

$$1.12 = \text{a)} \quad \frac{1}{\text{MTTF}} = \sum_i^n \frac{1}{\text{MTTF}_i} = \frac{1}{\text{MTTF}_{\text{PW}}} + \frac{1}{\text{MTTF}_{\text{CPU}}} + \frac{1}{\text{MTTF}_{\text{PB}}} + \frac{4}{\text{MTTF}_{\text{DIMM}}} + \frac{1}{\text{MTTF}_{\text{GPU}}} + \frac{8}{\text{MTTF}_{\text{HDD}}} =$$

$$\frac{1}{1,25 \cdot 10^5} + \frac{1}{10^6} + \frac{1}{2 \cdot 10^5} + \frac{4}{10^6} + \frac{1}{5 \cdot 10^5} + \frac{8}{10^5} = \frac{1}{100000} \rightarrow \text{MTTF} = \underline{\underline{100000 \text{ h}}}$$

b) $\text{MTTR} = 20 \text{ h}$

$$\text{MTBF} = \text{MTTF} + \text{MTTR} = \underline{\underline{10.020 \text{ h}}}$$

c) Availability = $\frac{\text{MTTF}}{\text{MTBF}} = \frac{10000}{10020} = 0.998 \Rightarrow \underline{\underline{99.8\%}}$

2.1-

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
$\sim x \sim y$	1111 1101	0xFD	$!x !y$	0000 0000	0x00
$x \& !y$	0000 0000	0x00	$x \&\& \sim y$	0000 0001	0x01

2.2-

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		$x \ll 4$		$x >> 3$ (lógico)		$x >> 3$ (aritmético)	
hex	binario	hex	binario	hex	binario	hex	binario
0xF0	1111 0000	0x00	0000 0000	0x3C	0011 1100	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

2.5- $\text{MOVL } \$A, \%eax$

loop: $\text{MOVL } (\%eax), \%ebx$

$\text{MOVL } (\$tabla, \%ebx), \%ecx$

$\text{MOVL } \%ecx, (\%eax)$

$\text{ADDL } \$1, \%eax$

$\text{CMPL } \%eax, 256(\$A)$

JNE loop

$\text{eax} \leftarrow \& A[0]$

$\text{ebx} \leftarrow A[i]$

$\text{ecx} \leftarrow \text{Tabla}[\text{ebx}]$

$A[i] \leftarrow \text{ecx}$

$++i$

if($\&A[i] \neq \&A[zsg]$) {

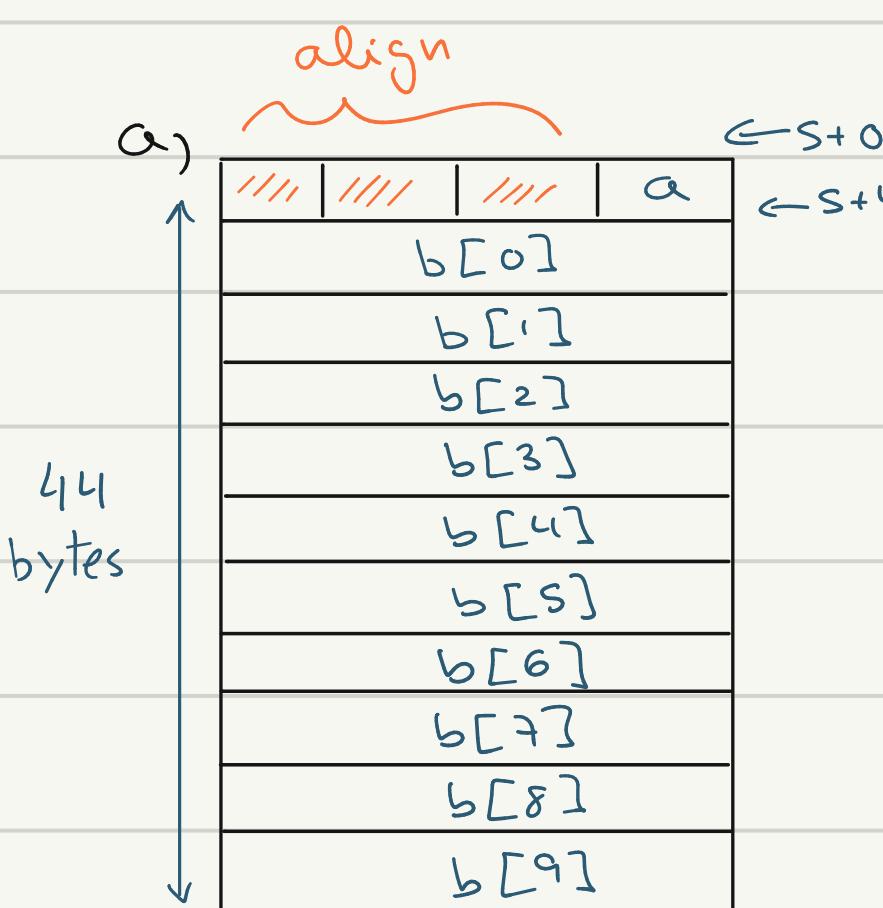
} → loop }

2.6.- MOVL 8(%ebp), %eax
MOVL 12(%ebp), %ebx
CMPL \$-10, %eax
JLE else
CMPL \$10, %eax
JGE else
if: MOVL %eax ,(%ebx) JMP end
else: LEAL %eax, %ebx
end: RET

Guardo i en eax
Guardo x en ebx
if ($i \leq -10$) \rightarrow else
if ($i \geq 10$) \rightarrow else
*x = i
x = &i

2.9.- `typedef struct {`

```
    |   char a;      1 byte
    |   int b[10];  4.10bytes } Hace falta
    |                           } alinear
  } elem;
elem s [100];
```



b) $\text{@s[i].b[j]} = \text{@s} + 44 \cdot i + 4(j+1)$

c) Para simplificar: $44i + 4(j+1) \Rightarrow 4(11i + j + 1)$

`imul $11, %esi, %eax`

`addl %edi, %eax`

`incl %eax`

`movl (%ebx,%eax,4), %eax`

`movb (%ebx,%eax), %dl`

2.10.- a)

REGs
i
Suma
Fila
%ebp
@RET
@M
m
n

b) `pushl %ebp`
`movl %esp, %ebp`
`subl $12, %esp`
`pushl %ebx`
`movl $0, -8(%ebp)` ← suma = 0
`movl $0, -4(%ebp)` ← fila = 0
`movl 12(%ebp), %ebx` ← i = m
`for: cmpl 16(%ebp), %ebx` ← i < n
`jge finfor`
`movl -4(%ebp), %ecx`
`leal %ecx, %eax`
`pushl %eax`
`imul $10, %ecx` ← 10 * fila

`addl %ebx, %ecx` ← 10 * fila + i
`move 8(%ebp), %edx`
`movl (%edx,%ecx,4), %ecx` ← mandar M[Fila][i]
`pushl %ecx`
`call Normaliza`
`addl $8, %esp`
`addl %eax, -8(%ebp)`
`incl %ebx`
`jmp for`
`finfor: movl -8(%ebp), %eax`
`incl %eax`
`popl %ebx`
`movl %ebp, %esp`
`popl %ebp`
`ret`

} Suma += Normaliza
 i++
 Suma + 1

2.14- a)

d[0]	-404 Not found
d[1]	400
...	
d[99]	-4
aux	
%ebp	
@RET	+8
a	+12
@b	+16
c	

b) leal -4(%ebp), %eax

pushl %eax

leal -404(%ebp), %eax

pushl %eax

pushl \$0

call examen

#

&aux

&d

&0

c) movl 0, -4(%ebp)

for: movl -4(%ebp), %eax

cmpl \$100, %eax

jge finfor

leal -404(%ebp), %ecx

movl (%ecx,%eax,4), %ecx

movl 12(%ebp), %edx

movl %ecx, (%edx,%eax,4)

incl %eax

jmp for

finfor: ...

aux=0

if(aux < 100)

{ d[aux]

{ b[aux]

aux++

d) pushl 1b(%ebp) # c

pushl 12(%ebp) # b

pushl 8(%ebp) # a

call examen

2.18.- a) $M = 5 \leftarrow @Matz + (\underline{5}i \cdot j) \cdot 4$
 $N = 7 \leftarrow @Mat. + (\underline{7}i \cdot j) \cdot 4$

b) 13

$$e) \text{ ciclos} = \frac{10}{0.5} + \frac{3}{0.8} = 23,75 \approx 24 \text{ ciclos}$$

c) 13

d) 10

$$f) \frac{10}{0.6} + \frac{3}{0.8} = 20,4 \text{ ciclos}$$

$$\text{Speed Up} = \frac{23.75}{20.4} = \underline{\underline{1.2x}}$$

2.19.- a)

i1	← +0
c[3]	← +4
c[2]	← +8
...	
///	← +36
///	← +40
i3	
tabla[0]	
...	
tabla[99]	
n	

← +0
 ← +4
 ← +8
 ...
 ← +36
 ← +40

b)

i	← -48
j	← -44
aux	← -40
%ebp	← -36
@ret	← -32
@P1	← -28
@x	← -24
y	← -20

c) movl 12(%ebp), %eax
 movl (%eax), %eax
 addl -4(%ebp), %eax
 movl %ebp, %esp
 popl %ebp
 ret

d) pushl 16(%ebp)
 movl -44(%ebp), %eax
 movl 8(%ebp), %ecx
 leal (%ecx, %eax, 40), %eax
 pushl %eax
 call T
 movl %eax, -40(%ebp)

e) movl -44(%ebp), %eax
 movl 16(%ebp), %ecx
 imull %eax, %ecx, -48(%ebp)

f) movb -13(%ebp), %eax
 leal -36(%ebp), %ecx
 addl -48(%ebp), %ecx
 movb %eax, (%ecx)

g) xorl %eax, %eax
 for: cmpl 20(%ebp), %eax
 jle endfor
 movl 8(%ebp), %ecx
 cmpl 4000(%ecx), %eax
 jle endfor
 movl 36(%ecx, %eax, 40), %edx
 addl %eax, %edx
 movl (%ecx, %eax, 40), %edx
 addl \$5, %eax
 jmp for
 endfor:

h) movl 16(%ebp), %eax
 cmpl -40(%ebp), %eax
 je else
 if: movl -48(%ebp), %ecx
 movl %ecx, -4(%ebp)
 jmp endif
 else: movl -44(%ebp), %ecx
 movl %ecx, -4(%ebp)

i) xorl %eax, %eax
 movl -36(%ebp, %eax), %ecx
 while: cmpb \$46, %ecx
 je endwhile
 movl \$35, (%ecx)
 incl %eax
 jmp while
 endwhile:

3.4.-

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.

$$t_{ma_1} = 0.2 \cdot 80 + 0.8 \cdot (0.9 \cdot 10 + 0.1 \cdot (10 + 100 + 10)) = 32.8 \text{ ns}$$

$\hookrightarrow 32800 \text{ ns}$
·1000
accesos

$$t_{ma_2} = (0.33 \cdot (10 + 100 + 100 + 10) + 0.66 \cdot (10 + 100 + 10)) \cdot 0.15 + 0.85 \cdot 10 = 31.3 \text{ ns}$$

$\hookrightarrow 31300 \text{ ns}$
·1000
accesos

b)

Indicad qué alternativa sería la más rápida para un programa que sólo realizará lecturas.

El primero porque tiene mayor tasa de aciertos. La segunda opción tarda más en fallo de bloque sucio.

c)

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

la palabra ocupa menos que el bloque.

Tenim una CPU amb les següents característiques:

3.5.-

- 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
Número de referències a memòria per instrucció (nr)	1 ref/inst	0.6 ref/inst
Taxa de fallades (m)	4 %	10 %
Penalització (Tpfi) al reemplaçar un bloc no modificat	10 cicles	15 cicles
Penalització (Tpfd) 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?

TmaI // Tsa + m · Tpd
TmaD // Tsa + m · Tpf
Tma // TmaI + TmaD

$$a) t_{maI} = 0.96 \cdot 1 + 0.04 \cdot (1+10) = 0.96 + 0.44 = 1.4 \text{ cicles}$$

$$d) CPI_{mem} = nr \cdot (Tma - Tsa) \\ 1.6 \cdot (1.85 - 1) = 1.36$$

$$b) t_{maD} = 0.9 \cdot 1 + 0.1 \cdot (0.2 \cdot (1+20) + 0.8 \cdot (1+15)) = 2.6 \text{ cicles}$$

$$CPI = CPI_{in} + CPI_{mem} = 1.5 + 1.36 = 2.86$$

$$c) t_{ma} = \frac{n^o \text{ Inst.} \cdot t_{maI} + n^o \text{ Dad.} \cdot t_{maD}}{n^o \text{ total}} = \frac{1 \cdot 1.4 + 0.6 \cdot 2.6}{1.6} = 1.85 \text{ cicles}$$

$$\begin{aligned} T_{exec} &= N \cdot CPI \cdot T_c = 1 \cdot 2.86 \cdot 10 = \\ &= 28.6 \text{ ns} \end{aligned}$$

3.7.- 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

b) 1000 iteraciones

c) 1ra → falla 3 veces. } $3 + \frac{999}{4} \approx 252$ miss
 demás → 1 de cada 4 falla } $4000 \text{ accesos} - 252 \text{ miss} = 3748 \text{ aciertos}$

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

g) 1000 iteraciones

Si En la mitad fallamos 3 de cada 4, y en la otra mitad no fallamos, por tanto:

$$\text{Fallos} = 500 \cdot 3 = 1500 \quad \text{Aciertos} = 4000 - 1500 = 2500$$

3.2.-

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

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
ABA	0	889	1	ABA	0	//	0

$$10-\alpha) P_c = C \cdot V^2 \cdot f = 5 \cdot 10^{-9} \cdot 2 \cdot 10^9 \cdot 1,2^2 = 14,4 \text{ W}$$

$$P_g = I \cdot V = 3 \cdot 1,2 = 3,6 \text{ W}$$

$$P_M = \frac{P_c + P_g}{2} = 14,4 + 3,6 = 18 \text{ W}$$

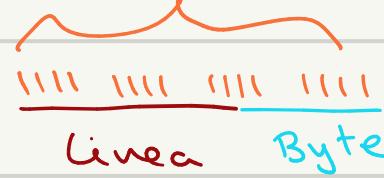
$$b) N_{conj} = \frac{\text{Capacidad}}{\text{Tamaño}} \cdot \text{vies} = \frac{128 \cdot 1024}{64} \cdot 2 = 1024 \text{ conj}$$

$$N_{bloques} = N_{conj} \cdot \text{vies} = 1024 \cdot 2 = 2048 \text{ bloques}$$

Nvias = 2, la cache es 2-asociativa.

$$\frac{N_{bloques}}{\text{vía}} < \frac{N_{bloques}}{N_{vias}} = \frac{2000}{2} = 1024 \text{ bloques/vía}$$

$$c) \begin{array}{c} 0x\text{FFFF FFFF FFFF} \\ \text{TAG} \end{array}$$



$$d) T_E = 1024 \cdot 8 \cdot 4 = 32768 \text{ bits}$$

$$T_D = 1024 \cdot 64 \cdot 8 = 524288 \text{ bits}$$

$$e) P_g = I \cdot V = (3 \cdot 10^{-6} \cdot 557056) \cdot 1,2 \cdot 2 = 41 \text{ W}$$

$$f) \text{MFLOPS} = \frac{\text{NMFO}}{t} = \frac{2 \cdot 10^3}{5} = 400 \text{ MFLOPS}$$

$$g) CPI_p = \frac{N_c}{N_s} = \frac{5 \cdot 2 \cdot 10^9}{4 \cdot 10^9} = 2,5 \text{ c/i}$$

$$CPI_{\perp} = \frac{N_c}{N_s} = \frac{10 \cdot 10^9 - (0,1 \cdot 10^9 \cdot 20)}{4 \cdot 10^9} = 2 \text{ c/i}$$

$$h) E_A = E_D + E_E \cdot N_v = (25 + 5) \cdot 10^{-9} \cdot 2 = 60 \text{ nJ}$$

$$i) NA/s = \frac{10^9}{5} = 0,2 \cdot 10^9 \text{ accesos/s}$$

$$P = \frac{E}{t} = 0,2 \cdot 10^9 \cdot 60 \text{ nJ} = 12 \text{ W}$$

$$j) P_T = 18 + 12 + 4 = 34 \text{ W}$$

$$k) E_C = P_T \cdot t = 34 \cdot 5 = 170 \text{ J}$$

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

$$l) \text{Cycles} = C_I + C_H + C_U = 8 \cdot 10^9 + 2 \cdot 10^9 + (0,9 \cdot 10^9) = 10,9 \cdot 10^9 \text{ ciclos}$$

$$T_{exe} = \frac{\text{Cycles}}{g} = \frac{10,9 \cdot 10^9}{2 \cdot 10^9} = 5,45 \text{ s}$$

$$\text{MFLOPS} = \frac{\text{NMFO}}{T_{exe}} = \frac{2 \cdot 10^3}{5,45} = 367 \text{ MFLOPS}$$

$$m) E_A = 2 \cdot E_I + E_D = 2 \cdot 5 + 25 = 35 \text{ nJ}$$

$$n) NA/s = \frac{10^9}{545} = 0,183 \cdot 10^9$$

$$P = \frac{E}{t} = 0,183 \cdot 10^9 \cdot 35 = 6,41 \text{ W}$$

$$o) P_T = 18 + 4 + 6,41 = 28,41 \text{ W}$$

$$p) E_C = P_T \cdot t = 28,41 \cdot 5,45 = 154,83 \text{ J}$$

$$Efic = \frac{367}{28,41} = 12,9 \text{ MFLOPS/W}$$

q) No, si el predictor acierta la cache no puede fallar.

$$r) P = I \cdot V = (3 \cdot 10^{-6} \cdot 8000) \cdot 1,2 = 0,029 \text{ W}$$

Es mucho menor $t = \frac{4}{0,029} = 138 \text{ veces}$

$$s) \text{Cicles} = C_I + C_H + C_U = 10 \cdot 10^9 + (1 \cdot 0,2 \cdot 10^9) = 10,2 \cdot 10^9 \text{ ciclos}$$

$$T_{exe} = \frac{\text{Cicles}}{g} = \frac{10,2 \cdot 10^9}{2 \cdot 10^9} = 5,1 \text{ s}$$

$$\text{MFLOPS} = \frac{\text{NMFO}}{s} = \frac{2 \cdot 10^3}{5,1} = 392 \text{ MFLOPS}$$

$$t) E_H = E_A + E_P = 30 + 1 = 31 \text{ nJ}$$

$$E_M = 2 \cdot E_A + E_P = 60 + 1 = 61 \text{ nJ}$$

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

$$u) NA/s = \frac{10^9}{5,1} = 0,196 \cdot 10^9$$

$$P = \frac{E}{t} = 0,196 \cdot 10^9 \cdot 37 = 7,252 \text{ W}$$

$$v) P_T = 18 + 4 + 0,029 + 7,252 = 29,28 \text{ W}$$

$$w) E_C = P_T \cdot t = 29,28 \cdot 5,1 = 149,33 \text{ J}$$

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

$$x) g = \frac{Efic_1}{Efic_2} = \frac{13,39}{12,9} = 1,038 \quad 3,8\%$$

11- a) X₁) paralela 1 ciclo

$$T_c = \max(E_{\text{trig}}, \text{vicio}, \text{datos}) + M_{\text{ux}} = \max(0.45, 0.05) + 0.1 = 0.55 \text{ ns}$$

$$T_A = \text{ciclos} \cdot T_c = 1 \cdot 0.55 = 0.55 \text{ s}$$

X₂) 2 etapas 2 ciclos

$$T_c = \max(E_{\text{etapa } 1}, E_{\text{etapa } 2}) = \max(0.5, 0.6) = 0.6 \text{ ns}$$

$$T_{\text{etapa } 1} = T_{\text{trig}} + T_{\text{vicio}} + T_{\text{reg}} = 0.3 + 0.15 + 0.05 = 0.5$$

$$T_{\text{etapa } 2} = T_{\text{datos}} + T_{\text{muv}} + T_{\text{reg}} = 0.45 + 0.1 + 0.05 = 0.6$$

$$T_A = \text{ciclos} \cdot T_c = 2 \cdot 0.6 = 1.2 \text{ ns}$$

X₃) 3 etapas 3 ciclos

$$T_c = \max(E_{\text{etapa } 1}, E_{\text{etapa } 2}, E_{\text{etapa } 3}) = \max(0.5, 0.5, 0.15) = 0.5 \text{ ns}$$

$$T_{\text{etapa } 1} = T_{\text{trig}} + T_{\text{vicio}} + T_{\text{reg}} = 0.3 + 0.15 + 0.05 = 0.5$$

$$T_{\text{etapa } 2} = T_{\text{datos}} + T_{\text{reg}} = 0.45 + 0.05 = 0.5$$

$$T_{\text{etapa } 3} = T_{\text{muv}} + T_{\text{reg}} = 0.10 + 0.05 = 0.15$$

$$T_A = \text{ciclos} \cdot T_c = 3 \cdot 0.5 = 1.5 \text{ ns}$$

X₄) 4 etapas 4 ciclos

$$T_c = \max(T_{\text{etapa } 1}, T_{\text{etapa } 2}, T_{\text{etapa } 3}, T_{\text{etapa } 4}) = \max(0.35, 0.20, 0.5, 0.15) = 0.5$$

$$T_{\text{etapa } 1} = T_{\text{trig}} + T_{\text{reg}} = 0.3 + 0.05 = 0.35$$

$$T_{\text{etapa } 2} = T_{\text{vicio}} + T_{\text{reg}} = 0.15 + 0.05 = 0.20$$

$$T_{\text{etapa } 3} = T_{\text{datos}} + T_{\text{reg}} = 0.45 + 0.05 = 0.5$$

$$T_{\text{etapa } 4} = T_{\text{muv}} + T_{\text{reg}} = 0.10 + 0.05 = 0.15$$

$$T_A = \text{ciclos} \cdot T_c = 4 \cdot 0.5 = 2 \text{ ns}$$

b) X₂ tiene un $T_c = 0.6$, mayor que ningún otro, y X₄ tiene un $T_A = 2$, mayor que ningún otro.

$$\int x_1 = \frac{1}{T_c} = \frac{1}{0.55} = 1.82 \text{ GHz}$$

$$\int x_3 = \frac{1}{T_{c3}} = \frac{1}{0.5} = 2 \text{ GHz}$$

$$d) CPI_1 = (0.6 \cdot 5 + 0.2 \cdot 4 + 0.2 \cdot (4+1)) = 4.8 \text{ c/i}$$

$$CPI_3 = (0.6 \cdot 5 + 0.2 \cdot 4 + 0.2 \cdot (4+3)) = 5.2 \text{ c/i}$$

$$e) T_{\text{exe}_1} = N \cdot CPI_1 \cdot T_c, \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{SpeedUp} = \frac{T_{\text{exe}_1}}{T_{\text{exe}_2}} = \frac{4.8 \cdot 0.55}{5.2 \cdot 0.5} = 1.015 \Rightarrow 1.5\%$$

$$\int_1 CPI_1 = (0.6 \cdot 5 + 0.2 \cdot 4 + 0.2 \cdot (0.1 \cdot (60+4+1) + 0.9 \cdot (9+1))) = 6 \text{ c/i}$$

$$CPI_3 = (0.6 \cdot 5 + 0.2 \cdot 4 + 0.2 \cdot (0.1 \cdot (60+4+3) + 0.9 \cdot (9+3))) = 6.4 \text{ c/i}$$

$$\int_2 T_{\text{exe}_1} = N \cdot CPI_1 \cdot T_c, \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{SpeedUp} = \frac{T_{\text{exe}_3}}{T_{\text{exe}_2}} = \frac{6 \cdot 0.55}{6.4 \cdot 0.5} = 1.031 \Rightarrow 3.1\%$$

6- a) T_{ma} { Unificada: CPI = 1.5 ciclos/instr
Dues caches: CPI = 1.2 ciclos/instr

$$T_{ma,1} = \text{ciclos}$$

$$\text{ciclos} = 0.089 \cdot 10 + 0.941 \cdot 1 = \underline{\underline{1.5 \text{ ciclos}}}$$

$$T_{ma,2} = \text{ciclos}$$

$$\text{ciclos} = 0.086 \cdot 10 + 0.914 \cdot 1 + 0.3(0.932 \cdot 1 + 0.068 \cdot 10) = \underline{\underline{1.74 \text{ ciclos}}}$$

b) $CPI_1 = 1.3(1.53 - 1) = 0.69 \frac{\text{ciclos}}{\text{instr}}$ } $T_{exec} = 2.19 \cdot 10 = \underline{\underline{21.9 \mu s}}$
 memoria $CPI_2 = 1.5 + 0.69 = 2.19 \frac{\text{ciclos}}{\text{instr}}$
 $CPI_2 = 1.3(1.74 - 1) = 0.96 \frac{\text{ciclos}}{\text{instr}}$ } $T_{exec} = 2.16 \cdot 10 = \underline{\underline{21.6 \mu s}}$
 $CPI_2 = 1.2 + 0.96 = 2.16 \frac{\text{ciclos}}{\text{instr}}$

c) La segona pega permet paral·lisme de dades i instruccions i, a més a més, tripe menys.

d) Seria millor pujar els 4KB de instruccions a 8KB, ja que falla menys.

9- a)

Bloc de memòria	73	55	43	45	73	45	13	43	73	55	45	73	15	43
Directa						X						X		
2-associativa					X	X				X	X	X		X
Directa + VC					X	X	X	X	X	X	X	X		

b) VC usa FIFO (First In First Out), que seria equivalent a LRU (Last Recently Used). Asi que no.

$$c) CPI_{ideal} = \frac{12 \cdot 10^9 \text{ ciclos}}{10 \cdot 10^9 \text{ instrucciones}} = \underline{\underline{1.2 \frac{\text{ciclos}}{\text{instr}}}}$$

$$d) nratio = \frac{3 \cdot 10^9 \text{ accessos}}{10 \cdot 10^9 \text{ instrucciones}} = \underline{\underline{0.3 \frac{\text{acessos}}{\text{instr}}}}$$

$$e) Ncicles = 10 \cdot 10^9 \cdot (1.2 + 0.3 \cdot 0.1 \cdot 10) = \underline{\underline{1.5 \cdot 10^{10} \text{ ciclos}}}$$

$$f) T_{exe} = 1.5 \cdot 10^{10} \cdot 10 \cdot 10^{-9} = \underline{\underline{150 \text{ s}}}$$

g) Probablemente sea debido al T_c (tiempo de ciclo).

$$h, Ncicles = 10 \cdot 10^9 (1.2 + 0.3 \cdot 0.05 \cdot 9) = \underline{\underline{1.34 \cdot 10^{10} \text{ ciclos}}}$$

$$i, T_{exe} = 1.34 \cdot 10^{10} \cdot 10 \cdot 10^{-9} = \underline{\underline{134 \text{ s}}}$$

$$j, Ncicles = 10 \cdot 10^9 (1.2 + 0.3 \cdot 0.06 \cdot 10) = \underline{\underline{1.41 \cdot 10^{10} \text{ ciclos}}}$$

$$k) T_{exe} = 1.41 \cdot 10^{10} \cdot 10 \cdot 10^{-9} = \underline{\underline{141 \text{ s}}}$$

l) La VC tiene un ciclo de penalización. (enunciado)

$$m) \begin{array}{c} 0.6 \\ \diagdown \quad \diagup \\ 0.1 \quad C \\ \diagup \quad \diagdown \\ 0.9 \quad C \end{array} \left\{ \begin{array}{l} V \\ \neg V \end{array} \right. \left\{ \begin{array}{l} \text{Por lo tanto} \\ C \cdot \neg V = 0.1 \cdot 0.4 = \underline{\underline{0.04}} \end{array} \right.$$

$$n) Ncicles = 10 \cdot 10^9 (1.2 + 0.3 \cdot (0.06 \cdot 11 + 0.04)) = \underline{\underline{1.41 \cdot 10^{10} \text{ ciclos}}}$$

$$o) T_{exe} = 1.41 \cdot 10^{10} \cdot 10 \cdot 10^{-9} = \underline{\underline{141 \text{ s}}}$$

12- a) $CPI_{ideal} = \frac{5 \cdot 10^9 \text{ ciclos}}{2 \cdot 10^9 \text{ instr}} = 2.5 \text{ c/i}$

b) Ciclos_f = $\frac{5 \cdot 10^9 \text{ ciclos}}{50 \cdot 10^6 \text{ fallos}} = 100 \text{ ciclos entre fallos}$

c) $CPI_B = 4 \cdot \frac{2 \cdot 10^9 \text{ Hz}}{2 \cdot 10^9 \text{ instr}} = 4 \text{ c/i}$

d) $T_{pf} = \frac{8 \cdot 10^9 - 5 \cdot 10^9 \text{ ciclos}}{50 \cdot 10^6 \text{ fallos}} = 60 \frac{\text{ciclos}}{\text{fallos}}$

e) $P_{fb} = 1 - (1 - p)^{60} = \underline{\underline{0.45}}$

f) El segundo fallo no permite continuar

g) Primero → Max = 59 ciclos (Todos excepto último)
Último → Min = 0 ciclos

h) Ciclos_p = $\frac{0+59}{2} = 29.5 \text{ ciclos}$

i) Ciclos = $5 \cdot 10^9 + 50 \cdot 10^9 \cdot 0.45 \cdot 29.5 = 5.67 \cdot 10^9$

$\begin{array}{cccc} T & T & T & T \\ \text{Ciclos} & \text{Fallos} & \text{Prob} & \text{media} \\ \text{ideal} & \text{2º fallo} & \text{ciclos}_p & \end{array}$

j) SpeedUp = $\frac{4}{\frac{5.67 \cdot 10^9}{1.9 \cdot 10^9}} = 1.34$

13:

Un procesador tiene una cache de primer nivel (que llamaremos L1) con bloques de 32 bytes y está conectado a un segundo nivel de jerarquía de memoria (que llamaremos L2) mediante un bus de 8 bytes de ancho. El primer nivel (L1) tiene un tiempo de acceso de 1 ciclo cuando se produce un acierto. Cuando fallamos en el primer nivel accedemos al segundo nivel (L2) para leer un bloque de datos. Suponemos que solo se realizan lecturas y que nunca hay fallos en L2. El siguiente cronograma ilustra un fallo en L1: se necesitan 5 ciclos de latencia y 4 para transferir los datos (T0-T3). Los datos se cargan en L1 mientras se transfieren (car L1), y una vez tenemos todo el bloque en L1, hay que realizar una lectura en L1 (Lect) para enviar el dato a la CPU (DATO), cosa que ocurre dentro del mismo ciclo.

Al ejecutar un programa en un simulador, hemos obtenido que, a una frecuencia de 2GHz el programa tardaría 2 segundos en el caso ideal de que no haya fallos de cache, que se han realizado 10^9 accesos a memoria y que el 20% de los accesos provocarían un fallo en L1.

- a) **Calculad** el tiempo de ciclo y los ciclos que tarda el programa en el caso ideal.

$$T_C = \frac{1}{f} = \frac{1}{2 \cdot 10^9} = \underline{0.5 \text{ ns}}$$

$$N_{\text{cycles}} = \frac{T_{\text{exe}}}{T_c} = \frac{2}{0.5 \cdot 10^9} = \underline{\underline{4 \cdot 10^9 \text{ cycles}}}$$

b)

- Calculad** los ciclos de penalización de un fallo de L1 y el tiempo de ejecución del programa teniendo en cuenta la penalización debida a los fallos de L1.

En caso de fallo, ciclos = Miss + Latencia + T0-T3 + Lectura = 11 ciclos. Es decir, 10 de penalización.

$$T_{\text{exe}} = (\text{Cicles}_{\text{ideal}} + \text{Cicles}_{\text{pen}}) \cdot T_c = (4 \cdot 10^9 + 0.2 \cdot 10 \cdot 10^9) \cdot 0.5 \cdot 10^{-9} = 3.5$$

Para mejorar el rendimiento, permitimos que el procesador pueda captar el dato en el mismo ciclo que se transfiere de L2 a L1 (continuación anticipada o *early restart*). Mediante simulación sabemos que cuando se produce un fallo en L1 en nuestro programa, en el 70% de los casos el dato deseado se corresponde al que se transfiere en el ciclo T0, mientras que hay una probabilidad del 10% de que se transfiera en cada uno de los ciclos restantes (T1-T3). Sabemos además que nunca se produce un fallo mientras se acaba de transferir el resto del bloque.

- c) **Completa** el siguiente cronograma en donde se ilustran las acciones a realizar en caso de fallo en L1, suponiendo que tenemos continuación anticipada y que el dato solicitado se corresponde al byte 12 del bloque.

d) **Calculad** el tiempo medio de penalización (en ciclos) de un fallo en L1 y el tiempo de ejecución del programa para el procesador con continuación anticipada.

$$\text{Penalización} = (m_{T_0} \cdot \text{Pen}_{T_0}) + (m_{T_1} \cdot \text{Pen}_{T_1}) + (m_{T_2} \cdot \text{Pen}_{T_2}) + (m_{T_3} \cdot \text{Pen}_{T_3}) = (0.7 \cdot 6) + (0.1 \cdot 7) + (0.1 \cdot 8) + (0.1 \cdot 9) = 6.6 \text{ ciclos}$$

$$T_{\text{exe}} = (\text{Cicles}_{\text{ideal}} + \text{Cicles}_{\text{pen}}) \cdot T_c = (4 \cdot 10^9 + 0.2 \cdot 6.6 \cdot 10^9) \cdot 0.5 \cdot 10^{-9} = 2.66 \text{ s}$$

Para mejorar más el rendimiento hemos modificado también L2 de forma que se transfiera el dato solicitado en el primer ciclo (T_0) de la transferencia (transferencia en desorden). Sabemos además que nunca se produce un fallo mientras se acaba de transferir el resto del bloque.

- e) **Completa** el siguiente cronograma en donde se ilustran las acciones a realizar en caso de fallo en L1, suponiendo que tenemos transferencia en desorden y que el dato solicitado se corresponde al byte 12 del bloque.

The diagram illustrates the timing of memory access across 16 clock cycles. The CLK signal shows active-low clock edges. The Ciclo signal indicates the memory address being accessed. The CPU row shows the state of the system: 'MISS' in L1 cache at cycle 1, followed by 'car.l.' (cache load) and 'Latencia' (latency) in L2 cache. The Tc label is placed under the L2 row at the end of the latency period.

- f) **Calculad el tiempo medio de penalización (en ciclos) de un fallo en L1 y el tiempo de ejecución del programa para el procesador con continuación anticipada.**

$$\text{Penalizació} = \text{latencia} + \text{Carga} = 5 + 1 = 6 \text{ ciclos}$$

$$T_{exe} = (\text{Ciclos ideal} + \text{Ciclos pen}) \cdot T_c = (4 \cdot 10^9 + 0.2 \cdot 6 \cdot 10^9) \cdot 0.5 \cdot 10^{-9} = 2.6 \text{ s}$$

g)

Calculad la ganancia (speedup) del sistema con continuación anticipada y del sistema con transferencia en desorden respecto el sistema sin ninguna mejora.

$$\text{Speed Up} = \frac{3}{2.66} = 1.13 \Rightarrow 13\%$$

$$\text{Speed Up}_2 = \frac{3}{2.6} = 1.15 \Rightarrow 15\%$$

Disponemos de un procesador de 16 bits con direcciones de 16 bits que tiene una memoria cache de datos con las siguientes características: 3-asociativa, con algoritmo de reemplazo LRU, 12 bloques y 64 bytes por bloque, política de escritura: *copy back + write allocate*

El contenido inicial de la memoria de etiquetas (tags) es el siguiente:

6 bit
byte

conjunto 0	DB	conjunto 1	DB	conjunto 2	DB	conjunto 3	DB
13	1	13	1	13	0	13	0
43	1	43	1	43	0	43	0
AC	0	AC	0	AC	1	AC	1

El DB=1 indica que el bloque correspondiente ha sido modificado. La información de reemplazo está implícita en la posición. Las posiciones inferiores corresponden a los bloques que llevan más tiempo sin utilizarse. Las posiciones superiores corresponden a los últimos bloques utilizados. Por ejemplo, en el conjunto 3, el bloque con tag 13 es el último utilizado, y el bloque con tag AC el que lleva más tiempo sin ser utilizado.

- a) **Rellenad** la siguiente tabla, indicando para cada referencia, el número de bloque de memoria que le corresponde, la etiqueta (TAG), a qué conjunto de MC va a parar, si es acierto o fallo (A/F), el bloque reemplazado cuando proceda, el número de bytes leídos de MP (si se lee de MP) y el número de bytes escritos en MP (si se escribe en MP)

tipo	dirección (hex)	bloque de memoria (hex)	TAG (hex)	conjunto MC	¿acierto o fallo? (A/F)	bloque reemplazado	bytes escritura MP	bytes lectura MP
LECT	B12B	2c4	B1	0	F	AC	0	64
LECT	B145	2c5	B1	1	F	AC	0	64
LECT	B1AF	2c6	B1	2	F	AC	64	64
LECT	B1C4	2c7	B1	3	F	AC	64	64
ESCR	4387	10e	43	2	H	—	0	0
LECT	1108	044	11	0	F	43	64	64
ESCR	1199	046	11	2	F	43	0	64
LECT	11AA	046	11	2	H	—	0	0

A la cache anterior le añadimos un **buffer de prefetch** de una entrada. En este buffer se hace prebúsqueda hardware del bloque **i+1** cuando se accede (tanto en acierto como en fallo) al bloque **i**, siempre que el **i+1** no esté ya en la cache o en el **buffer**. En este último caso, no se realiza **prefetch**.

- b) **Rellenad** la siguiente tabla (mismas referencias que la anterior) indicando, para cada referencia, el número de bloque de memoria que le corresponde, la etiqueta (TAG), a qué conjunto de MC va a parar, si se produce acierto o fallo en la cache (A/F), el número de bytes leídos de MP (si se lee de MP), el número de bytes escritos en MP (si se escribe en MP), el bloque de MP que se encuentra en el **buffer** (si procede), si se produce acierto o fallo (A/F) en el **buffer** y el bloque que se prebusca de MP (si procede).

tipo	dirección (hex)	bloque de memoria (hex)	TAG (hex)	conjunto MC	Cache ¿acierto o fallo?	bytes escritura MP	bytes lectura MP	bloque actual buffer	Buffer ¿acierto o fallo?	bloque prefetch buffer
LECT	B12B	2c4	B1	0	F	0	128	—	F	2c5
LECT	B145	2c5	B1	1	F	0	64	2c5	H	2c6
LECT	B1AF	2c6	B1	2	F	64	64	2c6	H	2c7
LECT	B1C4	2c7	B1	3	F	64	0	2c7	H	2c8
ESCR	4387	10e	43	2	H	0	0	2c8	F	—
LECT	1108	044	11	0	F	64	64	2c8	F	045
ESCR	1199	046	11	2	F	0	64	045	F	047
LECT	11AA	046	11	2	H	0	0	047	F	—

16:-

- a) **Completa el Cronograma 1:** hasta el ciclo 44.
Calcula el CPI para N=1.000.000 iteraciones.
Calcula el ancho de banda (en bytes por ciclo) entre el buffer y L2 para N=1.000.000 iteraciones.

Cronograma 1: Buffer de 1 entrada.

Iteración	<----- Iteración 0 ----->										<----- Iteración 1 ----->										Iteración 2								Iteración 3																					
Ciclo	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44						
movl %eax, a(,%esi,4)	A										-	-	A											-	-	A									-	-	A													
movl %eax, b(,%esi,4)	-	-	-	-	-	-	B							-	-	-	-	-	B								-	-	-	-	-	B					-	-	-	-	-	-								
incl %esi							i															i																			B									
cmpl \$N, %esi							c														c																													
jle A							j														j																													
Ocupación bus	a[0]					b[0]					a[1]					b[1]					a[2]					b[2]					a[3]					b[3]					a[4]									
# Buffer	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	
Buffer[0]	a[0]					b[0]					a[1]					b[1]					a[2]					b[2]					a[3]					b[3]					a[4]					b[4]				

CPI =

Ancho de banda =

- b) Rellena el **Cronograma 2**: hasta el ciclo 44.
Calcula el CPI para N=1.000.000 iteraciones.
Calcula el ancho de banda (en bytes por ciclo) entre el buffer y L2 para N=1.000.000 iteraciones.

Cronograma 2: Buffer de 2 entradas

Iteración	i0	i1	i2	i3	i4	i5	i6	i7																																						
Ciclo	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44		
movl %eax, a(%esi,4)	A		- A					- A				- A				- A			- A				- A				- A				- A				- A				- A							
movl %eax, b(%esi,4)	B		B		B		B		B		B		B		B		B		B		B		B		B		B		B		B		B		B		B		B		B					
incl %esi	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i							
cmpl \$N, %esi	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c								
j1 A	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j								
Ocupación bus																																														
# Buffer	0	1	2	2	2	2	1	1	2	2	2	2	1	1	2	2	2	2	1	1	2	2	2	2	1	1	2	2	2	2	1	1	2	2	2	2	1	1	2	2	2	2	1	1		
Buffer[0]	a[0]							a[1]							a[2]						a[3]							a[4]							a[5]						a[6]		a[7]			
Buffer[1]	b[0]							b[1]							b[2]						b[3]							b[4]							b[5]						b[6]		b[7]			

CPI = 15 Ancho de banda =

- c) **Rellena el Cronograma 3:** hasta el ciclo 44.
Calcula el CPI para N=1.000.000 iteraciones.
Calcula el ancho de banda (en bytes por ciclo) entre el buffer y L2 para N=1.000.000 iteraciones.

Cronograma 3: Buffer de 3 entradas

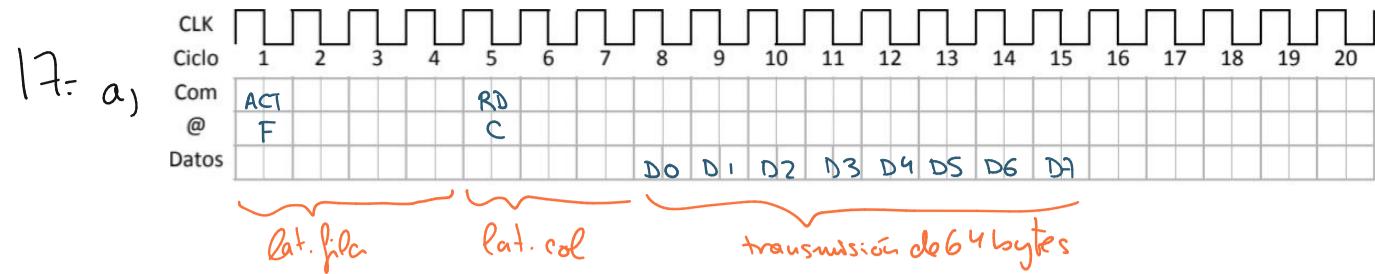
- Ancho de ba

d) Razóna ¿A qué crees que es

- e) Rellena el Cronograma 4: hasta el ciclo 44 en donde suponemos que tenemos un merge buffer de 3 entradas y 2 palabras (8 bytes) por entrada.
Calcula el CPI para N=1.000.000 iteraciones.

Mostrar los datos

Gráfico 4: Merge buffer de 3 entradas



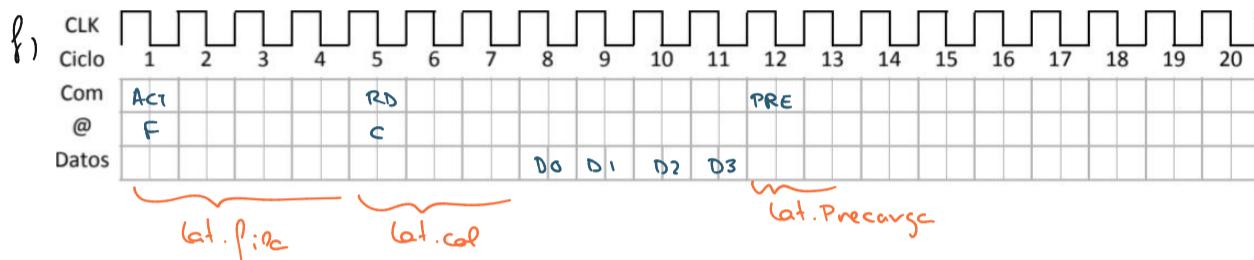
$$b) T_c = \frac{1}{200 \cdot 10^6} = 5 \text{ ns} \quad T_{exe} = n \cdot T_c = 17 \cdot 5 = 85 \text{ ns}$$

$$c) \text{Ancho banda}_T = \frac{64 \text{ bytes}}{8 \text{ ciclos}} \cdot 200 \cdot 10^6 = 1.6 \text{ GB/s}$$

$$d) \text{Ancho banda}_R = \frac{64 \text{ byte}}{17 \text{ ciclos}} \cdot 200 \cdot 10^6 = 752.94 \text{ MB/s}$$

$$e) P = P_1 + P_2 + P_3 = 1.5 \cdot \left(\frac{9}{25} \cdot 300 \cdot 10^{-3} + \frac{8}{25} \cdot 800 \cdot 10^{-3} + \frac{8}{25} \cdot 200 \cdot 10^{-3} \right) = 0.642 \text{ W}$$

$$E = P \cdot t = 0.642 \cdot (100 \cdot 5 \cdot 10^{-9}) = 3.2 \cdot 10^{-9} \text{ J}$$



18- a) $T_{exe} = N \cdot CPI \cdot T_c = 5 \cdot 10^9 \cdot 1.8 \cdot 10 \cdot 10^{-9} = 90 \text{ s}$

b) Accesos = $5 \cdot 10^9$

c) $C_{Pru} = 13 \text{ ciclos}$

d) $T_{ma} = t_{hit} + miss \cdot t_{pf} = 2.3 \text{ ciclos}$

e) $CPI = 1.8 + 1 \cdot 0.1 \cdot 13 = 3.1 \text{ c/i}$

Ciclos = $(90 / 10 \cdot 10^{-9}) + (0.5 \cdot 10^9 \cdot 13) = 1.55 \cdot 10^{10} \text{ ciclos}$

f) $T_{exe} = N \cdot CPI \cdot T_c = 155 \text{ s}$

g) $P(FL1 \wedge \neg FL2) = 0.1 \cdot 0.7 = 0.07 \Rightarrow 7\%$

h) $P(FL1 \wedge FL2) = 0.1 \cdot 0.3 = 0.03 \Rightarrow 3\%$

i) $N_{ciclos} = 5 \text{ ciclos}$

j) $N_{ciclos} = 15 \text{ ciclos}$

k) $N_{ciclos}_{TMA} = 2^{15} \text{ ciclos}$
 $T_{ma} = 25 \text{ ns}$

l) $CPI = 1.8 + 1 \cdot 0.1 \cdot 15 = 3.3 \text{ c/i}$

m) $t_{exe} = 3.3 \cdot 10 \cdot 5 = 165 \text{ s}$

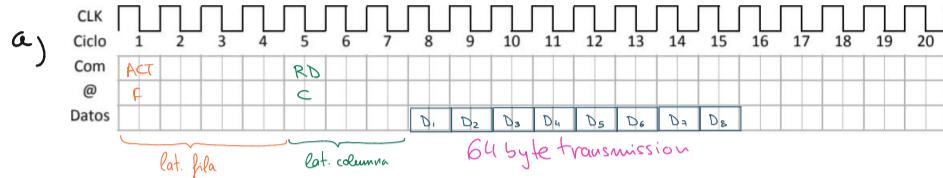
n) Speed Up = $\frac{165}{90} = 1.83$

17- SDRAM \rightarrow 8 chips - 1 byte per DIMM

latencias $\left\{ \begin{array}{l} \text{fila: 4 ciclos} \\ \text{columna: 3 ciclos} \\ \text{precarga: 2 ciclos} \end{array} \right.$

Frecuencia: 200 MHz

ACT: Active
RD: Read
PRE: PRECHARGE
@ F: ciclo dir. fila
@ C: ciclo dir. columna
 D_i : ciclo en que se transmite el dato $i = \{1, 2, 3, \dots\}$



$$b) T_c = \frac{1}{f} = \frac{1}{200 \cdot 10^6} = 5 \text{ ns}$$

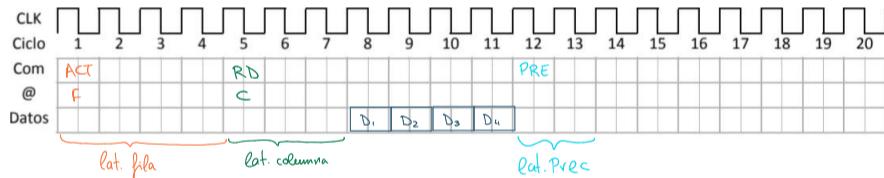
$$c) \text{Bandwidth}_{Th} = \frac{64 \text{ bytes}}{8 \text{ ciclos}} \cdot 200 \cdot 10^6 = 1.6 \text{ GB/s}$$

$$d) \text{Bandwidth}_R = \frac{64 \text{ bytes}}{17 \text{ ciclos}} \cdot 200 \cdot 10^6 = 753 \text{ MB/s}$$

$$e) \text{Potencia } \tau = P_1 + P_2 + P_3 \quad (V = 1.5V) \quad 100 \text{ ciclos} \rightarrow \frac{1}{25} \text{ accesos/ciclo}$$

$$\left. \begin{array}{l} P_1 = V \cdot I_1 \\ P_2 = V \cdot I_2 \\ P_3 = V \cdot I_3 \end{array} \right\} P_1 = V \cdot (I_1 + I_2 + I_3) = 1.5 \cdot \left(\frac{7+2}{25} \cdot 300 \cdot 10^{-3} + \frac{8}{25} \cdot 800 \cdot 10^{-3} + \frac{8}{25} \cdot 200 \cdot 10^{-3} \right) = 0.642 \text{ W}$$

f) Double Data Rate \rightarrow 2x Data transferred \rightarrow 16 bytes/ciclo



18- DRAM + Cache

$$a) T_{exec} = N \cdot CPI_{id} \cdot T_c = 5 \cdot 10^9 \cdot 1.8 \cdot 10 \cdot 10^{-9} = 90 \text{ s}$$

$$b) N \text{ accesos} = 5 \cdot 10^9 \leftarrow \text{accesos tardan 1 ciclo}$$

$$h) P(F_1 \wedge F_2) = 0.1 \cdot 0.3 = 0.03 \Rightarrow 3\%$$

$$c) T_{pf} \rightarrow \left. \begin{array}{l} (1) \text{ miss} \\ (14) \text{ hit} \end{array} \right\} T_{pf} = 13 \text{ ciclos}$$

$$i) T_{pf,g} = \left. \begin{array}{l} (1) \text{ miss} \\ (6) \text{ hit} \end{array} \right\} 5 \text{ ciclos}$$

$$d) T_{ma} = T_h + m \cdot T_{pf} = 2.3 \text{ ciclos}$$

$$j) T_{pf,h} = \left. \begin{array}{l} (1) \text{ miss} \\ (15) \text{ hit} \end{array} \right\} 14 \text{ ciclos}$$

$$T_{mam} = T_{ma} \cdot T_c = 2.3 \cdot 10 \cdot 10^{-9} = 2.3 \text{ ns}$$

$$k) T_{ma} = T_h + m \cdot T_{pf} = 1 + 0.03 \cdot 14 = 1.42 \text{ ciclos}$$

$$T_{mam} = T_{ma} \cdot T_c = 1.42 \cdot 10 \cdot 10^{-9} = 14.2 \text{ ns}$$

$$l) CPI = CPI_{id} + T_{pf} \cdot m = 1.8 + 0.1 \cdot 13 = 3.1 \text{ c/i}$$

$$l) CPI = 1.8 + 0.03 \cdot 1.42 = 1.84 \text{ c/i}$$

$$m) T_{exe} = N \cdot CPI \cdot T_c = 5 \cdot 10^9 \cdot 3.1 \cdot 10 \cdot 10^{-9} = 155 \text{ s}$$

$$m) T_{exe} = N \cdot CPI \cdot T_c = 5 \cdot 10^9 \cdot 1.84 \cdot 10 \cdot 10^{-9} = 92 \text{ s}$$

$$g) P(F_1 \wedge F_2) = 0.1 \cdot 0.7 = 0.07 \Rightarrow 7\%$$

$$n) \text{Speed Up} = \frac{T_{exe,2}}{T_{exe,1}} = \frac{155}{92} = 1.69 \Rightarrow 69\%$$

$$19.- \text{ a) } MTTF = \frac{10^9 \text{ horas}}{25000 \text{ fallos}} = 40000 \text{ h por Mbit}$$

$$b) \text{MTTF}_{\text{bit}} \left\{ \begin{array}{l} 1 \text{ Mbit} = 10^6 \text{ bit} \\ \text{MTTF} = 40000 \text{ h} \end{array} \right\} 40000 \cdot 10^6 = 4 \cdot 10^{10} \text{ horas por bit}$$

$$4 \cdot 10^{10} \text{ h} \cdot \frac{1 \text{ día}}{24 \text{ h}} \cdot \frac{1 \text{ año}}{\approx 365 \text{ días}} \cdot \frac{1 \text{ Milenio}}{10^6 \text{ años}} = 4.57 \text{ Mil. años (que no milenios)}$$

$$\hookrightarrow \text{MTTF}_{16GB} = \frac{\frac{40000}{131072}}{131072 / 20000} = 0.31 \text{ h per } 16GB$$

$$d) \text{MTTF}_{\substack{16GB \\ \text{ECC}}} \Rightarrow 1 \text{ de cada } 20000 \text{ es no recuperable} \Rightarrow \frac{40000}{6.55} = 6107 \text{ h}$$

$$\text{ej DIMM} \Rightarrow \frac{500000 \text{ sv}}{6107 \text{ gallos}} = 81.87 \frac{\text{DIMMs}}{\text{h}} \cdot 24 \text{ h} = 1965 \text{ DIMMs}$$

$$f_1: E = 70 \text{ MJ per chip} \cdot 18 \text{ chips per DIMM} \cdot 1965 \text{ DIMMs per die} \cdot 365 \text{ dies} = 903.7 \cdot 10^6 \frac{\text{MJ}}{\text{a\text{\'u}}}$$

$$\text{Emisión}_g = 903.7 \cdot 10^6 \text{ MJ por año} \cdot 50 \text{ g por MJ} = 4.52 \cdot 10^{10} \text{ g}$$

$$\text{Emissions} = 4.52 \cdot 10^{10} \text{ g} / 10^6 \text{ g por T} = 45185 \text{ toneladas}$$

20.- a) Falla una de cada 4, así que 25%.

$$C) \text{ CPI} = \frac{\text{cycles}}{\text{instr}} = \frac{43}{20} = 2.15 \text{ c/c}, \quad T_{exe} = \text{cycles} \cdot T_c = \frac{2.15 \cdot (64,000,000 \cdot 5)}{2 \cdot 10^9} = 0.34 \text{ s}$$

e) Uno completo y $\frac{64 \cdot 10^6 - 1}{4}$ parciales ($\approx 15.999.999$)

f_1 : 44 → 61 ⇒ 17 ciclos

$$g) CPI = \frac{36}{20} = 1.8 \text{ c/i}$$

$$T_{exec} = \text{cycles} \cdot T_c = \frac{1.8 \cdot (64.000.000 \cdot 5)}{2 \cdot 10^9} = 0.295$$

$$\text{Speed Up} = \frac{T_{\text{exec}_g}}{T_{\text{exec}_c}} = \frac{0.34}{0.29} = 1.17 \times$$

h) 7 de cada 8 pueden reusar pas \Rightarrow $\begin{cases} 2 \cdot 10^6 \text{ no reusar} \\ 14 \cdot 10^6 \text{ reusar} \end{cases} \} 16 \cdot 10^6$

Cronograma 7: Fallo que NO abre página.

i)

Cronograma 8: Fallo que SI abre página.

ج

$$k) t_1 = \frac{14}{2 \cdot 10^9} = 7 \cdot 10^{-9} \text{ s}$$

$$t_2 = \frac{31}{2 \cdot 10^9} = 15,5 \cdot 10^{-9} \text{ s}$$

$$l) CPI_1 = \frac{29}{20} = 1,45$$

$$CPI_2 = \frac{46}{20} = 2.3$$

$$CPI_{12} = \frac{1}{8} \cdot 2,5 + \frac{7}{8} \cdot 1,45 = 1,56 \Rightarrow Texe = \frac{1,56 \cdot 64 \cdot 10^6 \cdot S}{2 \cdot 10^9} = 0,255$$

$$\text{SpeedUp}_p = \frac{T_{exec}}{T_{exec}} = \frac{0.34}{0.25} = 1.36 \times$$

Cronograma 9: Prefech que NO abre página.

m)

Cronograma 10: Prefecth que SI abre página

o) Reusan → 10 ciclos

Norvegian \rightarrow 27 cids

$$P) CPI_1 = \frac{28}{20} = 1,4$$

$$CPI_2 = \frac{45}{30} = 2.3$$

$$CPT_{12} = 1.5 \Rightarrow T_{exe} = \frac{1.5 \cdot 64 \cdot 10^6 \cdot S}{2 \cdot 10^9} = 0.24S$$

$$\text{Speed Up} = \frac{0.34}{0.29} = 1.14 \times$$

9)

Cronograma 11: Fallo que abre página en la SDRAM con dos bancos

Disponemos de discos con las siguientes características:

4.1.

- Capacidad 1 Tbyte.
- Seek time medio (situar el cabezal en el cilindro) 8 ms.
- Latencia media (tiempo que tarda en pasar el sector deseado) 2 ms.
- Transfer Rate (ancho de banda durante la transferencia de datos) 256 Mbytes/s.
- MTTF 50.000 horas.
- Tamaño de sector 512 bytes.

a) **Calcula** cuánto tiempo tarda en transferirse un bloque de datos de 5000 sectores.

$$a) t_b = \frac{\# \text{sectores} \cdot \# \text{bytes/sector}}{\# \text{bytes/s}} = \frac{5000 \cdot 512}{256 \cdot 10^6} = 0.01 \text{ s} = 10 \text{ ms}$$

Si los 5000 sectores se encuentran almacenados de forma consecutiva en la misma pista, solo se pierde tiempo en situar el cabezal (seek time + latencia) en el primero de ellos ya que el disco es capaz de transferir los siguientes sectores a medida que gira el disco sin necesidad de situar el cabezal (suponemos que la velocidad de rotación es la adecuada para al Transfer Rate).

- b) **Calcula** el tiempo total necesario para leer un bloque de datos de 5000 sectores consecutivos desde que se envía la petición al disco hasta que los datos están en memoria (suponemos que el procesado de las peticiones por el disco, cálculo de CRCs, DMA, etc introducen un tiempo negligible).
- c) **Calcula** el ancho de banda efectivo al leer un bloque de datos de 5000 sectores consecutivos.

$$b) t_T = t_b + t_q + t_s = 10 + 2 + 8 = 20 \text{ ms}$$

$$c) 5000 \text{ sectores} \cdot 512 \text{ bytes/s} = 2,56 \text{ Mb}$$

$$\text{Ancho de Banda} = \frac{\# \text{bytes}}{\text{bytes/s}} = \frac{2,56 \text{ Mb}}{20 \cdot 10^{-3} \text{ s}} = 128 \text{ Mb/s}$$

En un computador con uno de estos discos ejecutamos una aplicación formada por 3 fases:

- La fase 1 lee 8 bloques de datos (de 5000 sectores consecutivos cada uno) de disco.
 - La fase 2 realiza los cálculos y representa el 40% del tiempo de la aplicación (ejecutada con un solo disco).
 - La fase 3 escribe 4 bloques de datos (de 5000 sectores consecutivos cada uno) a disco. Escribir un bloque de datos tarda lo mismo que leerlo.
- d) **Calcula** el tiempo que tarda la fase 2.

$$d) \text{fase}_1 \Rightarrow 8 \text{ bloques a } 20 \text{ ms por bloque} \Rightarrow 8 \cdot 20 = 160 \text{ ms}$$

$$\text{fase}_2 \Rightarrow 40\% \text{ total}$$

$$\text{fase}_3 \Rightarrow 4 \text{ bloques a } 20 \text{ ms por bloque} \Rightarrow 4 \cdot 20 = 80 \text{ ms}$$

$$t_T = t_1 + t_2 + t_3 \Rightarrow t_1 = 160 + 0.4 t_T + 80 \Rightarrow 0.6 t_T = 240 \Rightarrow t_T = 400 \text{ ms} \Rightarrow t_2 = t_T \cdot 0.4 = 160 \text{ ms}$$

Aunque los datos caben en un solo disco, para mejorar el rendimiento de la aplicación, instalamos un RAID 0 con 8 discos iguales con tiras de 5000 sectores de forma que los 8 bloques de datos de la fase 1 se encuentran en 8 discos distintos. Igualmente, los 4 bloques de la fase 3, también se encuentran en 4 discos distintos. Suponemos que el resto del sistema es capaz de soportar el ancho de banda necesario.

- e) **Calcula** el ancho de banda efectivo al leer los 8 bloques de datos del RAID 0
- f) **Calcula** el ancho de banda efectivo al escribir los 4 bloques de datos al RAID 0
- g) **Calcula** el speed-up de la fase 1
- h) **Calcula** el speed-up de la fase 3
- i) **Calcula** el speed-up de la aplicación

$$e) T_b = 5000 \text{ sectores} \cdot 512 \text{ bytes/s} = 2,56 \text{ Mb}$$

$$\text{Ancho de Banda} = \frac{T_b \cdot \# \text{bloques}}{\text{bytes/s}} = \frac{2,56 \cdot 8}{20 \cdot 10^{-3} \text{ s}} = 1024 \text{ Mb/s}$$

$$f) T_b = 5000 \text{ sectores} \cdot 512 \text{ bytes/s} = 2,56 \text{ Mb}$$

$$\text{Ancho de Banda} = \frac{T_b \cdot \# \text{bloques}}{\text{bytes/s}} = \frac{2,56 \cdot 4}{20 \cdot 10^{-3} \text{ s}} = 512 \text{ Mb/s}$$

$$g) \text{Speed Up}_1 = \frac{t_1}{t_b} = \frac{160}{20} = 8 \times$$

$$h) \text{Speed Up}_3 = \frac{t_3}{t_b} = \frac{80}{20} = 4 \times$$

$$i) \text{Speed Up}_T = \frac{t_{T_2}}{t_{T_1}} = \frac{400}{200} = 2 \times$$

Disponemos de 60 discos físicos de 300 Gbytes de capacidad por disco, que ofrecen un ancho de banda efectivo de 100 Mbytes/s por disco. Con estos discos deseamos montar un disco lógico en donde consideraremos las siguientes 4 opciones:

- RAID 6
- RAID 10 (mirror doble con 30 grupos de 2 discos)
- RAID 50 (con 6 grupos de 10 discos)
- RAID 51 (mirror doble con 2 grupos de 30 discos)

a) Calcular la cantidad de información útil que puede almacenar cada uno de los RAIDS considerados.

$$a) \text{ Raid 6} \rightarrow \# \text{ discos} \cdot \text{capacidad} = (60-2) \text{ discos} \cdot 300 \text{ Gbytes} = 17400 \text{ GB}$$

$$\text{Raid 10} \rightarrow \# \text{ discos} \cdot \text{capacidad} = (60/2) \text{ discos} \cdot 300 \text{ GB} = 9000 \text{ GB}$$

$$\text{Raid 50} \rightarrow \# \text{ discos} \cdot \text{capacidad} = (9 \cdot 6) \text{ discos} \cdot 300 \text{ GB} = 16200 \text{ GB}$$

$$\text{Raid 51} \rightarrow \# \text{ discos} \cdot \text{capacidad} = (\frac{60}{2}-1) \text{ discos} \cdot 300 \text{ GB} = 8700 \text{ GB}$$

Para analizar el ancho de banda consideraremos por separado el caso en que realizamos accesos a tiras consecutivas y el caso en que realizamos accesos a tiras aleatorias. En el caso de 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 de forma que el controlador del RAID siempre puede aprovechar el ancho de banda de todos los discos físicos.

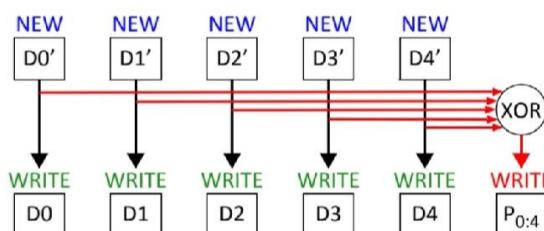
- b) Calcular el ancho de banda efectivo, si hacemos **lecturas** secuenciales, para cada uno de los RAIDS considerados.
- c) Calcular el ancho de banda efectivo, si hacemos **lecturas** aleatorias, para cada uno de los RAIDS considerados.

b) Dado que podemos aprovechar todos los discos en lectura:

$$\text{Raid 6} = \text{Raid 10} = \text{Raid 50} = \text{Raid 51} = V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot 60 \text{ discos} = 6 \text{ GB/s}$$

c) Dado que tenemos peticiones suficientes, el ancho de banda en todos será 6 GB/s.

Para aquellos RAIDS que tienen algún tipo de paridad (como 6, 50 o 51), en el caso de escrituras secuenciales es posible esperar a tener una cantidad de datos suficiente para calcular la paridad correspondiente y escribir simultáneamente en todos los discos. La siguiente figura muestra la escritura de 5 tiras de datos consecutivos D0-D4 en lo que podría ser un RAID 4 o 5 con 6 discos. Obsérvese que podemos aprovechar el ancho de banda de 5 discos para datos ya que el sexto es usado para escribir la paridad.



- d) Calcular el ancho de banda efectivo, si hacemos **escrituras** secuenciales, para cada uno de los RAIDS considerados.

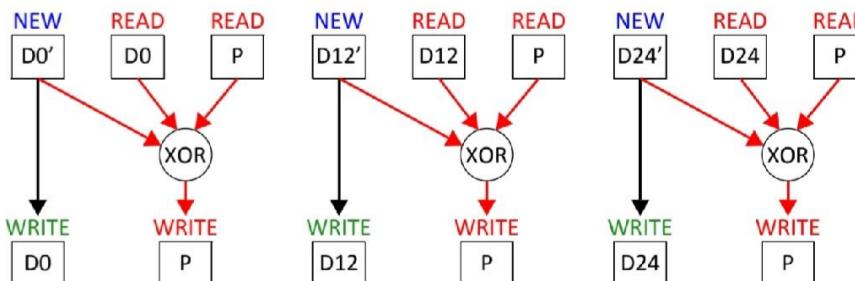
$$d) \text{ Raid 6} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot 58 = 5800 \text{ Mb/s}$$

$$\text{Raid 10} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot 30 = 3000 \text{ Mb/s}$$

$$\text{Raid 50} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot 54 = 5400 \text{ Mb/s}$$

$$\text{Raid 51} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot 29 = 2900 \text{ Mb/s}$$

En el caso de escrituras aleatorias (para RAIDS con algún tipo de paridad), se ha visto en teoría que es necesario leer los datos antiguos y la tira (o tiras) de paridad correspondiente para calcular la nueva paridad y además escribir tanto datos como paridad por cada escritura que se desea realizar. La siguiente figura muestra la escritura en paralelo de 3 tiras de datos independientes en lo que podría ser un RAID 5 con 6 discos (para acabarlo de entender es recomendable dibujar como estarían distribuidas las tiras de la 0 a la 24 y las paridades correspondientes en un RAID 5 con 6 discos). Obsérvese que para escribir 3 tiras de datos es necesario realizar 12 operaciones de disco realizando una lectura y una escritura en cada uno de los 6 discos, con lo que en la práctica el ancho de banda efectivo (de datos) es equivalente a 1,5 discos.



- e) Calcular el ancho de banda efectivo, si hacemos **escrituras** aleatorias, para cada uno de los RAIDS considerados.

$$e) \text{ Raid 6} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot \frac{60}{6} = 1000 \text{ Mb/s}$$

$$\text{Raid 10} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot \frac{60}{2} = 3000 \text{ Mb/s}$$

$$\text{Raid 50} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot \frac{60}{4} = 1500 \text{ Mb/s}$$

$$\text{Raid 51} \rightarrow V_{trans} \cdot \# \text{ discos} = 100 \text{ Mb/s} \cdot \frac{60}{8} = 750 \text{ Mb/s}$$

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

5.1 -

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

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

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

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

a) iremos por partes y de derecha a izquierda:

```
push D } C-D
push C }
sub

push B } B-A
push A }
sub

div   } (B-A)/(C-D)
push C } C - [(B-A)/(C-D)]
sub

pop R } R = C - [(B-A)/(C-D)] ✓
```

b) Con tipo acumulador:

```
load A } A-B
sub B }

store R } R = A-B

load C } C-D
sub D }

store T } Tenemos "R" ocupado, usamos un temporal
load R } Retomamos R

div T } R = R/T = (A-B)/(C-D)
store R }

load C } R = C - R = C - [(A-B)/(C-D)] ✓
sub R }

store R }
```

5.2 -

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

- 10^9 instrucciones dinámicas ejecutadas
 - CPI 2,5 ciclos/instrucción
 - el 30% de las instrucciones tienen un operando en memoria 1 acceso
 - el 10% de las instrucciones tienen dos operandos en memoria 2 accesos
 - el 70% de los accesos a memoria usan el modo base+desplazamiento
 - el 20% de los desplazamientos se codifican con más de 16 bits
 - el 20% de los accesos a memoria usan el modo registro indirecto (equivalente a base + desplazamiento cero)
 - el 20% de las instrucciones tienen un operando inmediato
 - el 15% de los inmediatos se codifican con más de 16 bits.
- a) ¿Cuántos accesos a datos se realizan durante la ejecución del programa?
- b) ¿A qué frecuencia debemos hacer funcionar el procesador para que el programa se ejecute en el tiempo previsto?

$$a) \# \text{accesos} = \underbrace{0.3 \cdot 10^9}_{\text{un oper.}} + \underbrace{2 \cdot 0.1 \cdot 10^9}_{2 \text{ oper.}} = 500 \cdot 10^6 \text{ accesos}$$

b) Buscamos $T_{exe} = 2.5$, tenemos $N = 10^9$ y $CPI = 2.5$:

$$f = \frac{N \cdot CPI}{T_{exe}} = \frac{10^9 \cdot 2.5}{2.5} = 10^9 \text{ Hz} = 1 \text{ GHz}$$

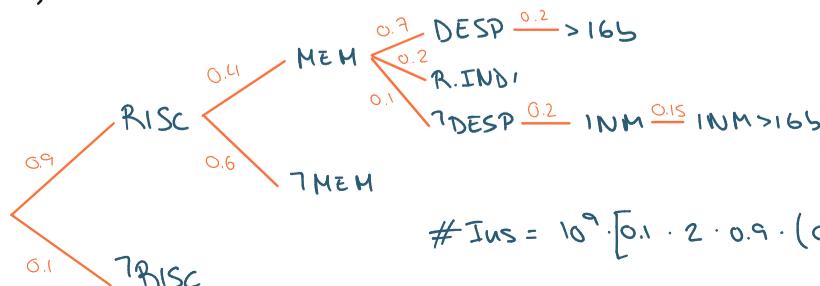
En la segunda generación del cachivache (el i-Cachivache 2) deseamos sustituir el procesador CISC por un RISC que esperamos tenga un menor consumo y aumente la duración de la batería. Para ello disponemos de un traductor que traduce del lenguaje máquina CISC al RISC. Al traducir el programa hemos obtenido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

- El 90% de las instrucciones CISC se han podido mapear sobre una instrucción RISC equivalente. El 10% restante ha necesitado 2 instrucciones RISC.
 - Los accesos a memoria han necesitado una instrucción adicional de Load/Store.
 - Los accesos a memoria que no se han podido implementar usando el modo base+desplazamiento han necesitado otra instrucción adicional para calcular la dirección.
 - Los load/store con desplazamientos de más de 16 bits han necesitado una instrucción adicional ya que el RISC sólo dispone de 16 bits para codificarlos.
 - Los inmediatos que necesitan más de 16 bits también han necesitado una instrucción adicional ya que los inmediatos también están limitados a 16 bits.
- c) ¿Cuántas instrucciones dinámicas ejecutará el procesador RISC?

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

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

c) Partiendo de todos los casos que da el enunciado, tendremos (además de un dolor de cabeza):



$$\# I_{us} = 10^9 \cdot [0.1 \cdot 2 \cdot 0.9 \cdot (0.6 \cdot 1 + 0.4 \cdot (0.7 \cdot (0.2 \cdot 2 + 0.8) + 0.2 \cdot 1 + 0.1 \cdot (0.2 \cdot 0.15 \cdot 2)))]] = 1.11 \cdot 10^5 \text{ instr. din.}$$

$$d) f = \frac{N \cdot CPI}{T_{exe}} = \frac{1.11 \cdot 10^9 \cdot 1,2}{2,5} = 0,53 \text{ GHz}$$

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

- e) Calcular la energía consumida por ambos procesadores al ejecutar el programa.
- f) Calcular la ganancia en duración de batería del RISC sobre el CISC

2) RISC:

$$P_T = P_f + P_c = 10 \cdot 1 + 50 \cdot 10^{-9} \cdot 1^2 \cdot 10^9 = 60 \text{ W}$$

$$E_T = P_T \cdot T_{exe} = 60 \cdot 2,5 = 150 \text{ J}$$

CISC:

$$P_T = P_f + P_c = 8 \cdot 1 + 40 \cdot 10^{-9} \cdot 1^2 \cdot 0,53 \cdot 10^9 = 29,2 \text{ W}$$

$$E_T = P_T \cdot T_{exe} = 29,2 \cdot 2,5 = 73 \text{ J}$$

$$g) \text{ ganancia} = \frac{E_{T_{RISC}}}{E_{T_{CISC}}} = \frac{150}{73} = 2,05 \times$$

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

- g) ¿A qué frecuencia debería funcionar la CPU RISC con el nuevo programa compilado?
- h) ¿Cuál será la ganancia en duración de batería con el nuevo código?

$$g) f = \frac{N \cdot CPI}{T_{exe}} = \frac{1,5 \cdot 10^9 \cdot 1,3}{2,5} = 0,78 \text{ GHz}$$

$$h) P_T = P_f + P_c = 8 \cdot 1 + 40 \cdot 10^{-9} \cdot 12 \cdot 0,78 \cdot 10^9 = 39,2 \text{ W}$$

$$E_T = P_T \cdot T_{exe} = 39,2 \cdot 2,5 = 98 \text{ J} \leftarrow \text{probablemente los apartados c y d están mal.}$$

$$\text{Ganancia} = \frac{E_{T_{RISC}}}{E_{T_{CISC_2}}} = \frac{150}{98} = 1,53 \times$$

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

5.3-

Tipo	Descripción	Ejemplos
LOAD	Rd $\leftarrow M[Rb + Ri * s + dspl]$	LOAD %r1 $\leftarrow M[%ebx + %edx * 4 + 36]$
STORE	M[Rb + Ri * s + dspl] $\leftarrow Rf$	STORE M[%ebx + %edx * 4 + 36] $\leftarrow %r3$
MOV	Rd $\leftarrow Rf_1$ Rd \leftarrow inmediato	MOVL %eax $\leftarrow \$3$ MOVL %r1 $\leftarrow %ebx$
Aritmética	Rd $\leftarrow Rf_1 \{op\} Rf_2$ Rd $\leftarrow Rf_1 \{op\}$ inmediato	ADDL %r1 $\leftarrow %r2 + %eax$ IMULL %eax $\leftarrow %r4 * \$20$
Comparación	flags $\leftarrow Rf_1 - Rf_2$ flags $\leftarrow Rf_1 -$ inmediato	CMPL %eax, %r5 CMPL %eax, \$100
Salto	mismos saltos que x86	JGE loop

Solo las instrucciones de LOAD y STORE pueden acceder a memoria, el resto sólo pueden tener registros o inmediatos como operandos. Además de los registros visibles en x86 disponemos de registros adicionales para almacenar valores intermedios (%r1 ... %r8). Los modos de direccionamiento de LOAD y STORE son los mismos que x86.

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

```

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

```

fin:

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

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

- c) Calcular el CPI de este código

Suponiendo una frecuencia de 3GHz

- d) Calcular el tiempo de ejecución de este código

$$b) \# \text{instrucciones} = 10^6 \cdot 7 + 1 = 7.000.001$$

$$\# \text{uops} = 10^6 \cdot 10 + 1 = 10.000.001$$

$$c) \text{CPI} = \frac{\text{ciclos}}{\# \text{instr}} = \frac{7.692.308}{7000.001} = 1,1 \text{ c/i}$$

$$\text{ciclos} = 10.000.001 \text{ uops} / 1,3 \text{ uops/ciclo} = 7.692.308 \text{ ciclos}$$

$$d) T_{exe} = \frac{N \cdot CPI}{f} = \frac{7000001 \cdot 1,1}{3 \cdot 10^9} = 2,56 \text{ ms}$$

Supongamos que las instrucciones x86 ocupan:

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

Las uops ocupan todas 6 bytes.

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

$$e) \text{uops} \Rightarrow 6 \text{bytes} \cdot 11 \text{ uops} = 66 \text{bytes}$$

código \Rightarrow Sacado de desglosarlo con el enunciado $\Rightarrow 44$ bytes

```

a) movl %ecx  $\leftarrow \$0$ 
loop: cmpl %ecx, $1000000
      jge fin
      load %eax  $\leftarrow x$ 
      imull %eax  $\leftarrow %rax \cdot %r1$ 
      load %r2  $\leftarrow$  suma
      addl %r2  $\leftarrow r2 + %rax$ 
      strel suma  $\leftarrow %r2$ 
      addl %r2  $\leftarrow %r2 + \$1$ 
      jmp loop

```

fin: