MEMORIA CACHÉ

MEMORIA PRINCIPAL
0x123456
0x682959
0x927930
0x452894
0x758493
0x819048
0x148593

MEMORIA CACHÉ
0x148593
0x927930
0x758493

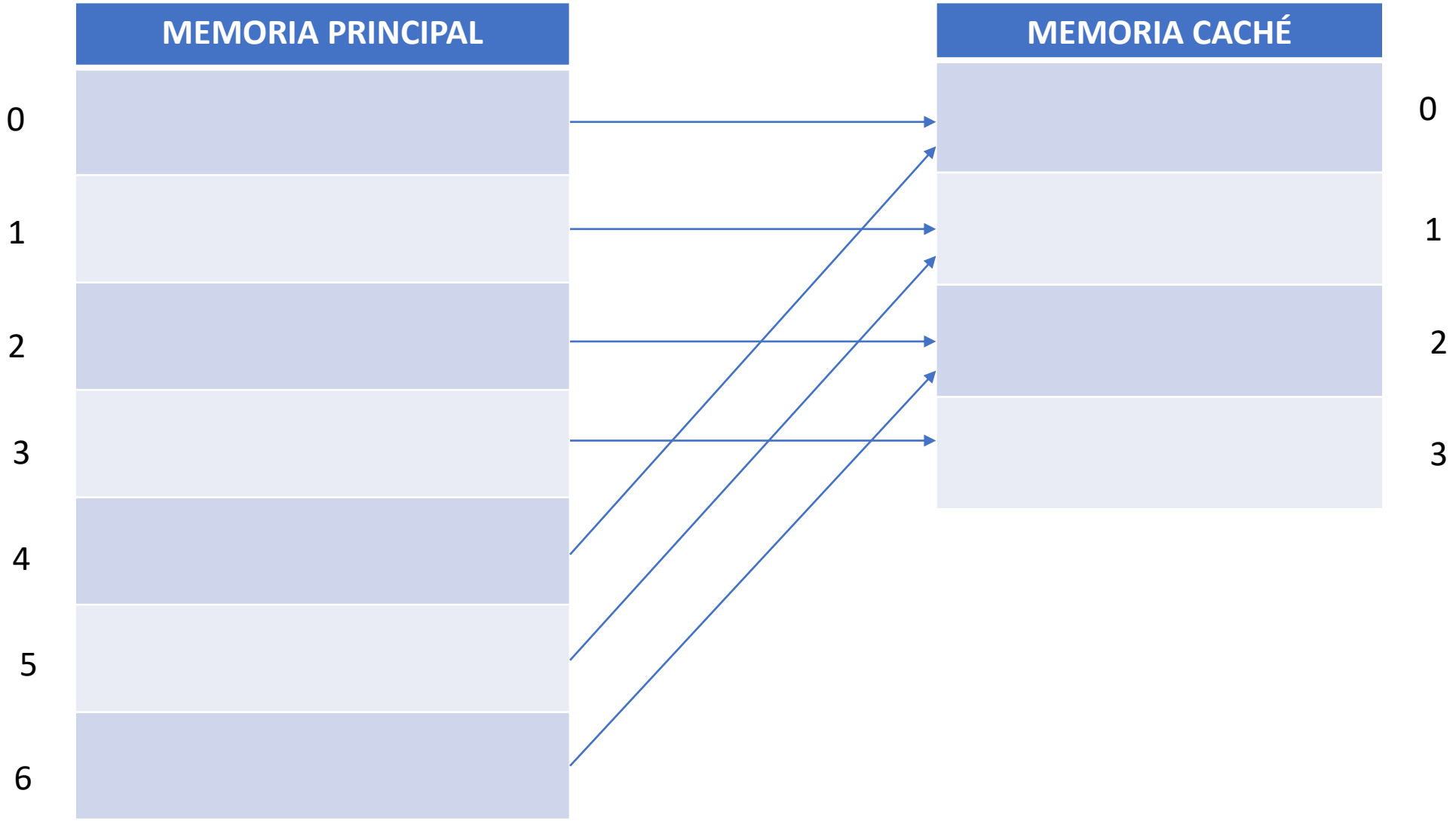
Parámetros que modifican el funcionamiento

- **Tipo de emplazamiento**
  - Correspondencia directa
  - Asociativa por conjuntos
  - Completamente asociativa

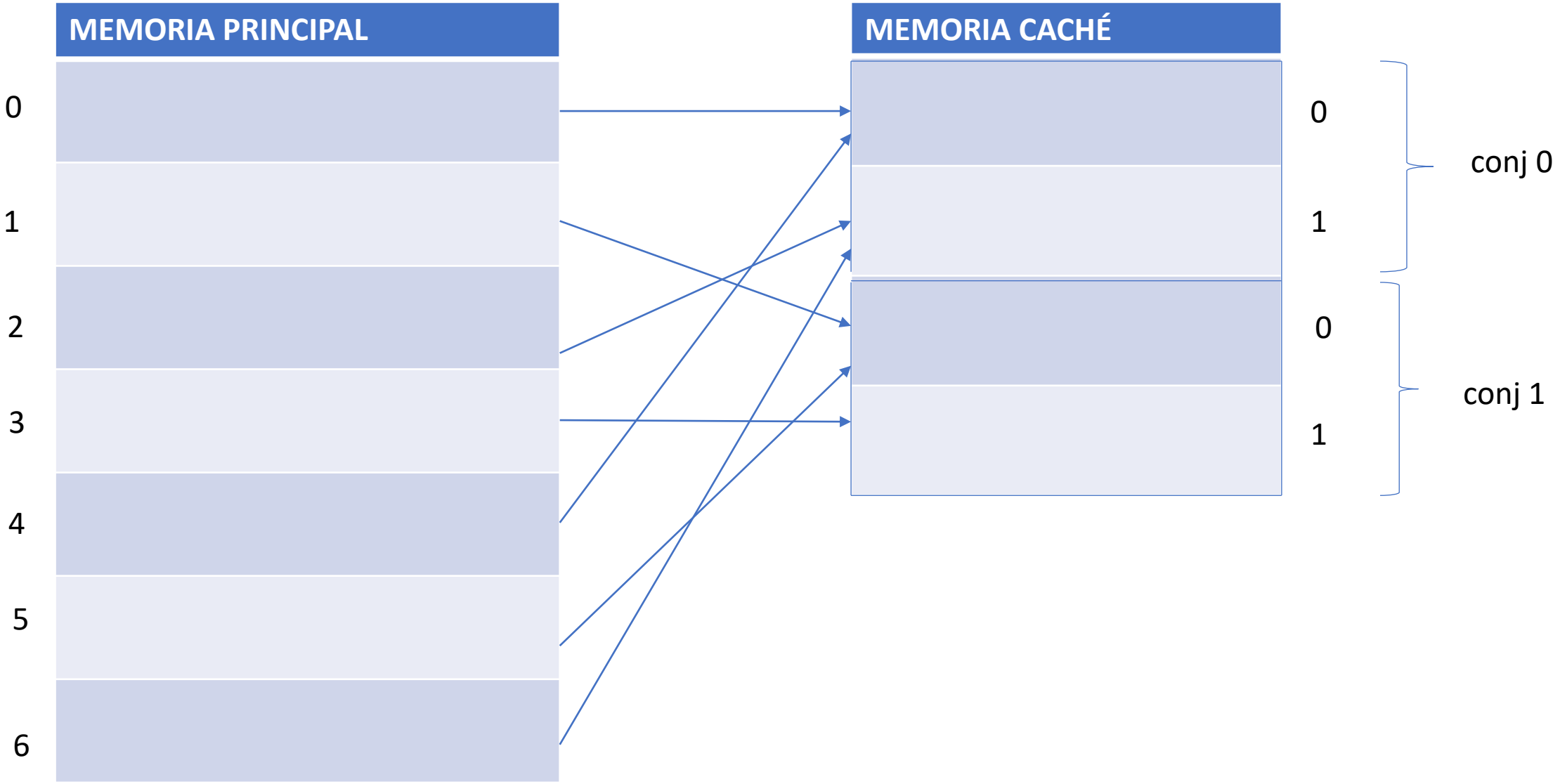
- **Algoritmo de reemplazamiento**
  - Aleatorio
  - FIFO (First In First Out)
  - LRU (Last Recently Used)

- **Políticas de escritura**
  - Write Through / CopyBack
  - Write Allocate / Write no Allocate

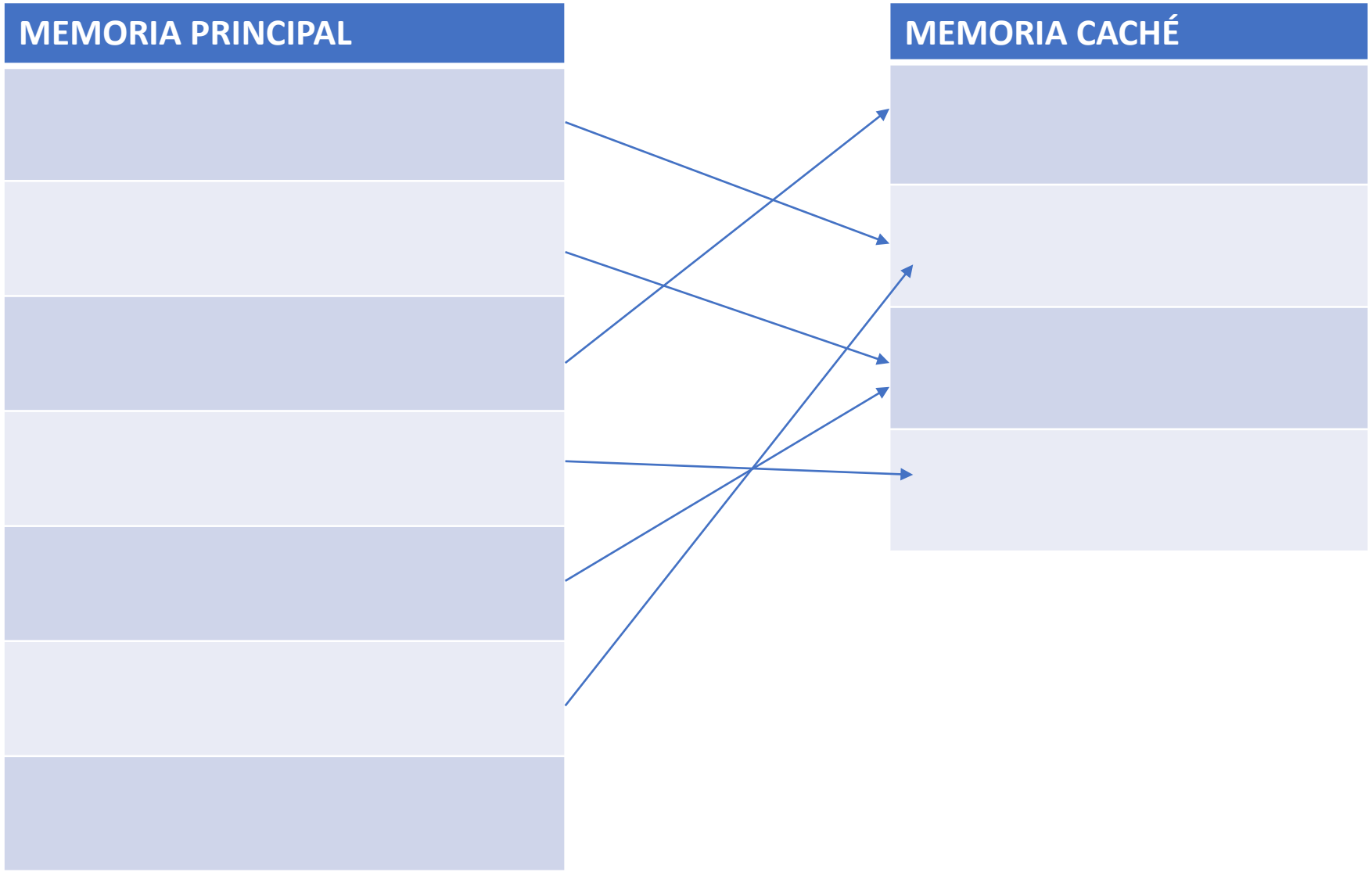
Correspondencia directa



Asociativa por conjuntos



Completamente asociativa



Algoritmos de reemplazamiento

- **FIFO** (First in first out): la linea en la que se ha hecho el primer acceso es la primera que se reemplaza

- **LRU** (Last recently used): la linea que hace más tiempo que no se accede es la que se reemplaza (hace falta llevar un recuento de los accesos)

- **Aleatorio**: la línea que se va a reemplazar se escoge aleatoriamente

Políticas de escritura

- **Write Through / CopyBack**
  - **Writhe Through** (escritura inmediata): se escribe en MP y MC a la vez
  - **CopyBack** (ecritura retardada): solo se escribe en MC (dirtybit)

- **Write Allocate / Write no Allocate**
  - **Write Allocate** (con asignación): copiamos el bloque a MC y escribimos en MC
  - **Write no Allocate** (sin asignación): no copiamos el bloque a MC, escribimos directamente en MP

Formato de las direcciones

- Correspondencia directa

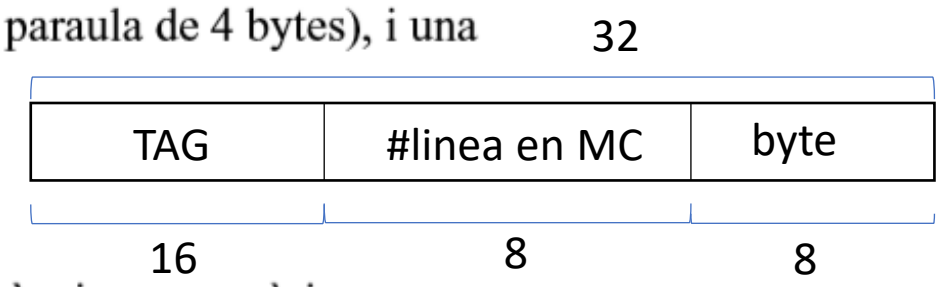
- Asociativa por conjuntos

- Completamente asociativa

Un sistema disposa d'un processador MIPS (32 bits d'adreces i mida de paraula de 4 bytes), i una memòria cache (MC) de 64 Kbytes amb la següent organització:

- Correspondència directa
- Blocs de 256 bytes = 2^8
- Escriptura immediata sense assignació

Estant la cache inicialment buida, un programa fa una seqüència de referències a memòria segons s'indica a la següent taula, on apareixen les adreces en hexadecimal i si són lectures o escriptures (L/E). Completa les columnes que falten indicant, per a cada referència: l'índex de MC; si és encert (e) o fallada (f); i el nombre de bytes de Memòria Principal (MP) llegits i/o escrits.

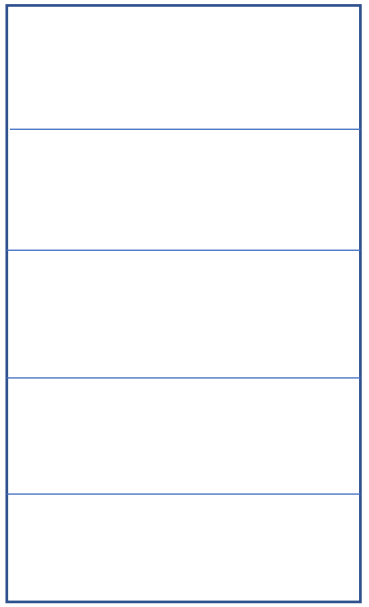


#lineas en MC =
Tamaño MC/tamaño linea
= 64KB / 256 B =
= 2^16 / 2^8 = 2^8 lineas

L/E	adreça (hex)	índex MC	encert (e)/ fallada (f)	bytes de MP	
				llegits	escrits
E	00010000				
L	00010008				
E	00010016				
L	00F40316				
E	03100004				
L	00010024				
L	03200308				
L	00F403A8				

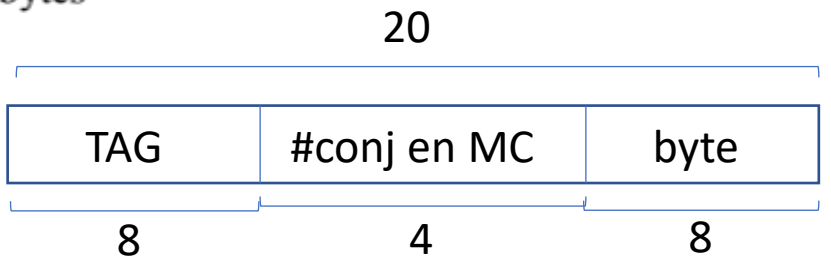
00000000000000010000000000000000
00000000000000010000000000001000
00000000000000010000000000010110
00000000111101000000011000010110
00000011000100000000000000000100
00000000000000010000000000100100
00000011001000000000011000001000
00000000111101000000011110101000

MC



Un sistema disposa d'un processador de 20 bits d'adreces, i una memòria cache (MC) de 8 Kbytes amb la següent organització:

- Correspondència associativa per conjunts, de grau 2 (2 blocs per conjunt)
- Blocs de 256 bytes = 2^8
- Reemplaçament LRU
- Escriptura retardada amb assignació.



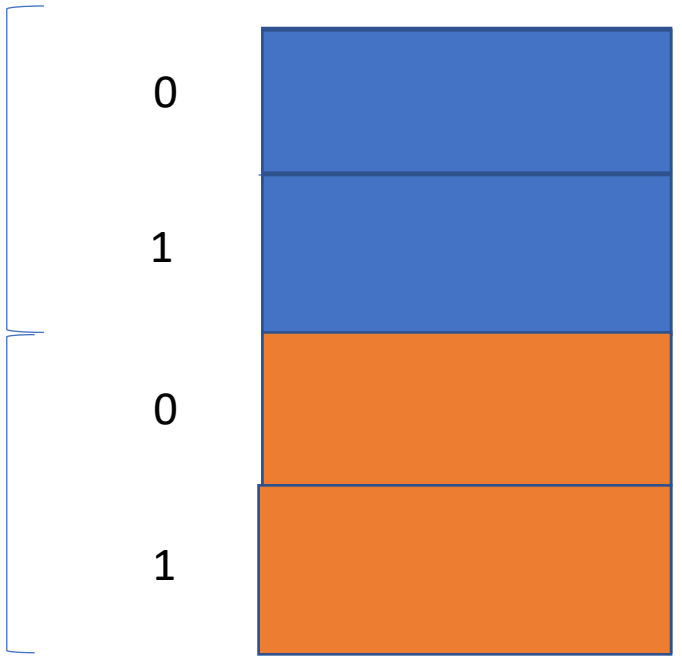
Estant la cache inicialment buida, un programa produeix una seqüència de referències a memòria segons s'indica a la següent taula, on apareixen les adreces en hexadecimal i si són lectures o escriptures (L/E). Completa les columnes que falten indicant, per a cada referència: el número de conjunt de MC; si és encert (e) o fallada (f); i el nombre de bytes de Memòria Principal (MP) llegits i/o escrits.

#lineas en MC = tamMC / tamLinea =
= 8KB / 256B = 2^13 / 2^8 = 2^5 lineas

32 lineas, 2 lineas por conjunto, 16 conjuntos = 2^4

L/E	adreça (hex)	núm. de conjunt	encert (e)/ fallada (f)	bytes de MP	
				llegits	escrits
L	01200				
E	082A0				
L	083F0				
E	01204				
L	04204				
E	083F4				
L	01208				
E	082A0				

00000001001000000000
00001000001010100000
00001000001111110000
00000001001000000100
00000100001000000100
00001000001111110100
00000001001000001000
00001000001010100000



Suposem que tenim un processador de 32 bits amb una memòria cache de dades de 64 bytes, on cada bloc té 16 bytes. Suposem que executem els següents programes.

```
//programa A
int M[4][16];

void main() {
int i, j;   //en registre
    for (i=0; i<3; i++)
      for (j=0; j<16; j++)
        M[i][j] = M[i+1][j];
}
```

```
//programa B
int M[4][16];

void main() {
int i, j;   //en registre
    for (j=0; j<16; j++)
      for (i=0; i<4; i++)
        M[i][j] = M[i][j] + 1;
}
```

Calcula el nombre de fallades de la cache suposant que la memòria cache és inicialment buida. L'adreça base de la matriu M és 0.

a) Suposant que la cache és de correspondència directa i té la política d'**escriptura retardada amb assignació**.

fallades_A = ____ fallades_B = ____

b) Suposant que la cache és **completament associativa** (algorisme de reemplaçament LRU), i que té la política d'**escriptura immediata sense assignació**.

fallades_A = ____ fallades_B = ____

Considera un processador amb adreces de 16 bits i una memòria cache amb aquesta configuració:

- capacitat total: 8 blocs
- mida del bloc: 16 bytes
- correspondència associativa per conjunts 2 blocs per conjunt
- política de reemplaçament aleatòria (random)
- escriptura retardada amb assignació (write-back, write-allocate)

En un moment donat l'estat de la part de control de la memòria cache és:

Conjunt	V	D	Etiqueta	V	D	Etiqueta
0	1	0	0x018	0	0	--
1	1	1	0x005	1	0	0x008
2	1	1	0x007	1	1	0x00A
3	0	0	--	1	0	0x003

a) (0,5 p.) Si la següent referència a tractar fos només una de les adreces de la llista següent, indica quines provocarien una fallada: 0x0605, 0x0435, 0x0151, 0x03C6

Provocarien fallada: [ ]

b) (0,5 p.) Si la següent referència a tractar fos només una de les adreces de la llista següent, ¿quines provocarien una escriptura a memòria principal?: 0x06D5, 0x043F, 0x02EF, 0x09F8

Segur que sí: [ ]

Pot ser que sí: [ ]

Segur que no: [ ]

c) (0,4 p.) Calcula el nombre de fallades en executar els següents programes, suposant que els enters es representen per 16 bits, la cache és inicialment buida i l'adreça inicial de M és 0x0.

```
int M[4][64];
void main(){
int i, j;        //en registres
  for (i=0; i<4; i++)
    for (j=0; j<64; j++)
                 M[i][j]= 0;
}
```

Nombre de fallades = [ ]

```
int M[4][64];
void main(){
int i, j;        //en registres
  for (j=0; j<64; j++)
    for (i=0; i<4; i++)
                 M[i][j]= 0;
}
```

Nombre de fallades = [ ]

d) (0,3 p.) Mantenint la capacitat de la cache i la mida del bloc ¿com es podria canviar l'associativitat per tal que els dos programes anteriors provoquessin el nombre mínim de fallades?

[ ]