 889        | 1 | ABA        | 0 | -          | 0 |

|              |            |              |               |            |
|--------------|------------|--------------|---------------|------------|
| 1) Miss de L | Conjunto 1 | 4) Miss de E | 8) Hit de E   | Conjunto 1 |
|              | EC5 0      |              |               | EC5 1      |
|              | 889 1      |              |               | 23D 0      |
| 2) Hit de E  | Conjunto 1 | 6) Miss de L | 9) Hit de L   | Conjunto 2 |
|              | EC5 1      |              |               | EC6 1      |
|              | 889 0      |              |               | ABA 0      |
| 3) Hit de E  | Conjunto 2 | 7) Miss de L | 10) Miss de L | Conjunto 1 |
|              | EC6 1      |              |               | EC5 0      |
|              | AB2 0      |              |               | 889 1      |

**Problema 10. Predicción de vía**

Una CPU funciona a un voltaje de 1,2 V y una frecuencia de 2GHz. Se ha determinado que esta CPU tiene una corriente de fugas de 3 A y que a pleno rendimiento tiene una carga capacitiva equivalente de 5 nF (nanoFaradios).

- a) **Calculad** la potencia media dinámica (debida a conmutación), la potencia media estática (debida a fugas) y la potencia media total.

Se desea integrar esta CPU con una memoria cache de datos 2-asociativa de 128 KB de capacidad y un tamaño de bloque de cache de 64 bytes. Las direcciones generadas por la CPU son de 48 bits. La corriente de fugas de la memoria RAM estática es de 3  $\mu$ A (microAmperios) por bit. La energía consumida durante un acceso a la memoria de etiquetas es de 5 nJ (nanoJoules) por vía y la consumida durante un acceso a la memoria de datos es de 25 nJ por vía.

- b) **Calculad** el numero de conjuntos, el de bloques de cache, el de vías y el de bloques por vía.  
 c) **Dibujad** una dirección indicando claramente los campos usados para seleccionar el byte dentro del bloque, seleccionar el conjunto de la cache y los bits usados como etiqueta.  
 d) **Calculad** el tamaño en bits de la memoria de datos y el de la memoria de etiquetas de una vía (por simplicidad ignoraremos el bit de validez y otros bits de control).  
 e) **Calculad** la potencia media estática (debida a fugas) de la cache.

Se desean comparar diversas implementaciones alternativas de cache de datos 2-asociativa: **paralela**, **serie**, y con **predictor de vía**. Para compararlas se usa un *benchmark* con  $4 \times 10^9$  instrucciones dinámicas que realiza  $10^9$  accesos de datos a memoria y  $2 \times 10^9$  operaciones aritméticas de punto flotante. Este *benchmark* tiene un 10% de fallos en la cache descrita anteriormente.

En la implementación **paralela**, se accede simultáneamente tanto a las memorias de etiquetas como las de datos de ambas vías. Un acceso a cache se realiza en 1 ciclo y la penalización media por fallo de cache es de 20 ciclos. El *benchmark* ejecutado con la implementación **paralela** de la cache ha tardado 5 segundos.

- f) **Calculad** los MFLOPS de la implementación **paralela**.  
 g) **Calculad** el CPI de la implementación **paralela** y el CPI que obtendríamos con una memoria ideal ( $CPI_{ideal}$ ) en donde todos los accesos tardan 1 ciclo.  
 h) **Calculad** la energía dinámica consumida por un acceso a la cache. Para simplificar asumiremos que todos los accesos consumen lo mismo sean acierto o fallo, la energía extra consumida en acceder a memoria principal en caso de fallo está fuera de los objetivos de este problema.  
 i) **Calculad** la potencia (dinámica) media consumida en acceder a la cache  
 j) **Calculad** la potencia media total (estática+dinámica) consumida por el sistema CPU-cache.  
 k) **Calculad** la energía total consumida para ejecutar el *benchmark* y la eficiencia en MFLOPS/Watt.

En la implementación **serie** un acceso tarda 2 ciclos. En el primer ciclo se accede a las memorias de etiquetas de ambas vías. Una vez determinada la vía que contiene el dato, en el segundo ciclo se accede solo a la memoria de datos de dicha vía. La penalización en caso de fallo sigue siendo de 20 ciclos ya que en el primer ciclo del acceso ya se puede determinar si es acierto o fallo. Obsérvese que respecto la implementación **paralela**, los aciertos tienen una penalización de 1 ciclo.

- l) **Calculad** el tiempo de ejecución y los MFLOPS de la implementación **serie**.  
 m) **Calculad** la energía consumida por un acceso a la cache.  
  
 n) **Calculad** la potencia (dinámica) media consumida en acceder a la cache  
 o) **Calculad** la potencia media total consumida por el sistema CPU-cache.  
 p) **Calculad** la energía total consumida para ejecutar el *benchmark* y la eficiencia en MFLOPS/Watt.

En la implementación con **predictor de vía**, este predice la vía probable en que se encuentra el dato y se accede solo a las memorias de etiquetas y de datos de esa vía. En caso de fallo del predictor hay que acceder a la otra vía (memorias de etiquetas y datos). El predictor usado consiste en una memoria de 8k x 1 bits indexado con los bits bajos del PC de las instrucciones de acceso a memoria. Este predictor tiene una tasa de aciertos del 80% del total de accesos a memoria y cada acceso al predictor consume 1nJ. En esta implementación se pueden dar las siguientes situaciones:

- El predictor acierta: se accede a 1 sola vía (datos+etiquetas) en 1 ciclo (no hay penalización) y al comprobar los tags se comprueba que es acierto de cache.
  - El predictor falla pero es acierto de cache: El acceso tarda 2 ciclos, en el primero se accede a la vía probable (aunque equivocada) al comprobar los tags se descubre que es fallo (de vía) y en el segundo se accede a la vía correcta y se descubre que es acierto de cache (1 ciclo de penalización).
  - El predictor falla y además es fallo de cache: En el primer ciclo se accede a la vía probable (aunque incorrecta), en el segundo ciclo se accede a la otra vía y se descubre que es fallo de cache (con lo que hay que acceder a memoria principal). Obsérvese que en este caso la penalización es de 21 ciclos ya que no se descubre el fallo de cache hasta que se accede a la 2a vía.
- q) ¿Puede darse al caso de que un acierto del predictor de vía sea fallo de cache? ¿porqué?  
 r) **Calculad** la potencia media estática (debida a fugas) del predictor y compárala con la de la cache (se calcula de la misma forma ya que se ha empleado el mismo tipo de memoria estática).  
 s) **Calculad** el tiempo de ejecución y los MFLOPS de la implementación con **predictor de vía**.  
 t) **Calculad** la energía consumida por un acceso en que el predictor acierta y uno en que el predictor falla (tener en cuenta la energía consumida por el acceso al predictor). Calcular también la energía media consumida por acceso.  
 u) **Calculad** la potencia (dinámica) media consumida en acceder a la cache  
 v) **Calculad** la potencia media total consumida por el sistema CPU-cache (acuérdate de las fugas del predictor).  
 w) **Calculad** la energía total consumida para ejecutar el *benchmark* y la eficiencia en MFLOPS/Watt.  
 x) **Calculad** la ganancia en eficiencia energética de la implementación serie sobre la paralela y la de predicción de vía sobre la serie.

- a) **Calculad la potencia media dinámica (debida a conmutación), la potencia media estática (debida a fugas) y la potencia media total.**

$$P_{\text{conm}} = CV^2F = 5 \times 10^{-9} F * (1,2V)^2 * 2 \times 10^9 \text{Hz} = 14,4 \text{ W}$$

$$P_{\text{fuga}} = IV = 3A * 1,2 \text{ V} = 3,6 \text{ W}$$

$$P_{\text{total}} = 14,4 \text{W} + 3,6 \text{W} = 18 \text{W}$$

- b) **Calculad el numero de conjuntos, el de bloques de cache, el de vías y el de bloques por vía.**

$$2^{17} \text{ bytes} / 2^6 \text{bytes/bloque} = 2^{11} \text{ bloques} = 2048 \text{ bloques}$$

$$2048 \text{ bloques} / 2 \text{ bloques/cjto} = 1024 \text{ cjtos}$$

2-asociativa -> 2 vias

1024 cjtos -> 1024 bloques/via

- c) **Dibujad una dirección indicando claramente los campos usados para seleccionar el byte dentro del bloque, seleccionar el conjunto de la cache y los bits usados como etiqueta.**

| TAG (32) | Cjto (10) | byte (6) |

- d) **Calculad el tamaño en bits de la memoria de datos y el de la memoria de etiquetas de una vía.**

1 via -> 1024 bloques

$$M_{\text{datos}} = 1024 \text{ bloq} * 64 \text{ bytes/bloq} * 8 \text{bits/byte} = 524288 \text{ bits}$$

$$M_{\text{etiq}} = 1024 \text{ bloq} * 32 \text{ bits/bloq} = 32768 \text{ bits}$$

- e) **Calculad la potencia media estática (debida a fugas) de la cache. bits totales = 557056 bits**

$$I_{\text{fuga}} = 557056 \text{ bits} * 3 \times 10^{-6} \text{ A/bit} = 1,671 \text{ A}$$

$$P_{\text{fuga}} = I * V * \# \text{vias} = 1,671 \text{ A} * 1,2 \text{ V} * 2 \text{ vias} = 4 \text{ W}$$

- f) **Calculad los MFLOPS de la implementación paralela.**

$$Mflops = 2 * 10^9 \text{ flop} / 5 \text{ s} * 10^{-6} \text{ MFLOPS/flop} = 400 \text{ MFLOPS}$$

- g) **Calculad el CPI de la implementación paralela y el CPI que obtendríamos con una memoria ideal ( $CPI_{\text{ideal}}$ ) en donde todos los accesos tardan 1 ciclo.**

$$\text{Ciclos totales} = 5 \text{ s} * 2 \times 10^9 \text{ Hz} = 10 \times 10^9 \text{ ciclos CPI}$$

$$\text{paralela} = 10 \times 10^9 \text{ ciclos} / 4 \times 10^9 \text{ instr} = 2,5 \text{ c/i}$$

$$\text{Cilos perdidos mem} = 0,1 \text{ fallos/acceso} * 10^9 \text{ accesos} * 20 \text{ ciclos/fallo} = 2 \times 10^9 \text{ ciclos}$$

$$\text{Ciclos ideal} = 10 \times 10^9 \text{ ciclos} - 2 \times 10^9 \text{ ciclos} = 8 \times 10^9 \text{ ciclos}$$

$$CPI_{\text{ideal}} = 8 \times 10^9 \text{ ciclos} / 4 \times 10^9 \text{ instr} = 2 \text{ c/i}$$

- h) **Calculad la energía dinámica consumida por un acceso a la cache. Para simplificar asumiremos que todos los accesos consumen lo mismo sean acierto o fallo, la energía extra consumida en acceder a memoria principal en caso de fallo está fuera de los objetivos de este problema.**

1 acceso cache -> 2 vias etiquetas + 2 vias datos

$$\text{Energia} = 2 * 5 \text{ nJ} + 2 * 25 \text{ nJ} = 60 \text{ nJ}$$

- i) **Calculad la potencia (dinámica) media consumida en acceder a la cache**

$$10^9 \text{ accesos} / 5 \text{ segundos} = 0,2 \times 10^9 \text{ accesos/s}$$

$$\text{Potencia} = \text{energia} / t = 0,2 \times 10^9 \text{ accesos/s} * 60 \times 10^{-9} \text{ J} = 12 \text{ J/s} = 12 \text{ W}$$

- j) **Calculad la potencia media total (estática+dinámica) consumida por el sistema CPU-cache.**

$$P_{\text{total}} = P_{\text{CPU}} + P_{\text{cache\_fugas}} + P_{\text{cache\_conmut}} = 18 \text{ W} + 4 \text{ W} + 12 \text{ W} = 34 \text{ W}$$

- k) **Calculad la energía total consumida para ejecutar el benchmark y la eficiencia en MFLOPS/Watt.**

$$E = P * t = 34 \text{ W} * 5 \text{ s} = 170 \text{ J}$$

$$\text{Eficiencia} = 400 \text{ MFLOPS} / 34 \text{ W} = 11,76 \text{ MFLOPS/W}$$

- l) **Calculad el tiempo de ejecución y los MFLOPS de la implementación serie.**

$$\text{Ciclos} = \text{ciclos ideal} + \text{ciclos perdidos fallos} + \text{ciclos perdidos aciertos}$$

$$= 10 \times 10^9 \text{ ciclos} + 0,9 \text{ aciertos/acceso} * 10^9 \text{ accesos} * 1 \text{ ciclos/acierto} = 10,9 * 10^9 \text{ ciclos}$$

$$\text{Texe} = 10,9 * 10^9 \text{ ciclos} / 2 \times 10^9 \text{ Hz} = 5,45 \text{ s}$$

$$Mflops = 2 * 10^9 \text{ flop} / 5,45 \text{ s} * 10^{-6} \text{ MFLOPS/flop} = 367 \text{ MFLOPS}$$

- m) **Calculad la energía consumida por un acceso a la cache.**

1 acceso -> 2 vias etiquetas + 1 via datos

$$E = 2 * 5 \text{ nJ} + 25 \text{ nJ} = 35 \text{ nJ} \quad (\text{en los fallos nos podríamos ahorrar el acceso a datos, pero lo damos por bueno})$$

- n) **Calculad la potencia (dinámica) media consumida en acceder a la cache**

$$10^9 \text{ accesos} / 5,45 \text{ segundos} = 0,183 \times 10^9 \text{ accesos/s}$$

$$\text{Potencia} = \text{energia} / t = 0,183 \times 10^9 \text{ accesos/s} * 35 \times 10^{-9} \text{ J} = 6,42 \text{ J/s} = 6,42 \text{ W}$$

- o) **Calculad la potencia media total consumida por el sistema CPU-cache.**

$$P_{\text{total}} = P_{\text{CPU}} + P_{\text{cache\_fugas}} + P_{\text{cache\_conmut}} = 18 \text{ W} + 4 \text{ W} + 6,42 \text{ W} = 28,42 \text{ W}$$

- p) **Calculad la energía total consumida para ejecutar el benchmark y la eficiencia en MFLOPS/Watt.**

$$E = P * t = 28,42 \text{ W} * 5,45 \text{ s} = 155 \text{ J}$$

$$\text{Eficiencia} = 367 \text{ MFLOPS} / 28,42 \text{ W} = 12,91 \text{ MFLOPS/W}$$

- q) **¿Puede darse al caso de que un acierto del predictor de vía sea fallo de cache? ¿porqué?**

- r) Calculad la potencia media estática (debida a fugas) del predictor y compárala con la de la cache (se calcula de la misma forma ya que se ha empleado el mismo tipo de memoria estática).

$$I_{fuga} = 8192 \text{ bits} * 3 \times 10^{-6} \text{ A/bit} = 24,6 \text{ mA P}$$

$$fuga = I * V = 24,6 \times 10^{-3} \text{ A} * 1,2 \text{ V} = 29,5 \text{ mW}$$

(es mucho menor, miliwatos vs watos)

- s) Calculad el tiempo de ejecución y los MFLOPS de la implementación con predictor de vía.

$$\begin{aligned} \text{Ciclos} &= \text{ciclos ideal} + \text{ciclos perdidos fallos cache} + \text{ciclos perdidos fallo predictor} \\ &= 10 \times 10^9 \text{ ciclos} + 0,2 \text{ aciertos/acceso} * 10^9 \text{ accesos} * 1 \text{ ciclos/acierto} = 10,2 * 10^9 \text{ ciclos} \end{aligned}$$

$$T_{exe} = 10,2 * 10^9 \text{ ciclos} / 2 \times 10^9 \text{ Hz} = 5,1 \text{ s}$$

$$Mflops = 2 * 10^9 \text{ flop} / 5,1 \text{ s} * 10^{-6} \text{ MFLOPS/flop} = 392 \text{ MFLOPS}$$

- t) Calculad la energía consumida por un acceso en que el predictor acierta y uno en que el predictor falla (tener en cuenta la energía consumida por el acceso al predictor). Calcular también la energía media consumida por acceso.

acierto predictor -> predictor + 1 vía etiquetas + 1 vía datos

fallo predictor -> predictor + 2 vías etiquetas + 2 vías datos

$$E_{acierto} = 1 \text{ nJ} + 5 \text{nJ} + 25 \text{ nJ} = 31 \text{ nJ} \quad E_{fallo} = 1 \text{nJ} + 10 \text{nJ} + 50 \text{nJ} = 61 \text{ nJ}$$

$$E_{media} = 0,8 * 31 \text{ nJ} + 0,2 * 61 \text{ nJ} = 37 \text{ nJ}$$

- u) Calculad la potencia (dinámica) media consumida en acceder a la cache  $10^9$  accesos / 5,1 segundos =  $0,196 \times 10^9$  accesos/s

$$\text{Potencia} = \text{energía/t} = 0,196 \times 10^9 \text{ accesos/s} * 37 \times 10^{-9} \text{ J} = 7,25 \text{ J/s} = 7,25 \text{ W}$$

- v) Calculad la potencia media total consumida por el sistema CPU-cache (acuérdate de las fugas del predictor).  $P_{total} = P_{CPU} + P_{cache\_fugas} + P_{pred\_fugas} + P_{cache\&pred\_conmut}$   
 $= 18 \text{ W} + 4 \text{ W} + 0,03 \text{ W} + 7,25 \text{ W} = 29,28 \text{ W}$

- w) Calculad la energía total consumida para ejecutar el benchmark y la eficiencia en MFLOPS/Watt.

$$E = P*t = 29,28 \text{ W} * 5,1 \text{ s} = 149 \text{ J}$$

$$\text{Eficiencia} = 392 \text{ MFLOPS} / 29,28 \text{ W} = 13,39 \text{ MFLOPS/W}$$

- x) Calculad la ganancia en eficiencia energética de la implementación serie sobre la paralela y la de predicción de vía sobre la serie.

$$\text{serie/paralelo} = 12,91 / 11,76 = 1,098 = 9,8\%$$

$$P_{via}/\text{serie} = 13,39 / 12,91 = 1,037 = 3,7\%$$

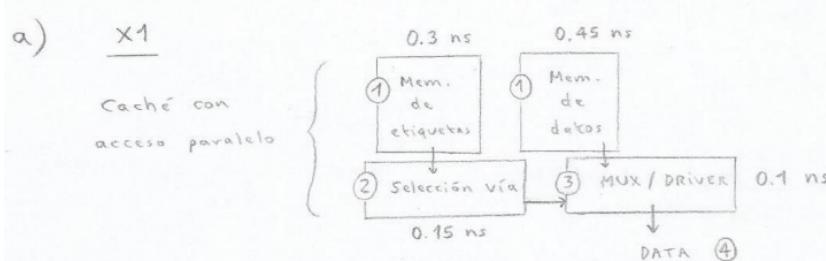
### Problema 11. Caches segmentadas

En un procesador X el camino crítico, y por tanto el tiempo de ciclo, está limitado por la memoria cache de datos. El tiempo de los componentes de la memoria cache de datos se desglosa de la siguiente forma:

| Componente                                       | Tiempo  |
|--------------------------------------------------|---------|
| Memoria de etiquetas                             | 0,30 ns |
| Selección de vía                                 | 0,15 ns |
| Memoria de datos                                 | 0,45 ns |
| Mux/Driver de datos                              | 0,10 ns |
| Registro de desacoplamiento (en caso que se use) | 0,05 ns |

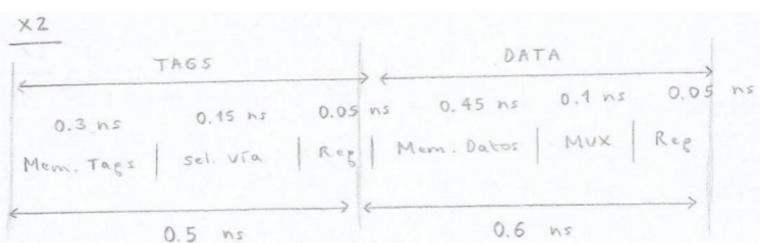
Queremos evaluar el rendimiento de 4 posibles implementaciones de la memoria cache para el procesador X:

- Procesador **X1**: La cache de datos tiene una implementación paralela y el tiempo de acceso a la cache es de 1 ciclo de procesador
  - Procesador **X2**: La cache de datos está segmentada en 2 etapas y el tiempo de acceso a la cache es de 2 ciclos de procesador
  - Procesador **X3**: La cache de datos está segmentada en 3 etapas y el tiempo de acceso a la cache es de 3 ciclos de procesador
  - Procesador **X4**: La cache de datos está segmentada en 4 etapas y el tiempo de acceso a la cache es de 4 ciclos de procesador
- a) **Calculad** el tiempo de ciclo de la cache de datos y el tiempo total de un acceso para los procesadores X1, X2, X3 y X4, usando la distribución mas adecuada de los componentes por etapas.
- b) **Razonad** porque descartamos las opciones X2 y X4 para el resto del problema
- c) **Calculad** la frecuencia de reloj de los procesadores X1 y X3
- d) **Calculad** el CPI del programa P para los procesadores X1 y X3 suponiendo que nunca hay fallos en la cache de datos.
- e) **Calculad** el speedup de X3 sobre X1 en % suponiendo que nunca hay fallos en la cache de datos
- Sabemos que el programa P tiene un 10% de fallos en la cache de datos utilizada y que el tiempo de penalización en ambos casos es de 60 ciclos.
- f) **Calculad** el speedup real de X3 sobre X1 en % teniendo en cuenta la jerarquía de memoria completa.



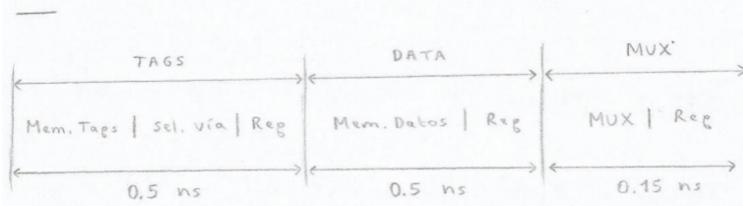
$$t_c = 0.45 \text{ ns} + 0.1 \text{ ns} = 0.55 \text{ ns}$$

$$t_{\text{acceso}} = 0.55 \text{ ns}$$



$$t_c = 0.6 \text{ ns}$$

$$t_{\text{acceso}} = 0.6 \times 2 = 1.2 \text{ ns}$$

X3

$$t_c = 0.5 \text{ ns}$$

$$t_{\text{acceso}} = 0.5 \times 3 = 1.5 \text{ ns}$$

b) -  $x_2$  tiene el peor  $t_c$

-  $x_4$  tiene el peor  $t_{\text{acceso}}$

c)  $f_{x1} = \frac{1}{t_{c_{x1}}} = 1.82 \text{ GHz}$        $f_{x3} = \frac{1}{t_{c_{x3}}} = 2 \text{ GHz}$

d)  $2 \cdot 10^9$  instrucciones

- $\rightarrow 1.2 \cdot 10^9$  aritméticas, 5 ciclos cada una (60%)
- $\rightarrow 400 \cdot 10^6$  de salto, 4 ciclos cada una (20%)
- $\rightarrow 400 \cdot 10^6$  de acceso a memoria, 4 ciclos + ciclos del acceso a la caché

x1 : 1 ciclo  
x3 : 3 ciclos

$$CPI_{x1} = 0.6 \times 5 + 0.2 \times 4 + 0.2(4+1) = 4.8 \text{ c/i}$$

$$CPI_{x3} = 0.6 \times 5 + 0.2 \times 4 + 0.2(4+3) = 5.2 \text{ c/i}$$

e)  $T_{\text{exe } x1} = 2 \cdot 10^9 \times 4.8 \times 0.55 \cdot 10^{-9} = 5.28 \text{ s}$

$$T_{\text{exe } x3} = 2 \cdot 10^9 \times 5.2 \times 0.5 \cdot 10^{-9} = 5.2 \text{ s}$$

$$\text{Speedup} = \frac{5.28}{5.2} = 1.015 \text{ (1.5 %)}$$

f)  $CPI = CPI_{\text{ideal}} + CPI_{\text{mem}} = 4.8 + 0.2 \times 0.1 \times 60$   
 $= 6 \text{ c/i}$

$$T_{\text{exe } x1} = 2 \cdot 10^9 \times 6 \times 0.55 \cdot 10^{-9} = 6.6 \text{ s}$$

$$CPI = 5.2 + 0.2 \times 0.1 \times 60 = 6.4 \text{ c/i}$$

$$T_{\text{exe } x3} = 2 \cdot 10^9 \times 6.4 \times 0.5 \cdot 10^{-9} = 6.4 \text{ s}$$

$$\text{Speedup} = \frac{6.6}{6.4} = 1.03125 \text{ (3.125 %)}$$