Sistema de memòria

Miquel Albert Orenga
Gerard Enrique Manonellas

PID_00218254

UOC
Universitat Oberta de Catalunya

Índex

Introducció

Tot computador necessita un sistema de memòria per a emmagatzemar els programes que s'executen i les dades necessàries per a executar aquests programes. Des del punt de vista del programador, seria desitjable disposar de quantitats il·limitades de memòria i de velocitat il·limitada si fos possible per a emmagatzemar el programa que es vol executar i les dades necessàries; això permetria al programador fer la tasca d'escriure programes sense haver de tenir present cap tipus de limitació. Lògicament aquest desig no és factible i les quantitats de memòria de què disposa un computador tenen una limitació en capacitat i velocitat.

La quantitat de memòria que pot tenir un computador respon bàsicament a un factor de cost: com més memòria instal·lada, més elevat és el cost. De manera semblant, la velocitat de la memòria també depèn del cost, les memòries més ràpides tenen un cost més elevat, però no es pot aconseguir tota la velocitat necessària simplement incrementant el cost, hi ha un factor tecnològic que limita la velocitat de la memòria: no podem adquirir memòria més ràpida que la que hi ha al mercat en un moment donat.

Hi ha diferents tipus de memòries, amb capacitats i temps d'accés diferents. En general, com més capacitat d'emmagatzematge té una memòria, més gran és el temps d'accés. És a dir, les memòries amb gran capacitat són memòries lentes, mentre que les memòries ràpides (temps d'accés petit) solen tenir poca capacitat d'emmagatzematge. Les memòries ràpides són més cares que les memòries lentes. Per això, els dissenyadors de computadors han d'arribar a un compromís a l'hora de decidir quanta memòria posen en els seus dissenys i de quina velocitat o temps d'accés.

En els darrers anys, l'evolució de la tecnologia ha permès reduir molt l'espai necessari per a emmagatzemar un bit d'informació. Això ha originat que la mida de les memòries augmenti molt amb relació a l'espai físic que ocupen, i que es redueixi el preu que s'ha de pagar per un bit d'informació. Així, els discos durs han passat dels 20 Mbytes de capacitat a mitjan dècada dels vuitanta als 2.000 Gbytes al final del 2010 (100.000 vegades més), tot i que ocupen el mateix espai físic (fins i tot són una mica més petits) i costen gairebé el mateix. Això ha representat una reducció important en el preu per bit. Aquest ha estat un factor molt important perquè els computadors actuals incorporin molta més memòria que els computadors de fa trenta anys. Per tant, a l'hora de dissenyar un sistema de memòria cal tenir presents les característiques de capacitat, velocitat (i temps d'accés) i cost per bit.

Altres característiques també importants que cal considerar són la localització, l'organització, el mètode d'accés o la tecnologia de fabricació.

Objectius

Amb l'estudi d'aquest mòdul es pretén que l'estudiant assoleixi els objectius següents:

1. Conèixer les característiques bàsiques d'una memòria.

2. Comprendre els conceptes bàsics sobre l'organització de la jerarquia de memòria d'un computador.

3. Analitzar com es gestionen les dades entre els diferents nivells de la jerarquia de memòries, especialment en la memòria cau.

4. Conèixer les diferents tecnologies utilitzades per a implementar els diferents tipus de memòries utilitzats en un computador.

1. Característiques de les memòries

Les característiques més importants dels diferents tipus de memòria són la localització, la capacitat, el mètode d'accés, l'organització de les dades en una memòria, el temps d'accés i velocitat i el cost. Les estudiarem en aquest apartat.

1.1. Localització de la memòria

Podem classificar els tipus de memòria per la seva localització dins del computador. Bàsicament es poden classificar en memòria dins del xip del processador, memòria interna (memòria a la placa base del computador) i memòria externa.

Dins del xip del processador habitualment hi ha els registres i un nivell de memòria cau o diversos.

La memòria interna correspon a la memòria principal (memòria RAM del computador) i addicionalment un nivell de memòria cau o diversos.

La memòria externa correspon als dispositius d'emmagatzematge secundari, com discos durs, unitats òptiques (CD-ROM, DVD, o Blu-ray), unitats de cinta, etc.

Vegeu també

En aquest mòdul aprofundirem en l'estudi del sistema de memòria format per la memòria que es troba al processador i la memòria interna, i farem una revisió més general de la memòria externa.

1.2. Capacitat de la memòria

La capacitat (o mida de la memòria) fa referència a la quantitat d'informació que es pot emmagatzemar. La unitat utilitzada per a especificar la capacitat d'emmagatzematge d'informació és el byte (1 byte = 8 bits), i a l'hora d'indicar la capacitat s'utilitzen prefixos diferents que representen múltiples del byte.

En el sistema internacional de mesures (SI) s'utilitzen prefixos que representen múltiples i submúltiples d'una unitat; aquests prefixos SI corresponen sempre a potències de 10.

Prefixos SI

Cada prefix del sistema internacional rep un nom diferent i utilitza un símbol per a representar-lo. Els prefixos que utilitzarem més habitualment són:

10^{-12}	pico (p)
10^{-9}	nano (n)
10^{-6}	micro (μ)
10^{-3}	mili (m)

10^3	kilo (K)
10^6	mega (M)
10^9	giga (G)
10^{12}	tera (T)

Exemples de prefixos SI

10^3 bytes = 1.000 bytes = 1 Kilobyte (KB o Kbyte)

10^6 bytes = 10^3 KB = 1000 KB = 1 Megabyte (MB o Mbyte)

10^9 bytes = 10^3 MB = 1000 MB = 1 Gigabyte (GB o Gbyte)

10^{12} bytes = 10^3 GB = 1.000 GB = 1 Terabyte (TB o Tbyte)

Ara bé, en informàtica, la capacitat d'emmagatzematge habitualment s'indica en múltiples que siguin potències de 2; en aquest cas s'utilitzen els prefixos definits per la International Electrotechnical Commission (IEC).

Prefixos IEC

Els prefixos IEC representen múltiples per a les unitats d'informació bit i byte. Els noms provenen d'afegir el terme *binari* als prefixos SI. Per exemple *kibi* seria la contracció de *kilo binari*. Els prefixos que utilitzarem més habitualment són:

2^{10}	kibi (Ki)
2^{20}	mebi (Mi)
2^{30}	gibi (Gi)
2^{40}	tebi (Ti)

Exemples de prefixos IEC

2^{10} bytes = 1024 bytes = 1 KiB (kibibyte)

2^{20} bytes = 1024 KiB = 1 MiB (mebibyte)

2^{30} bytes = 1024 MiB = 1 GiB (gibibyte)

2^{40} bytes = 1024 GiB = 1 TiB (tebibyte)

La indústria utilitza majoritàriament les unitats SI. Per exemple, si ens fixem en les característiques d'un disc dur que es comercialitzi amb 1 TB de capacitat, realment la capacitat del disc serà de 1.000 GB = 1.000.000 MB = 1.000.000.000 KB = 1.000.000.000.000 bytes.

En canvi quan connectem aquest disc a un computador i mostrem les propietats del dispositiu, veurem que en la majoria de sistemes operatius se'ns mostrarà la capacitat en unitats IEC; en aquest cas, 976.562.500 KiB = 953.674 MiB = 931 GiB = 0,91 TiB.

1.3. Mètodes d'accés

Cada tipus de memòria utilitza un mètode a l'hora d'accedir a les posicions de memòria. Hi ha mètodes d'accés diferents característics de cada tipus de memòria:

1) Seqüencial. S'accedeix des de la darrera posició a la qual s'ha accedit, llegint en ordre totes les posicions de memòria fins a arribar a la posició desitjada. El temps d'accés depèn de la posició a la qual es vol accedir i de la posició a la qual s'ha accedit anteriorment.

Usos de l'accés seqüencial

L'accés seqüencial s'utilitza bàsicament en dispositius de cinta magnètica.

2) Directe. La memòria s'organitza en blocs i cada bloc de memòria té una adreça única, s'accedeix directament al principi d'un bloc i dins d'aquest es fa un accés seqüencial fins a arribar a la posició de memòria desitjada. El temps d'accés depèn de la posició a la qual es vol accedir i de la darrera posició a la qual s'ha accedit.

Usos de l'accés directe

L'accés directe és un mètode d'accés que s'utilitza en discos magnètics.

3) Aleatori. La memòria s'organitza com un vector, en què cada element individual de memòria té una adreça única. S'accedeix a una posició determinada proporcionant-ne l'adreça. El temps d'accés és independent de la posició a la qual s'ha accedit i és independent de la darrera posició a la qual s'ha accedit.

Les operacions bàsiques utilitzades quan treballem amb la memòria són:

a) Operació de lectura: en aquesta operació cal proporcionar a la memòria l'adreça on es troba la informació desitjada. L'acció que fa la memòria consisteix a subministrar la informació continguda en l'adreça indicada.

b) Operació d'escriptura: en aquesta operació cal subministrar a la memòria la informació que s'ha d'emmagatzemar i l'adreça de memòria on es vol emmagatzemar. L'acció que es du a terme consisteix a enregistrar la informació a l'adreça especificada.

Usos de l'accés aleatori

L'accés aleatori se sol utilitzar en memòries RAM i ROM.

4) Associatiu. Es tracta d'un tipus de memòria d'accés aleatori on l'accés es fa basant-se en el contingut i no pas en l'adreça. S'especifica el valor que es vol localitzar i es compara aquest valor amb una part del contingut de cada posició de memòria; la comparació es fa simultàniament amb totes les posicions de la memòria.

Usos de l'accés associatiu

Aquest mètode d'accés se sol utilitzar en les memòries cau.

1.4. Organització de les dades d'una memòria

En aquest subapartat només ens referirem a la manera d'organitzar les dades en memòries que es trobin en el xip del processador i en la memòria interna. L'organització de la memòria externa es fa de manera diferent.

Bàsicament, els elements que hem de tenir en compte són els següents:

1) Paraula de memòria. És la unitat d'organització de la memòria des del punt de vista del processador, la mida de la paraula de memòria s'especifica en bytes o bits. És el nombre de bytes màxim que es poden llegir o escriure en un sol cicle d'accés a la memòria.

Exemple

Memòria de 2Kbytes amb una paraula de memòria de 2 bytes. Per tant, necessitarem 10 bits per a poder fer referència a les 1.024 (2^{10}) posicions de memòria que emmagatzemaran 2 bytes (16 bits) cadascuna.

Memòria interna

Adreça ← (10 bits) →	Paraula ← (16 bits) →
0000000000	**00000000 00000000**
0000000001	**00000000 00000000**
0000000010	**00000000 00000000**
0000000011	**00000000 00000000**
0000000100	**00000000 00000000**
...	**...**
↑ (1024 adreces) ↓	**contingut emmagatzemat a la memòria**
...	**...**
1111111100	**00000000 00000000**
1111111101	**00000000 00000000**
1111111110	**00000000 00000000**
1111111111	**00000000 00000000**

2) Unitat d'adreçament. La memòria interna es pot veure com un vector d'elements, una col·lecció de dades contigües, en què cada dada és accessible indicant la seva posició o adreça dins del vector.

La unitat d'adreçament especifica quina és la mida de cada element d'aquest vector, habitualment a la memòria s'accedeix com un vector de bytes, cada byte tindrà la seva adreça, encara que pot haver-hi sistemes que accedeixin a la memòria com un vector de paraules, on cada adreça correspongui a una paraula.

El nombre de bits utilitzats per a especificar una adreça de memòria fixen el límit màxim d'elements adreçables, la mida del mapa de memòria; si tenim n bits per a les adreces de memòria el nombre màxim d'elements adreçables serà de 2^n.

3) Unitat de transferència. En un accés a memòria es pot accedir a un byte o a diversos, amb un màxim que vindrà determinat pel nombre de bytes d'una paraula de memòria; és a dir, en un sol accés es llegeixen o escriuen un byte o diversos.

Quan s'especifica l'adreça de memòria a la qual es vol accedir s'accedeix a partir d'aquesta adreça a tants bytes com indiqui l'operació de lectura o escriptura.

En memòria externa, s'accedeix habitualment a un bloc de dades de mida molt superior a una paraula. En discos és habitual transferir blocs de l'ordre dels Kbytes.

Exemple

En els processadors x86 de 32 i 64 bits, la unitat d'adreçament és d'un byte, però la mida de la paraula de memòria és de 4 bytes (32 bits).

Els registres del processador (accessibles per al programador) habitualment tenen una mida igual a la mida de la paraula de memòria; per exemple, en un processador de 32 bits (com l'Intel 386) la mida dels registres era de 32 bits (4 bytes).

Els processadors x86-64 són processadors amb registres de 64 bits, però en canvi la mida de la paraula de memòria continua essent de 32 bits; això és així per a mantenir la compatibilitat amb processadors anteriors. Cal no oblidar que l'arquitectura x86-64 és una extensió de l'arquitectura de 32 bits x86-32.

La mida de la paraula de memòria dels processadors x86 de 32 i 64 bits és de 32 bits (4 bytes) i en un cicle de memòria es pot accedir a 1, 2 o 4 bytes.

En l'arquitectura CISCA la mida de paraula és també de 32 bits però accedim sempre a una paraula de 4 bytes.

1.4.1. Ordenació dels bytes en memòria

Encara que normalment la unitat d'adreçament de la memòria és el byte, és habitual que es puguin fer accessos a memòria en múltiples de byte, fins a la mida de la paraula (2, 4, i fins i tot 8 bytes). En aquest cas, només s'indica l'adreça del primer byte de la paraula i s'utilitzen dos mètodes a l'hora d'accedir a la paraula:

- ***Big-endian***: l'adreça especificada correspon al byte de més pes de la paraula.

- ***Little-endian***: l'adreça especificada correspon al byte de menys pes de la paraula.

Exemple

Suposem una memòria de capacitat reduïda (només 256 bytes) en què la mida de la paraula a la qual es pot accedir és de 16 bits (2 bytes). Les adreces de memòria seran de 8 bits; per a accedir a una paraula de memòria s'indica només l'adreça del primer byte.

Adreça		Valor
0	00000000	01100110
1	00000001	11100011
...	...	...
14	**00001110**	**00000000**
15	**00001111**	**11111111**
...	...	...
254	11111110	00001111
255	11111111	11001100

Si indiquem l'adreça 00001110 (adreça 14), obtenim els valors de les posicions de memòria 14 i 15 així:

1) ***Little-endian***: l'adreça 14 correspon al byte de menys pes de la paraula de 16 bits. Obtenim el valor:

1111111100000000 (adreces 15 i 14)

2) ***Big-endian***: l'adreça 14 correspon al byte de més pes de la paraula de 16 bits. En aquest cas obtenim el valor:

0000000011111111 (adreces 14 i 15)

1.5. Temps d'accés i velocitat

En memòries d'accés aleatori, memòria RAM, el **temps d'accés** (o **latència**) és el temps que transcorre des que una adreça de memòria és visible per als circuits de la memòria fins que la dada és emmagatzemada (escriptura) o està disponible per a ser utilitzada (lectura). En memòries d'accés no aleatori (discos) és el temps necessari perquè el mecanisme de lectura o escriptura se situï en la posició necessària per a començar la lectura o escriptura.

En memòries d'accés aleatori, el temps d'un **cicle de memòria** es considera el temps d'accés més el temps necessari abans que es pugui començar un segon accés a la memòria.

Finalment, la **velocitat de transferència** és la velocitat a què es pot llegir o escriure una dada de memòria. En les memòries d'accés aleatori serà l'invers del temps de cicle. La velocitat de transferència es mesura en bytes per segon; és habitual indicar la velocitat d'una memòria en MiB/segon o GiB/segon.

1.6. Cost

Considerem el cost per unitat d'emmagatzematge (cost per bit). Podem observar que hi ha una relació directament proporcional entre la velocitat i el cost/bit: a mesura que augmenta la velocitat augmenta també el cost/bit. Això implica que, amb un pressupost fixat, podrem adquirir memòries molt ràpides però relativament petites, o memòries més lentes, però de molta més capacitat.

Exemple de cost

L'any 2010 amb 100 € es podia aconseguir una memòria RAM de 4GB amb un temps d'accés de 5ns o un disc magnètic d'1TB (1.000 GB) amb un temps d'accés de 5 ms (5.000.000 ns), per tant, pel mateix cost podem tenim una memòria mil cops més gran, però un milió de vegades més lenta.

1.7. Característiques físiques

La memòria es pot classificar segons característiques físiques diferents; bàsicament podem distingir dues classificacions. La primera distingeix entre:

- **Memòria volàtil**: memòria que necessita un corrent elèctric per a mantenir el seu estat; aquestes memòries inclouen registres, memòria cau i memòria principal.

- **Memòria no volàtil**: manté l'estat sense necessitat de corrent elèctric, inclou memòries de només lectura, memòries programables, memòria flaix, dispositius d'emmagatzematge magnètic i òptic.

La segona classificació distingeix entre:

- **Memòria de semiconductors**: és una memòria que utilitza elements semiconductors, transistors, en la seva construcció; inclou: registres, memòria cau, memòria principal, memòries de només lectura, memòria flaix.

- **Memòria magnètica**: utilitza superfícies imantades per a guardar la informació; dins d'aquesta categoria s'inclouen bàsicament discos i cintes magnètiques.

- **Memòria òptica**: utilitza elements d'emmagatzematge que poden ser llegits i escrits mitjançant llum làser; s'inclouen dispositius de CD, DVD, Bluray.

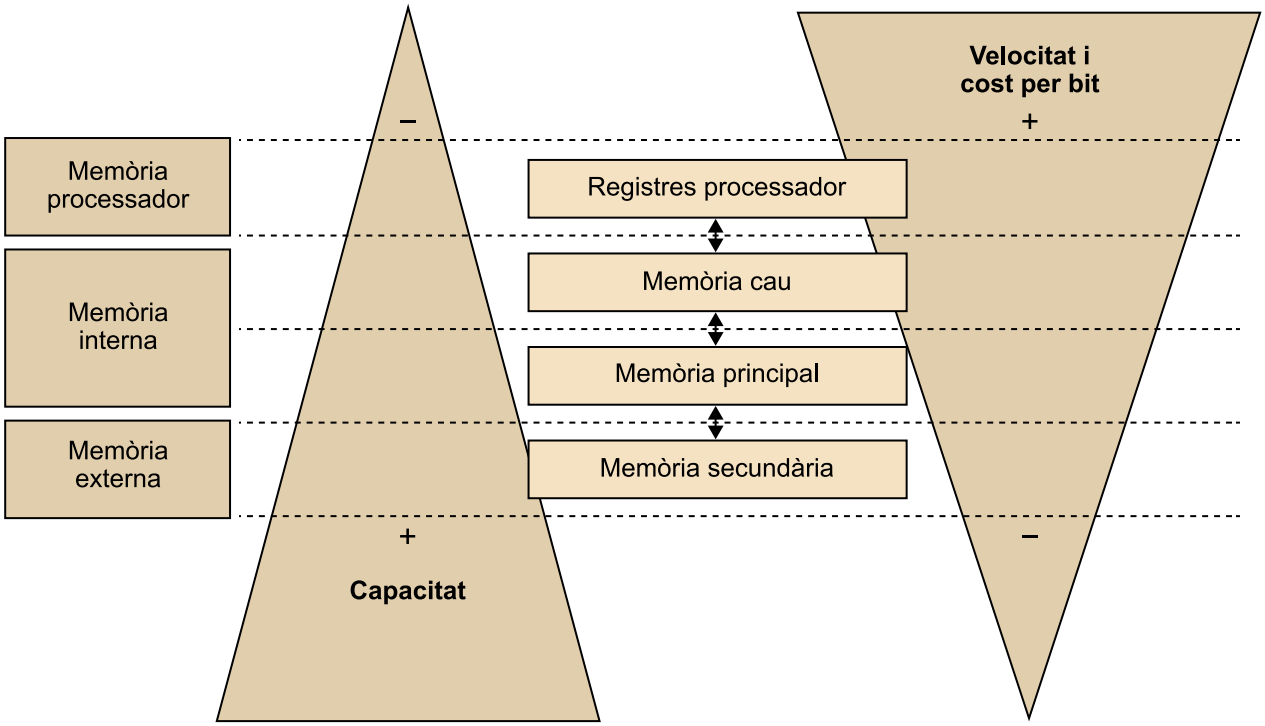
2. Jerarquia de memòries

L'objectiu en el disseny del sistema de memòria d'un computador és que tingui una gran capacitat i un temps d'accés reduït amb el preu més baix possible. Com que no hi ha cap tecnologia que compleixi simultàniament aquests requisits, la memòria del computador s'estructura en diversos nivells amb l'objectiu d'aconseguir unes prestacions millors, i forma el que anomenarem **jerarquia de memòries**.

En una jerarquia de memòries s'utilitzen diferents tipus de memòria amb característiques diferents de capacitat, velocitat i cost, que dividirem en nivells diferents: memòria del processador, memòria interna i memòria externa.

Cada nivell de la jerarquia es caracteritza també per la distància a la qual es troba del processador. Els nivells més propers al processador són els primers en ser utilitzats; això és així perquè també són els nivells amb una velocitat més elevada.

A continuació es mostra quina és la variació de la capacitat, velocitat i cost per bit per als nivells típics d'una jerarquia:



L'objectiu final de la jerarquia de memòries és aconseguir que, quan el processador accedeixi a una dada, aquesta es trobi en el nivell més ràpid de la jerarquia. Aconseguim tenir una memòria amb un cost moderat, una velocitat propera a la del nivell més ràpid i la capacitat del nivell més gran.

Cada nivell de la jerarquia de la memòria es relaciona només amb el nivell superior i inferior, llevat de casos excepcionals. El processador només té accés als registres i obté les dades de memòria mitjançant la memòria cau.

Per això, quan el processador necessita una dada i aquesta no està disponible a la memòria cau, s'ha de portar a la cau des del nivell en què estigui disponible.

D'altra banda, si el processador modifica una dada en un nivell de la jerarquia de memòries, s'ha de garantir que la modificació s'efectua a tota la resta de nivells en què la dada es trobi emmagatzemada. Si no fos així, la següent vegada que s'accedís a aquesta dada, podria ser que s'agafés un valor incorrecte. Aquest problema es denomina *coherència*.

Com que els nivells de memòria més propers al processador no són gaire grans, es podria pensar que es perd molt de temps portant les dades des d'un nivell fins a un altre, ja que aquest moviment ha de ser constant. En realitat, això no és cert: les dades es reutilitzen amb molta freqüència, per la qual cosa és útil que siguin al nivell més proper al processador. Més endavant estudiarem per què es produeix aquesta reutilització i, per tant, per què és efectiva la jerarquia de memòries.

A continuació hi ha una descripció de les característiques principals dels diferents nivells d'una jerarquia de memòria.

2.1. Registres

El registre és l'espai de memòria que hi ha dins del processador, s'integra dins del mateix xip del processador. S'utilitzen cel·les de memòria de tipus estàtic, SRAM, per a la seva implementació.

És l'espai de memòria en el qual el processador pot accedir més ràpidament a les dades. Aquest espai de memòria és accessible al programador de llenguatge d'assemblador i, si es gestiona bé, permet minimitzar el nombre d'accessos a la memòria interna que són força més lents.

2.2. Memòria interna

La memòria interna en un computador modern està formada típicament per dos nivells principals: memòria cau i memòria principal. En els computadors actuals és freqüent trobar la memòria cau dividida també en nivells.

2.2.1. Memòria cau

Les **memòries cau** són memòries de capacitat reduïda, però més ràpides que la memòria principal, que utilitzen un mètode d'accés associatiu. Es poden trobar dins del xip del processador o a prop, estan dissenyades per a reduir el temps d'accés a la memòria. A la memòria cau s'emmagatzemen les dades que es preveu que seran utilitzades més habitualment, de manera que sigui possible reduir el nombre d'accessos que ha de fer el processador a la memòria principal (ja que el temps d'accés a la memòria principal sempre és superior al temps d'accés a la memòria cau).

No és accessible per part del programador, és gestionada pel maquinari i el sistema operatiu i s'implementa utilitzant tecnologia SRAM.

Els processadors moderns utilitzen diferents nivells de memòria cau, és el que es coneix com a **memòria cau de primer nivell**, **segon nivell**, etc. Actualment és habitual disposar de fins a tres nivells de memòria cau, referits com a L1, L2 i L3. Cada cop és més habitual que alguns d'aquests nivells s'implementin dins del xip del processador i que el nivell més proper al processador estigui dividit en dues parts, una de dedicada a les instruccions i una altra de dedicada a les dades.

Memòria cau dels processadors d'Intel i AMD

Els darrers processadors d'Intel i AMD inclouen tres nivells de memòria cau: un primer nivell (L1) dividit en memòria cau d'instruccions i memòria cau de dades, i els altres nivells (L2 i L3), unificats. Els processadors actuals tenen un disseny multinucli (*multicore*); un processador integra en un sol xip diversos nuclis completament funcionals, cada nucli disposa d'una memòria cau de primer nivell (L1) i de segon nivell (L2), i la memòria cau de tercer nivell (L3) és compartida per tots els nuclis del processador. En aquests processadors tota la memòria cau s'integra dins del xip del microprocessador.

2.2.2. Memòria principal

En la memòria principal s'hi emmagatzemen els programes que s'han d'executar i les seves dades, és la memòria visible al programador, mitjançant el seu espai d'adreces.

La memòria principal s'implementa utilitzant diferents xips connectats a la placa principal del computador, té una capacitat molt més elevada que la memòria cau (de l'ordre de Gbytes o de Tbytes en supercomputadors).

Utilitza tecnologia DRAM (Dynamic RAM), que és més lenta que la SRAM, però amb una capacitat d'integració molt més elevada, fet que permet obtenir més capacitat en menys espai.

2.3. Memòria externa

La memòria externa correspon a dispositius d'emmagatzematge secundari: discos magnètics, cintes magnètiques, discos òptics, dispositius de memòria flaix, etc., i també es poden considerar sistemes d'emmagatzematge en xarxa.

Aquests dispositius són gestionats pel sistema de fitxers del sistema operatiu mitjançant el sistema d'entrada/sortida.

Els dispositius que formen la memòria externa es connecten al computador amb algun tipus de bus (sèrie o paral·lel). Aquests dispositius es poden trobar físicament dins del computador connectats per busos interns del computador (IDE, SATA, SCSI, etc.), o poden ser fora del computador connectats per busos externs (USB, Firewire, eSATA, Infiniband, etc.).

2.4. Memòria virtual

Diem que un computador utilitza memòria virtual quan les adreces de memòria dels programes es refereixen a un espai de memòria superior a l'espai de memòria físic, espai de memòria principal.

La memòria virtual allibera el programador de les restriccions de la memòria principal. En aquests computadors diferenciem entre el mapa d'adreces lògiques o virtuals (les adreces que utilitzen els programes) i el mapa d'adreces físiques o reals (les adreces de la memòria principal). L'espai de memòria virtual utilitza com a suport un dispositiu d'emmagatzematge extern (habitualment un disc magnètic), mentre que l'espai de memòria físic es correspon amb la memòria principal del computador.

Actualment, pràcticament tots els computadors utilitzen memòria virtual. La utilització de la memòria virtual implica resoldre dos problemes: la traducció d'adreces lògiques a adreces físiques i l'assignació d'espai de memòria físic als programes que s'han d'executar. La gestió de la memòria virtual la fa el sistema operatiu.

Nota

En aquest mòdul ens centrarem a analitzar com es gestiona l'espai d'adreces físiques del computador; la gestió de l'espai d'adreces virtuals correspon a l'àmbit dels sistemes operatius i no l'analitzarem.

2.5. Funcionament de la jerarquia de memòries

Els dos factors bàsics que fan que l'esquema de jerarquia de memòries funcioni satisfactòriament en un computador són els següents:

- El flux de dades entre els nivells de la jerarquia de memòries es pot fer en paral·lel amb el funcionament normal del processador.

- El principi de **proximitat referencial** dels programes.

> El codi dels programes s'organitza en subrutines, té estructures iteratives i treballa amb conjunts de dades agrupades. Això, unit al fet que l'execució del codi és seqüencial, produeix que durant un interval de temps determinat s'utilitzi només una petita part de tota la informació emmagatzemada: aquest fet s'anomena **proximitat referencial**.

A causa d'aquesta característica, s'ha provat empíricament que aproximadament el 90% de totes les instruccions executades corresponen al 10% del codi d'un programa.

Distingim dos tipus de proximitat referencial:

1) Proximitat temporal. És quan, en un interval de temps determinat, la probabilitat que un programa accedeixi de manera repetida a les mateixes posicions de memòria és molt gran.

La proximitat temporal és deguda principalment a les estructures iteratives; un bucle executa les mateixes instruccions repetidament, de la mateixa manera que les crides repetitives a subrutines.

2) Proximitat espacial. És quan, en un interval de temps determinat, la probabilitat que un programa accedeixi a posicions de memòria properes és molt gran.

La proximitat espacial és deguda principalment al fet que l'execució dels programes és seqüencial –s'executa una instrucció darrere l'altra llevat de les bifurcacions–, i també a la utilització d'estructures de dades que estan emmagatzemades en posicions de memòria contigües.

3. Memòria cau

La memòria cau se situa entre la memòria principal i el processador, pot estar formada per un nivell o diversos. En aquest apartat explicarem el funcionament de la memòria cau considerant un únic nivell, però el funcionament és semblant si en té diversos.

La memòria cau té un temps d'accés inferior al de la memòria principal amb l'objectiu de reduir el temps d'accés mitjà a les dades, però també té una mida molt més reduïda que la memòria principal. Si una dada és a la memòria cau, és possible proporcionar-la al processador sense accedir a la memòria principal, si no, primer es porta la dada de la memòria principal a la memòria cau i després es proporciona la dada al processador.

Si, en la majoria d'accessos a memòria, la dada és a la memòria cau, el temps d'accés mitjà serà proper al temps d'accés de la memòria cau. Això és factible per la característica de proximitat referencial dels programes.

Per a treballar amb memòria cau, la memòria principal s'organitza en **blocs** de paraules, de manera que quan cal portar dades de la memòria principal a la memòria cau es porta un bloc sencer de paraules de memòria, no es treballa amb paraules individuals.

La memòria cau també s'organitza en blocs que s'anomenen **línies**. Cada línia està formada per un conjunt de paraules (el mateix nombre de paraules que tingui un bloc de memòria principal), més una etiqueta composta per uns quants bits. El contingut de l'etiqueta permetrà saber quin bloc de la memòria principal es troba en cada línia de la memòria cau en un moment donat.

Adreça	Memòria principal	Bloc	Paraula
0		Bloc 0	0
1			1
...			...
$k-1$			$k-1$
k		Bloc 1	0
$k+1$			1
...			...
$2k-1$			$k-1$
...	...	...	...
2^n-k		Bloc $(2^n/k)-1$	0
			1
...			...
2^n-1			$k-1$

Memòria cau				
	Etiqueta del bloc x	Paraules de la línia		
Línia		0	...	$k-1$
0			...	
1			...	
...	...		...	
$m-1$			...	

A l'esquerra, memòria principal de 2^n paraules, organitzada en $2^n/k$ blocs de k paraules. Memòria cau amb m línies, amb k paraules per línia.
A la dreta, memòria cau amb m línies, amb k paraules per línia, l'*etiqueta del bloc x* identifica quin bloc de la memòria principal tenim emmagatzemat en aquella línia de la memòria cau.

3.1. Encerts i fallades

Cada vegada que el processador vol accedir a una paraula de memòria, primer s'accedeix a la memòria cau, si la paraula de memòria es troba emmagatzemada a la memòria cau es proporciona al processador i direm que s'ha produït un **encert**; en cas contrari, es porta el bloc de dades de la memòria principal que conté la paraula de memòria cap a la memòria cau i, quan la paraula ja és a la memòria cau, es proporciona al processador; en aquest cas direm que s'ha produït una **fallada**.

Quan hi ha una fallada, el maquinari de la memòria cau ha de fer la seqüència de tasques següent:

1) Sol·licitar a la memòria principal el bloc on hi ha la dada que ha produït la fallada.

2) Portar el bloc de dades sol·licitat a la memòria cau. Les operacions realitzades per aquesta tasca dependran de les polítiques d'assignació i algorismes de reemplaçament que veurem més endavant.

3) El processador obté la dada de la memòria cau com si s'hagués produït un encert.

Un accés amb fallada en la memòria cau pot ser força més costós en temps que un accés amb encert, per això és molt important tenir un nombre reduït de fallades.

3.2. Rendiment de la memòria cau

A partir del concepte d'encert i fallada es defineixen els paràmetres que utilitzarem per a avaluar el rendiment d'una memòria cau: taxa de fallades i temps mitjà d'accés.

La taxa de fallades es defineix de la manera següent:

T_f = Nombre de fallades / Nombre d'accessos a la memòria

D'altra banda es defineix la taxa d'encerts així:

T_e = Nombre d'encerts / Nombre d'accessos a la memòria = $1 - T_f$

Un dels objectius del disseny del sistema de memòria és obtenir una taxa de fallades tan baixa com sigui possible. Generalment s'espera que sigui inferior al 10%.

Es pot calcular també el temps mitjà d'accés t_m a partir de la taxa de fallades i de la taxa d'encerts, coneixent el temps d'accés en cas d'encerts t_e i el temps d'accés en cas de fallada t_f, ja que el temps de fallada té en compte el temps necessari per a portar tot un bloc de la memòria principal a la memòria cau i el temps d'accés a la dada.

$$t_m = T_f \times t_f + T_e \times t_e = T_f \times t_f + (1 - T_f) \times t_e = T_f \times (t_f - t_e) + t_e$$

Si la taxa de fallades és zero, el temps mitjà d'accés a memòria és igual al temps d'accés a la memòria cau.

3.3. Línia de memòria cau

Hem vist que la memòria cau s'organitza en línies; una línia està formada bàsicament per un conjunt de paraules més una etiqueta que identifica quin bloc de la memòria principal ocupa aquella línia de la memòria cau.

La línia de memòria cau és la unitat de transferència entre la memòria cau i la memòria principal. La mida de la línia és un dels paràmetres fonamentals del disseny de la memòria cau. Cal decidir quantes paraules s'emmagatzemaran en una línia de memòria cau, és a dir, quina és la mida d'una línia.

Hem vist que les dades es porten de la memòria principal a la memòria cau quan hi ha una fallada. Si es produeix una fallada, es porta a la memòria cau la dada que l'ha provocat i la resta de dades del bloc de memòria on es troba aquesta dada. D'aquesta manera, s'espera que els accessos següents siguin encerts en la memòria cau.

La mida de la línia és d'uns quants bytes d'informació, una mida habitual és entre els 32 bytes i 128 bytes. Augmentar la mida de la línia permet aprofitar la localitat espacial, però fins a cert punt. Quan es produeix una fallada, el temps necessari per a portar una línia més gran augmenta, a més, disminueix el nombre de línies disponibles de la memòria cau (la mida de la memòria cau és fixa) i tindrem més competència per a aconseguir un bloc, això farà que es treguin de la cau línies que encara no s'han utilitzat en la seva totalitat i es reduirà l'efecte de la localitat espacial, i tot això pot representar un augment en la taxa de fallades.

3.4. Polítiques d'assignació

El nombre de línies disponibles en la memòria cau és sempre molt més petit que el nombre de blocs de memòria principal. En conseqüència, la memòria cau, a més de la informació emmagatzemada, ha de mantenir alguna informació que relacioni cada posició de la memòria cau amb la seva adreça de la memòria principal.

Per a accedir a una dada se n'especifica l'adreça a la memòria principal; a partir d'aquesta adreça cal verificar si aquesta dada és a la memòria cau. Aquesta verificació la farem a partir del camp *etiqueta* de la línia de la memòria cau que indica quin bloc de memòria principal es troba en cadascuna de les línies de la memòria cau.

La política d'assignació determina on podem posar un bloc de la memòria principal dins de la memòria cau i condiciona com trobar una dada dins la memòria cau.

Es defineixen tres polítiques d'assignació diferents per a emmagatzemar dades dins d'una memòria cau:

1) Política d'assignació directa: un bloc de la memòria principal només pot ser en una única línia de la memòria cau. La memòria cau d'assignació directa és la que té la taxa de fallades més alta, però s'utilitza molt perquè és la més barata i fàcil de gestionar.

2) Política d'assignació completament associativa: un bloc de la memòria principal pot ser en qualsevol línia de la memòria cau. La memòria cau completament associativa és la que té la taxa de fallades més baixa. No obstant això, no se sol utilitzar perquè és la més cara i complexa de gestionar.

3) Política d'assignació associativa per conjunts: un bloc de la memòria principal pot ser en un subconjunt de les línies de la memòria cau, però dins del subconjunt pot trobar-se en qualsevol posició.

La memòria cau associativa per conjunts és una combinació de la memòria cau d'assignació completament associativa i la memòria cau d'assignació directa. El nombre d'elements de cada subconjunt no acostuma a ser gaire gran, un nombre habitual d'elements és entre 4 i 64. Si el nombre d'elements del subconjunt és *n*, la memòria cau es denomina *n-associativa*.

Línies de la memòria cau on podem assignar el bloc x segons les diferents polítiques d'assignació

Adreça	Memòria principal	Bloc
0		Bloc 0
1		
...		
$k-1$		
k		Bloc 1
$k+1$		
...		
$2 \cdot k-1$		
...	...	...
$x \cdot k+0$		Bloc x
$x \cdot k+(k-1)$		
...	...	...
2^n-k		Bloc $(2^n / k)-1$
...		
2^n-1		

Memòria cau d'accés directe		
Línia	**Etiqueta**	**Paraules del bloc**
0		
...		...
Línia assignada al bloc x		
...		...
$m-1$		

Memòria cau d'assignació completament associativa		
Línia		**Contingut de la cau**
0		
...	...	...
Bloc x assignat a qualsevol línia		
...	...	...
$m-1$		

Memòria cau d'assignació associativa per conjunts		
Línia		**Contingut de la cau**
0		
...	...	...
...	...	...
Conjunt de línies assignat al bloc x		
		...
...	...	...
...	...	...
$m-1$		

3.4.1. Memòria cau d'assignació directa

Per a utilitzar aquest tipus de memòria cau, s'assigna cada bloc de la memòria principal a una única línia de la memòria cau.

Per a relacionar una línia de la memòria cau amb un bloc de la memòria principal a partir de l'adreça especificada per a accedir a una paraula de la memòria principal, hem de determinar a quin bloc pertany l'adreça. Es divideix l'adreça en dues parts: número de bloc, que correspon a la part més significativa de l'adreça, i número de paraula, que correspon a la part menys significativa.

Si tenim una memòria principal de 2^n paraules i una memòria cau de 2^m línies de 2^k paraules per línia, la memòria principal es dividirà en blocs de 2^k paraules. Una adreça de memòria estarà formada per n bits, utilitzarà els k bits menys significatius per al número de paraula i els $n - k$ bits restants per al número de bloc.

Aclariment

No cal que el nombre de línies de la cau sigui una potència de 2. Tot i que habitualment ho és, pot ser un valor on: $2^{(m-1)} <$ nre. línies $< 2^{(m)}$, i utilitzarem m bits per a codificar el número de línia.

Per exemple: si tenim una cau de 5 línies ($2^2 < 5 < 2^3$), codificarem el número de línia utilitzant 3 bits per al número de línia.

Adreça de memòria	
Número de bloc	Número de paraula
$n - 1$ ... k	$k - 1$ ... 0
← ($n - k$ bits) →	← (k bits) →

Càlcul del número de bloc

A partir d'una adreça de memòria a es pot calcular el número de bloc b fent l'operació següent: $b = a$ div 2^k, en què *div* és la divisió entera.

El número de paraula p es pot calcular fent l'operació següent: $p = a$ mod 2^k, en què *mod* és el residu de la divisió entera.

Per a determinar a quina línia de la memòria cau podem assignar cada bloc, cal dividir el número de bloc en dues parts: una etiqueta, que correspon a la part més significativa del número de bloc, i un número de línia, corresponent a la part menys significativa.

Si tenim un número de bloc que utilitza $n - k$ bits, d'aquests $n - k$ bits utilitzarem m bits per a especificar el número de línia i la resta de bits ($n - k - m$) per a especificar l'etiqueta.

Adreça de memòria		
Número de bloc		Número de paraula
Etiqueta	Número de línia	
$n - 1$ ... $k + m$	$k + m - 1$ ... k	$k - 1$ ... 0
← ($n - k - m$ bits) →	← (m bits) →	← (k bits) →

Càlcul del número d'etiqueta

A partir del número de bloc b es pot calcular l'etiqueta e fent l'operació següent: $e = b$ div 2^m, en què *div* és la divisió entera.

El número de línia l es pot calcular fent l'operació següent: $l = b$ mod 2^m, en què *mod* és el residu de la divisió entera.

Tenim 2^{n-k} blocs a la memòria principal i 2^m línies a la memòria cau ($2^{n-k} > 2^m$), per tant a cada línia de la memòria cau hi podem assignar 2^{n-k-m} (= $2^{n-k} / 2^m$) blocs diferents. Només un d'aquests 2^{n-k-m} pot ser a la memòria cau en cada moment.

El número de línia indicarà en quina de les 2^m línies de la memòria cau es pot trobar el bloc de dades al qual volem accedir de la memòria principal. L'etiqueta ens permetrà saber si el bloc al qual volem accedir de la memòria principal és el bloc que en aquest moment està emmagatzemat en aquella línia de la memòria cau.

Quan es porta un bloc de la memòria principal a la línia corresponent de la memòria cau, el número de l'etiqueta del bloc s'emmagatzema en el camp etiqueta de la línia, així podrem saber quin dels 2^{n-k-m} blocs està emmagatzemat en aquella línia de la cau. El camp etiqueta és el que ens permet identificar de manera única cada un dels blocs que podem assignar a una mateixa línia de la memòria cau.

De manera general, es pot dir que si tenim una memòria cau de 2^m línies, els blocs de memòria principal que es poden trobar en cada una de les línies de la memòria cau són els que es mostren a la taula següent.

Número de línia	Blocs assignats
0	0, 2^m, $2 \times (2^m)$, ...
1	1, $2^m + 1$, $2 \times (2^m) + 1$, ...
2	2, $2^m + 2$, $2 \times (2^m) + 2$, ...
...	...
$2^m - 1$	$2^m - 1$, $2^m + (2^m - 1)$, $2 \times 2^m + (2^m - 1)$, ...

Per a determinar si un accés a una adreça de memòria produeix un encert en la memòria cau cal fer el següent: a partir de l'adreça de memòria es determina quin és el seu número de línia (bits $k + m - 1$ .. k), el qual s'utilitza com a índex per a accedir a la cau i obtenir l'etiqueta que identifica el bloc que està emmagatzemat en aquesta línia i que es compara amb el camp etiqueta de l'adreça

(bits n – 1 .. k + m); si coincideixen es tracta d'un encert, llavors s'utilitza el número de paraula (bits k –1 .. 0) per a obtenir la paraula sol·licitada i servir-la al processador.

Si l'etiqueta de l'adreça i l'etiqueta de la línia no coincideixen, es tracta d'una fallada i caldrà portar tot el bloc de memòria principal a la memòria cau reemplaçant el bloc que hi tenim actualment emmagatzemat.

Exemple

Si tenim una memòria principal de 2^{16} (64 K) paraules i una memòria cau de 2^{10} (1.024) paraules organitzada en 2^4 (16) línies de 2^6 (64) paraules per línia, la memòria principal es dividirà en blocs de 2^6 (64) paraules. Una adreça de memòria tindrà 16 bits, els 6 bits menys significatius per al número de paraula i els 16 – 6 = 10 bits restants per al número de bloc; en total tindrem 2^{10} (1.024) blocs de 2^6 (64) paraules. Les adreces es dividiran de la manera següent:

Adreça de memòria	
Número de bloc	Número de paraula
15 6	5 0

El número de bloc de 10 bits es divideix en etiqueta i número de línia: 4 bits per a la línia, ja que hi ha 2^4 línies i 10 – 4 = 6 bits per a l'etiqueta. Així les adreces de memòria principal es dividiran de la manera següent:

Adreça de memòria		
Número de bloc		Número de paraula
Etiqueta	Número de línia	
15 10	9 6	5 0

L'assignació de blocs de la memòria principal a la memòria cau seria de la manera següent:

Número de línia	Blocs assignats
0	0, 16, 32, 48, 64, ..., 1008
1	1, 17, 33, 49, 65, ..., 1009
2	2, 18, 34, 50, 66, ..., 1010
...	...
15	15, 31, 63, 79, ..., 1023

A cada línia de la memòria cau podem assignar 2^6 = 64 blocs diferents.

El mapa d'adreces de memòria principal quedaria de la manera següent:

	Adreça de memòria		
	Número de bloc		Número de paraula
	Etiqueta	Número de línia	
Bloc 0	0	0	0
			...
			63 ($2^6 - 1$)
Bloc 1	0	1	0
			...
			63 ($2^6 - 1$)
...			
Bloc 15	0	15	0
			...
			63 ($2^6 - 1$)
Bloc 16	1	0	0
			...
			63 ($2^6 - 1$)
...			
...			
...			
Bloc 1007	62	15	0
			...
			63 ($2^6 - 1$)
Bloc 1008	63	0	0
			...
			63 ($2^6 - 1$)
...			
Bloc 1023	63	15	0
			...
			63 ($2^6 - 1$)

Es pot observar que tots els blocs que poden ser en una mateixa línia de la cau tenen un valor d'etiqueta diferent; es podrà utilitzar el valor de l'etiqueta per a saber quin bloc en concret es troba en cada línia de la memòria cau. La taula anterior mostra que tots els blocs ombrejats estan assignats a la línia 0 de la memòria cau i podrem saber quin es troba a la memòria cau pel número de l'etiqueta.

A continuació es mostra un contingut possible de la memòria cau:

	Número de bloc	
	Etiqueta	Línia
Bloc 16	1	0
Bloc 1	0	1
	...	...
Bloc 1023	63	15

Memòria cau				
Línia	Etiqueta	Paraules de la línia		
		0	...	$2^k - 1$
0	1	M(1024)	...	M(1087)
1	0	M(64)	...	M(127)
...	...	...	...	...
15	63	M(65472)	...	M(65535)

A la línia 0 tenim el bloc 16 (etiqueta: 1) a la línia 1 tenim el bloc 1 (etiqueta: 0) i a la línia 15 tenim el bloc 1023 (etiqueta: 63). M(x) indica que en aquesta paraula de la línia de la cau hi ha emmagatzemada la paraula de memòria amb l'adreça x.

A continuació es mostra la descomposició d'una de les adreces del bloc 16 que es troba a la memòria cau.

	Adreça de memòria		
	64 = 000001 0000 000000		
	Número de bloc		Número de paraula
	Etiqueta	Número de línia	
Bloc 16	1 = 000001	0000	000000

3.4.2. Memòria cau completament associativa

A diferència de la memòria cau directa, un bloc de memòria principal es pot trobar en qualsevol línia de la memòria cau.

Per a relacionar una línia de la memòria cau amb un bloc de la memòria principal a partir de l'adreça especificada per a accedir a una paraula de la memòria principal, hem de determinar a quin bloc pertany l'adreça. Es divideix l'adreça en dues parts: número de bloc que correspon a la part més significativa de l'adreça i número de paraula que correspon a la part menys significativa.

Si tenim una memòria principal de 2^n paraules i una memòria cau de 2^m línies de 2^k paraules per línia, la memòria principal es dividirà en blocs de 2^k paraules. Una adreça de memòria estarà formada per n bits, utilitzarà els k bits menys significatius per al número de paraula i els $n - k$ bits restants per al número de bloc.

Adreça de memòria	
Número de bloc	Número de paraula
$n - 1$ k	$k - 1$ 0
← ($n - k$ bits) →	← (k bits) →

Càlcul del número de bloc

A partir d'una adreça de memòria a es pot calcular el número de bloc b fent l'operació següent: $b = a \text{ div} 2^k$, en què *div* és la divisió entera.

El número de paraula p es pot calcular fent l'operació següent: $p = a \bmod 2^k$, en què *mod* és el residu de la divisió entera.

Cal tenir present que a cada línia de la memòria cau hi podem assignar qualsevol bloc de la memòria principal i hem de poder saber quin es troba en cada moment en la memòria cau.

Quan es porta un bloc de memòria principal a la memòria cau, cal decidir quina línia reemplacem amb el nou bloc de dades; per a prendre aquesta decisió es poden utilitzar diferents algorismes de reemplaçament que explicarem més endavant. El número de bloc de l'adreça de memòria s'emmagatzema en el camp etiqueta de la línia; així podrem saber quin bloc està emmagatzemat en cadascuna de les línies de la cau.

Per a determinar si un accés a una adreça de memòria produeix un encert en la memòria cau cal fer el següent:

A partir de l'adreça de memòria es determina quin és el seu número de bloc (bits $n - 1$ .. k), es compara simultàniament el número de bloc d'aquesta adreça amb el camp etiqueta de totes les línies de la memòria cau, si es produeix una coincidència significa que s'ha produït un encert, llavors s'utilitza el número de paraula (bits $k - 1$ .. 0) per a obtenir la paraula sol·licitada i servir-la al processador.

Si el número de bloc de l'adreça no coincideix amb cap etiqueta de la memòria cau, es tracta d'una fallada i caldrà portar tot el bloc de memòria principal a la memòria cau reemplaçant un dels blocs que hi tenim actualment emmagatzemats.

Exemple

Tenim una memòria principal de 2^{16} (64 K) paraules i una memòria cau de 2^{10} (1.024) paraules organitzada en 2^6 (64) línies de 2^4 (16) paraules per línia.

La memòria principal es dividirà en blocs de 2^4 (16) paraules. Una adreça de memòria tindrà 16 bits, els 4 bits menys significatius per al número de paraula i els $16 - 4 = 12$ bits restants per al número de bloc; en total tindrem 2^{12} (4.096) blocs de 2^4 (16) paraules. Les adreces es dividiran de la manera següent:

Adreça de memòria	
Número de bloc	Número de paraula
15 4	3 0

A continuació es mostra un possible contingut de la memòria cau:

	Número de bloc
	Etiqueta
Bloc 4095	4095
Bloc 1024	1024
	...
Bloc 1	1

Memòria cau				
Línia	Etiqueta	Paraules de la línia		
		0	...	$2^k - 1$
0	4095	M(65520)	...	M(65535)
1	1024	M(16384)	...	M(16399)
...	...	...	...	...
63	1	M(16)	...	M(31)

M(x) indica que en aquesta paraula de la línia de la cau hi ha emmagatzemada la paraula de memòria amb l'adreça x.

A continuació es mostra la descomposició d'una de les adreces del bloc 16 que es troba a la memòria cau.

	Adreça de memòria	
	16384 = 010000000000 0000	
	Número de bloc	Número de paraula
Bloc 1024	1024 = 010000000000	0000

3.4.3. Memòria cau associativa per conjunts

Un bloc de la memòria principal pot trobar-se en un únic conjunt de línies de la memòria cau, però dins del conjunt pot trobar-se en qualsevol línia.

Per a relacionar una línia de la memòria cau amb un bloc de la memòria principal a partir de l'adreça especificada per a accedir a una paraula de la memòria principal, hem de determinar a quin bloc pertany l'adreça. Es divideix l'adreça en dues parts: número de bloc que correspon a la part més significativa de l'adreça i número de paraula que correspon a la part menys significativa.

Si tenim una memòria principal de 2^n paraules i una memòria cau de 2^m línies de 2^k paraules per línia, la memòria principal es dividirà en blocs de 2^k paraules. Una adreça de memòria estarà formada per n bits, utilitzarà els k bits menys significatius per al número de paraula i els $n - k$ bits restants per al número de bloc.

Adreça de memòria	
Número de bloc	Número de paraula
$n-1$ k	$k-1$ 0
← ($n - k$ bits) →	← (k bits) →

Càlcul del número de bloc

A partir d'una adreça de memòria a es pot calcular el número de bloc b fent l'operació següent: $b = a \text{ div} 2^k$, en què *div* és la divisió entera.

El número de paraula p es pot calcular fent l'operació següent: $p = a \bmod 2^k$, en què *mod* és el residu de la divisió entera.

En una memòria cau associativa per conjunts s'ha d'organitzar la memòria cau en conjunts; dividim les 2^m línies de la memòria cau en 2^c conjunts de $\omega = 2^{m-c} = (2^m/2^c)$ línies cadascun, i direm així que és una memòria cau ω -associativa.

Si tenim tants conjunts com línies ($2^c = 2^m$) i cada conjunt té una sola línia ($\omega = 1$), som davant del mateix cas que una memòria cau d'assignació directa; si tenim un sol conjunt ($2^c = 1$) de 2^m línies ($\omega = 2^m$), es tracta d'una memòria completament associativa.

Per a determinar a quin conjunt de la memòria cau podem assignar cada bloc de la memòria principal, cal dividir el número de bloc en dues parts: una etiqueta que correspon a la part més significativa del número de bloc, i un número de conjunt corresponent a la part menys significativa.

Si tenim un número de bloc que utilitza $n - k$ bits, d'aquests $n - k$ bits utilitzarem c bits per a especificar el número de conjunt, i la resta de bits ($n - k - c$) per a especificar l'etiqueta.

Adreça de memòria		
Número de bloc		Número de paraula
Etiqueta	Número de conjunt	
$n-1$ ← ($n-k-c$ bits) → $k+c$	$k+c-1$ ← (c bits) → k	$k-1$ ← (k bits) → 0

Càlcul del número d'etiqueta

A partir del número de bloc b es pot calcular l'etiqueta e fent l'operació següent: $e = b \text{ div } 2^c$, en què *div* és la divisió entera.

El número de conjunt j es pot calcular fent l'operació següent: $j = b \bmod 2^c$, en què *mod* és el residu de la divisió entera.

Tenim 2^{n-k} blocs a la memòria principal i 2^c conjunts a la memòria cau ($2^{n-k} > 2^c$), per tant a cada conjunt de la memòria cau hi podem assignar 2^{n-k-c} ($=2^{n-k}/2^c$) blocs diferents. Com que cada conjunt disposa de ω línies, només ω blocs dels 2^{n-k-c} poden trobar-se en un conjunt de la memòria cau en cada moment.

El número de conjunt de l'adreça indicarà en quin dels 2^c conjunts de la memòria cau es pot trobar el bloc al qual volem accedir de la memòria principal. L'etiqueta ens permetrà saber, comparant simultàniament totes les etiquetes

de les línies que formen el conjunt, si el bloc al qual volem accedir de la memòria principal és un dels blocs que en aquest moment està emmagatzemat en una línia d'aquell conjunt de la memòria cau.

Quan es porta un bloc de memòria principal a la memòria cau, el camp etiqueta del bloc s'emmagatzema en el camp etiqueta de la línia seleccionada dins del conjunt que li correspongui (segons el número de conjunt que especifica l'adreça), d'aquesta manera podrem saber quin bloc està emmagatzemat en cadascuna de les línies de la cau.

De manera general es pot dir que, si tenim una memòria cau de 2^m línies, els blocs de memòria principal que es poden trobar en cada una de les línies de la memòria cau són els següents:

Línies	Número de conjunt	Blocs assignats
0, ..., ω – 1	0	0, 2^c, 2 × (2^c), 3 × (2^c)...
ω, ..., 2ω – 1	1	1, 2^c + 1, 2 × (2^c) + 1, 3 × (2^c) + 1...
	...	...
(2^c – 1) ω, ..., 2^c ω – 1	2^c – 1	2^c – 1, 2^c +(2^c – 1), 2 × 2^c+(2^c – 1), ..., 3 × 2^c+ (2^c – 1)

Per a determinar si un accés a una adreça de memòria produeix un encert en la memòria cau cal fer el següent: a partir de l'adreça de memòria es determina quin és el seu número de conjunt (bits $k + c - 1$ .. k). Aquest número de conjunt s'utilitza com a índex per a accedir a les etiquetes de les línies que identifiquen els blocs que estan emmagatzemats en aquest conjunt i que es comparen simultàniament amb el camp etiqueta de l'adreça (bits $n - 1$ .. $k + c$). Si es produeix una coincidència significa que s'ha produït un encert, llavors s'utilitza el número de paraula (bits $k - 1$ .. 0) per a obtenir la paraula sol·licitada i servir-la al processador.

Si l'etiqueta de l'adreça no coincideix amb cap etiqueta del conjunt, es tracta d'una fallada i caldrà portar tot el bloc de memòria principal a una de les línies d'aquell conjunt de la memòria cau reemplaçant un dels blocs que hi tenim actualment emmagatzemats. Com que un bloc de memòria principal pot anar a qualsevol línia del conjunt, cal decidir quina línia reemplaçarem amb el nou bloc de dades; per a prendre aquesta decisió es poden utilitzar diferents algorismes de reemplaçament que explicarem més endavant.

Exemple

Si tenim una memòria principal de 2^{16} (64 K) paraules i una memòria cau de 2^{10} (1.024) paraules organitzada en 2^6 (64) línies de 2^4 (16) paraules per línia, dividim les línies de la cau en 2^4 (16) conjunts de 2^2 (4) línies; per tant, tindrem ω = 4 i direm que és una memòria cau 4-associativa.

La memòria principal es dividirà en blocs de 2^4 (16) paraules. Una adreça de memòria tindrà 16 bits, els 4 bits menys significatius per al número de paraula i els 16 – 4 = 12 bits restants per al número de bloc, en total tindrem 2^{12} (4.096) blocs de 2^4 (16) paraules. Les adreces es dividiran de la manera següent:

Adreça de memòria	
Número de bloc	Número de paraula
15 4	3 0

El número de bloc de 12 bits es divideix en etiqueta i número de conjunt: 4 bits per al número de conjunt, ja que hi ha 2^4 conjunts, i 12 – 4 = 8 bits per a l'etiqueta. Així les adreces de memòria principal es dividiran de la manera següent:

Adreça de memòria		
Número de bloc		Número de paraula
Etiqueta	Número de conjunt	
15 8	7 4	3 0

L'assignació de blocs de la memòria principal a la memòria cau seria de la manera següent:

Línies	Número de conjunt	Blocs assignats
0, 1, 2, 3	0	0, 16, 32, 48, 64, ..., 4080
4, 5, 6, 7	1	1, 17, 33, 49, 65, ..., 4081
...	...	...
60, 61, 62, 63	15	15, 31, 47, 63, 79, ..., 4095

El mapa d'adreces de la memòria principal quedaria de la manera següent:

	Adreça de memòria		
	Número de bloc		Número de paraula
	Etiqueta	Número de conjunt	
Bloc 0	0	0	0
			...
			15 ($2^4 - 1$)
Bloc 1	0	1	0
			...
			15 ($2^4 - 1$)
...			
Bloc 15	0	15	0
			...
			15 ($2^4 - 1$)
Bloc 16	1	0	0
			...
			15 ($2^4 - 1$)
...			
...			
...			
Bloc 4079	254	15	0
			...
			15 ($2^4 - 1$)
Bloc 4080	255	0	0
			...
			15 ($2^4 - 1$)
...			
Bloc 4095	255	15	0
			...
			15 ($2^4 - 1$)

Es pot observar que tots els blocs que es poden trobar en un mateix conjunt de la cau tenen un valor d'etiqueta diferent; es pot utilitzar el valor de l'etiqueta per a saber quin bloc en concret es troba en cada línia de la memòria cau. La taula anterior mostra que els blocs ombrejats estan assignats tots al conjunt 0 de la memòria cau i podrem saber quin es troba a la memòria cau pel número de l'etiqueta.

A continuació es mostra un possible contingut de la memòria cau:

	Número de bloc	
	Etiqueta	Conjunt
Bloc 16	1	0
Bloc 0	0	0
Bloc 4080	255	0
Bloc 1008	63	0
Bloc 1	0	1
	...	...
	...	...
	...	...
Bloc 4095	255	15

	Memòria cau				
			Paraules de la línia		
Conjunt	Línia	Etiqueta	0	...	$2^k - 1$
0	0	1	M(256)	...	M(271)
	1	0	M(0)	...	M(15)
	2	255	M(65280)	...	M(65295)
	3	63	M(16128)	...	M(16143)
1	4	0	M(16)	...	M(31)
	...	...	...	...	...
...	...	...	...	...	...
15	...	...	...	...	...
	63	255	M(65520)	...	M(65535)

M(x) indica que en aquesta paraula de la línia de la cau hi ha emmagatzemada la paraula de memòria amb l'adreça x.

A continuació es mostra la descomposició d'una de les adreces del bloc 1008 que es troba a la memòria cau.

	Adreça de memòria		
	16128 = 00111111 0000 0000		
	Número de bloc		Número de paraula
	Etiqueta	Número de conjunt	
Bloc 1008	63 = 00111111	0000	0000

3.5. Algorismes de reemplaçament

Quan es produeix una fallada i s'ha de portar a la memòria cau un bloc de memòria principal determinat, si aquest bloc de memòria es pot emmagatzemar en més d'una línia de la memòria cau, s'ha de decidir a quina línia de totes les possibles es posa, i sobreescriure les dades que es troben en aquella línia. L'algorisme de reemplaçament s'encarrega d'aquesta tasca.

En una memòria cau directa no és necessari cap algorisme de reemplaçament, ja que un bloc només pot ocupar una única línia dins de la memòria cau. En una memòria cau completament associativa només s'aplica l'algorisme de reemplaçament per a seleccionar una de les línies de la memòria cau. En una memòria cau associativa per conjunts només s'aplica l'algorisme de reemplaçament per a seleccionar una línia dins d'un conjunt concret.

Perquè aquests algorismes de reemplaçament no penalitzin el temps mitjà d'accés a memòria s'han d'implementar en maquinari i, per tant, no haurien de ser gaire complexos.

A continuació es descriuen de manera general els algorismes de reemplaçament més comuns, però es poden trobar altres algorismes o variants d'aquests.

1) FIFO (*first in first out*). Per a escollir la línia s'utilitza una cua, de manera que la línia que fa més temps que està emmagatzemada en la memòria cau serà la reemplaçada. Aquest algorisme pot reduir el rendiment de la memòria cau perquè la línia que fa més temps que es troba emmagatzemada a la memòria cau no ha de ser necessàriament la que s'utilitzi menys.

Es pot implementar fàcilment utilitzant tècniques de *buffers* circulars (o *round-robin*): cada cop que s'ha de substituir una línia s'utilitza la línia del *buffer* següent, i quan s'arriba a l'última es torna a començar pel principi.

2) LFU (*least frequently used*). En aquest algorisme s'escull la línia que hem utilitzat menys vegades.

Es pot implementar afegint un comptador del nombre d'accessos a cada línia de la memòria cau.

3) LRU (*least recently used*). Aquest algorisme escull la línia que fa més temps que no s'utilitza. És l'algorisme més eficient, però el més difícil d'implementar, especialment si s'ha d'escollir entre moltes línies. S'utilitza habitualment en memòries cau associatives per conjunts, amb conjunts petits de 2 o 4 línies.

Per a memòries cau 2-associatives, es pot implementar afegint un bit en cada línia; quan es fa referència a una de les dues línies, aquest bit es posa a 1 i l'altre es posa a 0, per a indicar quina de les dues línies ha estat la darrera que s'ha utilitzat.

4) Aleatori. Els algorismes anteriors es basen en factors sobre la utilització de les línies de la cau, en canvi aquest algorisme escull la línia que s'ha de reemplaçar a l'atzar. Aquest algorisme és molt simple i s'ha demostrat que té un rendiment només lleugerament inferior als algorismes que tenen en compte factors d'utilització de les línies.

3.6. Comparativa entre diferents sistemes de memòria cau

Utilitzarem un exemple per a veure com els accessos a memòria d'un programa poden produir encerts i fallades en la memòria cau i com modifiquen el contingut de la memòria cau.

Suposem una memòria principal de 2^{10} (1.024) paraules, en què cada adreça de memòria correspon a una paraula, i una memòria cau de 2^4 (4) línies de 2^2 (4) paraules; per tant la memòria principal també estarà organitzada en blocs de mida 4 paraules.

Per a determinar el número de bloc que correspon a una adreça de memòria, dividim l'adreça *d* entre el nombre de paraules d'un bloc:

$$b = d \text{ div } 2^k = d \text{ div } 4$$

Aquest número de bloc s'aplica a totes les organitzacions de la memòria cau.

3.6.1. Memòria cau d'assignació directa

En una memòria cau directa un bloc de memòria només es pot trobar en una única línia de la memòria cau. A un bloc de memòria *b* li correspondrà l'etiqueta *e* i l'assignarem a la línia *l* de la memòria cau, per determinar l'etiqueta i la línia, dividim el número de bloc *b* entre el nombre de línies de la memòria cau:

$$e = b \text{ div } 2^m = b \text{ div } 4$$
$$l = b \text{ mod } 2^m = b \text{ mod } 4$$

Per tant, els blocs de memòria principal que es poden assignar a cada línia de la memòria cau són els següents:

l: número de línia	Blocs	Bloc: etiqueta (6 bits) línia (2 bits)
0 (00)	0, 4, 8, 12, ..., 252	0(000000) 0(00), 1(000001) 0(00), 2(000010) 0(00), ..., 63(111111) 0(00)
1 (01)	1, 5, 9, 13, ..., 253	0(000000) 1(01), 1(000001) 1(01), 2(000010) 1(01), ..., 63(111111) 1(01)
2 (10)	2, 6, 10, 14, ..., 254	0(000000) 2(10), 1(000001) 2(10), 2(000010) 2(10), ..., 63(111111) 2(10)
3 (11)	3, 7, 11, 15, ..., 255	0(000000) 3(11), 1(000001) 3(11), 2(000010) 3(11), ..., 63(111111) 3(11)

Mostrem a quines línies de la memòria cau s'assignen els primers 16 blocs de memòria principal amb les adreces de memòria que conté el bloc:

l: número de línia	b:e (a_0 ,a_1 ,a_2 ,a_3): bloc assignat : etiqueta (adreces del bloc)			
0	0:0 (0,1,2,3)	4:1 (16,17,18,19)	8:2 (32,33,34,35)	12:3 (48,49,50,51)
1	1:0 (4,5,6,7)	5:1 (20,21,22,23)	9:2 (36,37,38,39)	13:3 (52,53,54,55)
2	2:0 (8,9,10,11)	6:1 (24,25,26,27)	10:2 (40,41,42,43)	14:3 (56,57,58,59)
3	3:0 (12,13,14,15)	7:1 (28,29,30,31)	11:2 (44,45,46,47)	15:3 (60,61,62,63)

Cal remarcar que especifiquem el número de bloc i el número d'etiqueta, però que el valor que tindrem realment emmagatzemat a la cau és només l'etiqueta associada al bloc.

Suposem que durant l'execució d'un programa s'accedeix a les adreces de memòria següents: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La taula següent mostra l'evolució del contingut de la memòria cau, i indica el número de bloc, l'etiqueta del bloc i les adreces de memòria del bloc que hi ha a cada una de les 4 línies de la memòria cau. Inicialment la memòria cau és buida. Quan es produeix un encert, s'indica amb una *E* la línia on s'ha produït l'encert. Cada cop que hi ha una fallada, s'indica amb una lletra *F* quina línia de la memòria cau es reemplaçarà i s'actualitza el contingut portant el nou bloc de memòria principal a aquesta línia de la memòria cau.

	Estat inicial	1	Fallada	2	4	Fallada	10	Fallada	15	Fallada
Línia 0		F	0:0 (0, 1, 2, 3)	E		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)
Línia 1					F	1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)
Línia 2							F	2:0 (8, 9, 10, 11)		2:0 (8, 9, 10, 11)
Línia 3									F	3:0 (12, 13, 14, 15)

	1	26	Fallada	27	28	Fallada	29	36	Fallada
Línia 0	E		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)
Línia 1			1:0 (4, 5, 6, 7)			1:0 (4, 5, 6, 7)		F	9:2 (36, 37, 38, 39)
Línia 2		F	6:1 (24, 25, 26, 27)	E		6:1 (24, 25, 26, 27)			6:1 (24, 25, 26, 27)
Línia 3			3:0 (12, 13, 14, 15)		F	7:1 (28, 29, 30, 31)	E		7:1 (28, 29, 30, 31)

	37	38	40	Fallada	10	Fallada	11	12	Fallada	13	9
Línia 0				0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)		
Línia 1	E	E		9:2 (36, 37, 38, 39)		9:2 (36, 37, 38, 39)			9:2 (36, 37, 38, 39)		
Línia 2			F	10:2 (40, 41, 42, 43)	F	2:0 (8, 9, 10, 11)	E		2:0 (8, 9, 10, 11)		E
Línia 3				7:1 (28, 29, 30, 31)		7:1 (28, 29, 30, 31)		F	3:0 (12, 13, 14, 15)	E	

	30	Fallada	8	12	Fallada	40	Fallada	17	Fallada	40
Línia 0		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)	F	4:1 (16, 17, 18, 19)	
Línia 1		9:2 (36, 37, 38, 39)			9:2 (36, 37, 38, 39)		9:2 (36, 37, 38, 39)		9:2 (36, 37, 38, 39)	
Línia 2		2:0 (8, 9, 10, 11)	E		2:0 (8, 9, 10, 11)	F	10:2 (40, 41, 42, 43)		10:2 (40, 41, 42, 43)	E

Línia 3	F	7:1 (28,29,30,31)		F	3:0 (12, 13, 14, 15)		3:0 (12, 13, 14, 15)		3:0 (12, 13, 14, 15)	

3.6.2. Memòria cau completament associativa

Utilitzem ara una memòria cau completament associativa. En una memòria cau completament associativa, un bloc de memòria principal es pot trobar en qualsevol línia de la cau. Les adreces de memòria es divideixen en número de bloc i número de paraula.

Mostrem els primers 16 blocs de la memòria principal amb les adreces de memòria que conté el bloc:

b (a_0 ,a_1 ,a_2 ,a_3): bloc (adreces del bloc)			
0 (0,1,2,3)	4 (16,17,18,19)	8 (32,33,34,35)	12 (48,49,50,51)
1 (4,5,6,7)	5 (20,21,22,23)	9 (36,37,38,39)	13 (52,53,54,55)
2 (8,9,10,11)	6 (24,25,26,27)	10 (40,41,42,43)	14 (56,57,58,59)
3 (12,13,14,15)	7 (28,29,30,31)	11 (44,45,46,47)	15 (60,61,62,63)

Cal remarcar que el número de bloc és el número d'etiqueta que tindrem emmagatzemat a la memòria cau.

1) FIFO. Utilitzem aquest algorisme de reemplaçament i la mateixa seqüència d'adreces de memòria que en el cas anterior: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La taula següent mostra l'evolució del contingut de la memòria cau, i indica el número de bloc i les adreces de memòria del bloc que hi ha a cada una de les 4 línies de la memòria cau. Inicialment la memòria cau és buida. Quan es produeix un encert, s'indica amb una *E* la línia on s'ha produït l'encert. Cada cop que hi ha una fallada, s'indica amb una lletra *F* quina línia de la memòria cau es reemplaçarà i s'actualitza el contingut portant el nou bloc de memòria principal a aquesta línia de la memòria cau.

	Estat inicial	1	Fallada	2	4	Fallada	10	Fallada	15	Fallada
Línia 0		F	0 (0, 1, 2, 3)	E		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)
Línia 1					F	1 (4, 5, 6, 7)		1 (4, 5, 6, 7)		1 (4, 5, 6, 7)
Línia 2							F	2 (8, 9, 10, 11)		2 (8, 9, 10, 11)
Línia 3									F	3 (12, 13, 14, 15)

	26	Fallada	27	28	Fallada	29	36	Fallada	37	38
Línia 0	F	6 (24, 25, 26, 27)	E		6 (24, 25, 26, 27)			6 (24, 25, 26, 27)		

Línia 1		1 (4, 5, 6, 7)		F	7 (28, 29, 30, 31)	E		7 (28, 29, 30, 31)			
Línia 2		2 (8, 9, 10, 11)			2 (8, 9, 10, 11)		F	9 (36, 37, 38, 39)	E	E	
Línia 3		3 (12, 13, 14, 15)			3 (12, 13, 14, 15)			3 (12, 13, 14, 15)			

	40	Fallada	10	Fallada	11	12	Fallada	13	9
Línia 0		6 (24, 25, 26, 27)	F	2 (8, 9, 10, 11)	E		2 (8, 9, 10, 11)		E
Línia 1		7 (28, 29, 30, 31)		7 (28, 29, 30, 31)		F	3 (12, 13, 14, 15)	E	
Línia 2		9 (36, 37, 38, 39)		9 (36, 37, 38, 39)			9 (36, 37, 38, 39)		
Línia 3	F	10 (40, 41, 42, 43)		10 (40, 41, 42, 43)			10 (40, 41, 42, 43)		

	30	Fallada	8	12	40	17	Fallada	40	Fallada
Línia 0		2 (8, 9, 10, 11)	E				2 (8, 9, 10, 11)	F	10 (40, 41, 42, 43)
Línia 1		3 (12, 13, 14, 15)		E			3 (12, 13, 14, 15)		3 (12, 13, 14, 15)
Línia 2	F	7 (28, 29, 30, 31)					7 (28, 29, 30, 31)		7 (28, 29, 30, 31)
Línia 3		10 (40, 41, 42, 43)			E	F	4 (16, 17, 18, 19)		4 (16, 17, 18, 19)

2) LRU. Utilitzem ara l'algorisme de reemplaçament LRU i la mateixa seqüència que en els casos anteriors: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La taula següent mostra l'evolució del contingut de la memòria cau, i indica el número de bloc i les adreces de memòria del bloc que hi ha a cada una de les 4 línies de la memòria cau. Inicialment la memòria cau és buida. Quan es produeix un encert, s'indica amb una *E* la línia on s'ha produït l'encert. Cada cop que hi ha una fallada, s'indica amb una lletra *F* quina línia de la memòria cau es reemplaçarà i s'actualitza el contingut portant el nou bloc de memòria principal a aquesta línia de la memòria cau.

	Estat inicial	1	Fallada	2	4	Fallada	10	Fallada	15	Fallada
Línia 0		F	0 (0, 1, 2, 3)	E		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)		0 (0, 1, 2, 3)
Línia 1					F	1 (4, 5, 6, 7)		1 (4, 5, 6, 7)		1 (4, 5, 6, 7)
Línia 2							F	2 (8, 9, 10, 11)		2 (8, 9, 10, 11)
Línia 3									F	3 (12, 13, 14, 15)

	1	26	Fallada	27	28	Fallada	29	36	Fallada	37	38
Línia 0	E		0 (0, 1, 2, 3)			0 (0, 1, 2, 3)			0 (0, 1, 2, 3)		
Línia 1		F	6 (24, 25, 26, 27)	E		6 (24, 25, 26, 27)			6 (24, 25, 26, 27)		
Línia 2			2 (8, 9, 10, 11)		F	7 (28, 29, 30, 31)	E		7 (28, 29, 30, 31)		

Línia 3			3 (12, 13, 14, 15)			3 (12, 13, 14, 15)		F	9 (36, 37, 38, 39)	E	E

	40	Fallada	10	Fallada	11	12	Fallada	13	9
Línia 0	F	10 (40, 41, 42, 43)		10 (40, 41, 42, 43)			10 (40, 41, 42, 43)		
Línia 1		6 (24, 25, 26, 27)	F	2 (8, 9, 10, 11)	E		2 (8, 9, 10, 11)		E
Línia 2		7 (28, 29, 30, 31)		7 (28, 29, 30, 31)		F	3 (12, 13, 14, 15)	E	
Línia 3		9 (36, 37, 38, 39)		9 (36, 37, 38, 39)			9 (36, 37, 38, 39)		

	30	Fallada	8	12	40	17	Fallada	40
Línia 0		10 (40, 41, 42, 43)			E		10 (40, 41, 42, 43)	E
Línia 1		2 (8, 9, 10, 11)	E				2 (8, 9, 10, 11)	
Línia 2		3 (12, 13, 14, 15)		E			3 (12, 13, 14, 15)	
Línia 3	F	7 (28, 29, 30, 31)				F	4 (16, 17, 18, 19)	

3.6.3. Memòria cau associativa per conjunts

En una memòria cau associativa per conjunts amb dos conjunts de dues línies, un bloc de memòria es pot trobar en un únic conjunt i dins del conjunt en qualsevol línia. A un bloc de memòria *b* li correspondrà l'etiqueta *e* i l'assignarem al conjunt *j* de la memòria cau, per determinar l'etiqueta i el conjunt, dividim el número de bloc *b* entre el nombre de línies de cada conjunt:

$$e = b \text{ div } 2^c = b \text{ div } 2$$
$$j = b \bmod 2^c = b \bmod 2$$

Per tant, els blocs de memòria principal que es poden assignar a cada conjunt de la memòria cau són els següents:

***j*: número de conjunt**	**Línia**	**Blocs**	**Bloc: etiqueta (7 bits) conjunt de 1 bit**
0	0	0, 2, 4, 6, 8, ..., 254	0:0(0000000) 0, 2:1(0000001) 0, 4:2(0000010) 0, ..., 254:127(1111111) 0
	1		
1	2	1, 3, 5, 7, 9, ..., 255	1:0(0000000) 1, 3:1(0000001) 1, 5:2(0000010) 1, ..., 255:127(1111111) 1
	3		

Mostrem a quins conjunts de la memòria cau s'assignen els primers 16 blocs de memòria principal amb les adreces de memòria que conté el bloc:

j: número de conjunt	b:e (a_0 ,a_1 ,a_2 ,a_3): bloc assignat : etiqueta (adreces del bloc)			
0	0:0 (0,1,2,3) 2:1 (8,9,10,11)	4:2 (16,17,18,19) 6:3 (24,25,26,27)	8:4 (32,33,34,35) 10:5 (40,41,42,43)	12:6 (48,49,50,51) 14:7 (56,57,58,59)
1	1:0 (4,5,6,7) 3:1 (12,13,14,15)	5:2 (20,21,22,23) 7:3 (28,29,30,31)	9:4 (36,37,38,39) 11:5 (44,45,46,47)	13:6 (52,53,54,55) 15:7 (60,61,62,63)

Cal remarcar que especifiquem el número de bloc i el número d'etiqueta, però que el valor que tindrem realment emmagatzemat a la cau és només l'etiqueta associada al bloc.

1) LRU. Utilitzem l'algorisme de reemplaçament LRU i la mateixa seqüència que en els casos anteriors: 1, 2, 4, 10, 15, 1, 26, 27, 28, 29, 36, 37, 38, 40, 10, 11, 12, 13, 9, 30, 8, 12, 40, 17, 40.

La taula següent mostra l'evolució del contingut de la memòria cau, i indica el número de bloc, l'etiqueta del bloc i les adreces de memòria del bloc que hi ha a cada una de les 4 línies de la memòria cau. Inicialment la memòria cau és buida. Quan es produeix un encert, s'indica amb una *E* la línia on s'ha produït l'encert. Cada cop que hi ha una fallada, s'indica amb una lletra *F* quina línia de la memòria cau es reemplaçarà i s'actualitza el contingut portant el nou bloc de memòria principal a aquesta línia de la memòria cau.

	Estat inicial	1	Fallada	2	4	Fallada	10	Fallada	15	Fallada
Línia 0		F	0:0 (0, 1, 2, 3)	E		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)		0:0 (0, 1, 2, 3)
Línia 1							F	2:1 (8, 9, 10, 11)		2:1 (8, 9, 10, 11)
Línia 2					F	1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)		1:0 (4, 5, 6, 7)
Línia 3									F	3:1 (12, 13, 14, 15)

	1	26	Fallada	27	28	Fallada	29	36	Fallada
Línia 0	E		0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)			0:0 (0, 1, 2, 3)
Línia 1		F	6:3 (24,25,26,27)	E		6:3 (24,25,26,27)			6:3 (24,25,26,27)
Línia 2			1:0 (4, 5, 6, 7)		F	7:3 (28,29,30,31)	E		7:3 (28,29,30,31)
Línia 3			3:1 (12, 13, 14, 15)			3:1 (12, 13, 14, 15)		F	9:4 (36,37,38,39)

	37	38	40	Fallada	10	Fallada	11	12	Fallada	13	9
Línia 0			F	10:5 (40,41,42,43)		10:5 (40,41,42,43)			10:5 (40,41,42,43)		
Línia 1				6:3 (24,25,26,27)	F	2:1 (8,9,10,11)	E		2:1 (8,9,10,11)		E
Línia 2				7:3 (28,29,30,31)		7:3 (28,29,30,31)		F	3:1 (12,13,14,15)	E	
Línia 3	E	E		9:4 (36,37,38,39)		9:4 (36,37,38,39)			9:4 (36,37,38,39)		

	30	Fallada	8	12	40	17	Fallada	40
Línia 0		10:5 (40,41,42,43)			E		10:5 (40,41,42,43)	E
Línia 1		2:1 (8,9,10,11)	E			F	4:2 (16,17,18,19)	
Línia 2		3:1 (12,13,14,15)		E			3:1 (12,13,14,15)	
Línia 3	F	7:3 (28,29,30,31)					7:3 (28,29,30,31)	

Podem comparar les taxes de fallades dels casos anteriors i observar que, en aquesta seqüència d'accessos, en l'algorisme LRU és on s'obtenen menys fallades i la memòria cau d'assignació directa és la que obté més fallades.

La taxa de fallades en cada cas és:

Assignació directa:	$T_f = 14/25 = 0,56$
Completament associativa amb FIFO:	$T_f = 13/25 = 0,52$
Completament associativa amb LRU:	$T_f = 12/25 = 0,48$
Associativa per conjunts amb LRU:	$T_f = 12/25 = 0,48$

3.7. Polítiques d'escriptura

Quan accedim a la memòria cau podem fer lectures o escriptures; fins ara hem vist la problemàtica d'accedir a la memòria cau per a llegir una dada. Quan s'ha de fer una operació d'escriptura apareix una nova problemàtica perquè les dades que tenim a la memòria cau són una còpia de les dades que tenim a la memòria principal i s'ha de garantir la coherència de les dades.

Analitzarem el cas amb un únic processador i un únic nivell de memòria cau entre el processador i la memòria principal. Si tenim més d'un processador amb una memòria cau local a cada processador, la modificació d'una dada en una d'aquestes memòries cau invalida el valor de la dada en la memòria principal, però també invalida el valor de la dada si es troba en una altra memòria cau, de manera semblant, si tenim altres dispositius que puguin modificar directament una dada de la memòria principal, el valor d'aquesta dada queda invalidat en les memòries cau on es pugui trobar. La gestió de la coherència en aquests sistemes és més complexa i no s'analitzarà.

A continuació comentarem diferents polítiques per a gestionar les escriptures i mantenir la coherència entre les dades de la memòria cau i la memòria principal:

1) Escriptura immediata (*write trough*): quan s'escriu en la memòria cau, també s'escriu en la memòria principal transferint tot el bloc que conté la dada modificada; d'aquesta manera en tot moment la còpia que tenim en la cau

és idèntica a la que tenim en la memòria principal. La política d'escriptura immediata és la més fàcil d'implementar, però té l'inconvenient que produeix un gran flux d'informació entre la memòria cau i la memòria principal.

2) Escriptura ajornada (*write back*): les escriptures s'efectuen només sobre la memòria cau. La memòria principal s'actualitza quan s'elimina una línia de la memòria cau que ha estat modificada. Això implica afegir alguns bits a cada línia de la memòria cau per a saber si la línia s'ha modificat o no.

Si s'ha de reemplaçar una línia que ha estat modificada primer cal copiar la línia modificada a la memòria principal i a continuació portar el nou bloc, cosa que augmenta significativament el temps per a accedir a la dada.

Fallada en l'escriptura

Quan es vol fer una escriptura d'una adreça que no és a la memòria cau, es produirà una fallada; aquesta fallada es pot tractar de diferents maneres:

a) Escriure directament a memòria principal i no portar la dada a la memòria cau. Aquesta tècnica es pot utilitzar en l'escriptura immediata. Evitem transferir el bloc a la cau però això no té en compte la proximitat referencial, ja que és molt probable que hi hagi nous accessos al mateix bloc de dades.

b) Portar el bloc a la memòria cau i escriure simultàniament a la cau i a la memòria principal. Aquesta tècnica és la que s'utilitza habitualment en l'escriptura immediata.

c) Portar el bloc a la memòria cau i escriure només a la cau. Aquesta tècnica s'utilitza habitualment en escriptura ajornada.

En màquines reals cada cop més s'utilitzen polítiques d'escriptura ajornada, però el tractament de les fallades en cas d'escriptura és diferent, principalment perquè es consideren diferents nivells de memòria cau i perquè múltiples dispositius (processadors, DMA, canals d'E/S) poden accedir a la memòria.

4. Memòria interna

Tots els tipus de memòria interna s'implementen utilitzant tecnologia de semiconductors, i tenen el transistor com a element bàsic de la seva construcció.

L'element bàsic en tota memòria és la cel·la. Una cel·la permet emmagatzemar un bit, un valor 0 o 1 definit per una diferència de potencial elèctric. La manera de construir una cel·la de memòria varia segons la tecnologia utilitzada.

La memòria interna és una memòria d'accés aleatori; es pot accedir a qualsevol paraula de memòria especificant una adreça de memòria.

Una manera de classificar la memòria interna segons la perdurabilitat és la següent:

- Memòria volátil
  - SRAM (*static random access memory*)
  - RAM (*dynamic random access memory*)

- Memòria no volàtil
  - ROM (*read only memory*)
  - PROM (*programmable read only memory*)
  - EPROM (*erasable programmable read only memory*)
  - EEPROM (*electrically erasable programmable read only memory*)
  - Memòria flaix

4.1. Memòria volàtil

La memòria volàtil és la memòria que necessita un corrent elèctric per a mantenir el seu estat, de manera genèrica anomenada *RAM*.

Les memòries volàtils poden ser de dos tipus:

1) SRAM. La memòria estàtica d'accés aleatori (SRAM) implementa cada cel·la de memòria utilitzant un flip-flop bàsic per a emmagatzemar un bit d'informació, i manté la informació mentre el circuit de memòria rep alimentació elèctrica.

Per a implementar cada cel·la de memòria són necessaris diversos transistors, típicament sis transistors; això fa que tingui una capacitat d'integració limitada, i el cost és també elevat en relació amb altres tipus de memòria RAM com la DRAM; en canvi és el tipus de memòria RAM més ràpid.

Usos de la memòria SRAM

La memòria SRAM s'utilitza en la construcció dels registres del processador i en la memòria cau.

2) DRAM. La memòria dinàmica d'accés aleatori implementa cada cel·la de memòria utilitzant la càrrega d'un condensador. A diferència dels flip-flops, els condensadors amb el temps perden la càrrega emmagatzemada i fan necessari un circuit de refresc per a mantenir la càrrega i mantenir, per tant, el valor de cada bit emmagatzemat. Això fa que tingui un temps d'accés més gran que la SRAM.

Cada cel·la de memòria està formada només per un transistor i un condensador; per tant, les cel·les de memòria són molt més petites que les cel·les de memòria de la SRAM, i això permet una gran escala d'integració i, alhora, fer memòries més grans en menys espai.

Usos de la memòria DRAM

La memòria DRAM s'utilitza en la construcció de la memòria principal del computador.

4.2. Memòria no volàtil

La memòria no volàtil manté l'estat sense necessitat de corrent elèctric.

Les memòries no volàtils poden ser de diferents tipus:

1) Memòria de només lectura o ROM (*read only memory*). Tal com indica el seu nom, es tracta de memòries de només lectura que no permeten operacions d'escriptura i, per tant, la informació que contenen no es pot esborrar o modificar.

Aquest tipus de memòries es poden utilitzar per a emmagatzemar els microprogrames en una unitat de control microprogramada; també es poden utilitzar en dispositius que necessiten treballar sempre amb la mateixa informació.

La gravació de la informació en aquests tipus de memòries forma part del procés de fabricació del xip de memòria. Aquests processos impliquen la fabricació d'un gran volum de memòries ROM amb la mateixa informació; és un procés costós i no és rendible per a un nombre reduït d'unitats.

2) Memòria programable de només lectura o PROM (*programmable read only memory*). Quan cal fabricar un nombre reduït de memòries ROM amb la mateixa informació gravada es recorre a un altre tipus de memòries ROM: les memòries ROM programables (PROM).

A diferència de les anteriors, la gravació no forma part del procés de fabricació dels xips de memòria, sinó que s'efectua posteriorment amb un procés elèctric utilitzant un maquinari especialitzat per a la gravació de memòries d'aquest tipus.

Com que el procés de programació no forma part del procés de fabricació, l'usuari final d'aquest tipus de memòries pot gravar-ne el contingut segons les seves necessitats.

Cal destacar que, tal com passa amb les memòries ROM, el procés de gravació o programació només es pot fer un cop.

Aquest tipus de memòria té unes aplicacions semblants a les de les memòries ROM.

3) Memòria reprogramable majoritàriament de lectura. Aquestes poden ser de tres tipus:

a) EPROM (*erasable programmable read only memory*). Es tracta de memòries en les quals habitualment es fan operacions de lectura, però el contingut de les quals pot ser esborrat i gravat de nou.

Cal destacar que el procés d'esborrar és un procés que esborra completament tot el contingut de la memòria; no se'n pot esborrar només una part. Per a esborrar s'aplica llum ultraviolada sobre el xip de memòria EPROM; per a permetre aquest procés, el xip disposa d'una petita finestra sobre la qual s'aplica la llum ultraviolada.

La gravació de la memòria es fa mitjançant un procés elèctric utilitzant un maquinari específic.

Tant per al procés d'esborrar com per al procés de gravar, cal treure el xip de memòria de la seva localització d'ús habitual, ja que la realització d'aquestes dues tasques implica la utilització de maquinari específic.

b) EEPROM (*electrically erasable programmable read only memory*). Té un funcionament semblant a l'EPROM, permet esborrar el contingut i gravar informació nova, però, a diferència de les memòries EPROM, totes les operacions són realitzades elèctricament.

Per a gravar dades no cal esborrar-les prèviament; es permet modificar directament només un byte o diversos sense modificar la resta d'informació.

Són memòries majoritàriament de lectura, ja que el procés d'escriptura és considerablement més lent que el procés de lectura.

c) Memòria flaix. La memòria flaix és un tipus de memòria semblant a les memòries EEPROM, en què l'esborrament és elèctric, amb l'avantatge que el procés d'esborrar i gravar és molt ràpid. La velocitat de lectura és superior a la velocitat d'escriptura, però totes dues són del mateix ordre de magnitud.

Aquest tipus de memòria no permet esborrar la informació byte a byte, sinó que cal esborrar blocs de dades sencers; això fa que tot el contingut de la memòria es pugui esborrar en pocs segons, però també alenteix el procés d'escriptura, ja que per a escriure una dada nova cal esborrar prèviament tot el bloc.

Té una capacitat d'integració molt elevada i per aquest motiu també s'utilitza per a dispositius d'emmagatzematge extern.

5. Memòria externa

La memòria externa està formada per dispositius d'emmagatzematge secundari (discos magnètics, CD, DVD, Blu-ray, etc.). Aquests dispositius es poden trobar físicament dins del computador o fora del computador.

La memòria externa és de tipus no volàtil; per tant, les dades que es vulguin mantenir durant un temps indefinit o de manera permanent es poden emmagatzemar en dispositius de memòria externa

El mètode d'accés varia segons el dispositiu: generalment els dispositius basats en disc utilitzen un mètode d'accés directe, mentre que altres dispositius com les cintes magnètiques poden utilitzar accés seqüencial.

Les dades emmagatzemades en la memòria externa són visibles al programador en forma de blocs de dades, no com a dades individuals (bytes), normalment en forma de registres o fitxers. L'accés a aquests dispositius es fa mitjançant el sistema d'E/S del computador i és gestionat pel sistema operatiu.

Vegeu també

La manera d'accedir als dispositius d'emmagatzematge secundari es veurà amb més detall quan s'analitzi el sistema d'E/S.

5.1. Discos magnètics

Els discos magnètics són dispositius formats per un conjunt de plats amb superfícies magnètiques i un conjunt de capçals de lectura i escriptura. La informació es grava en aquestes superfícies. Un sol dispositiu integra diversos plats, que habitualment n'utilitzen les dues cares per a emmagatzemar la informació.

Els plats i els capçals són accionats per motors elèctrics. Els plats fan un moviment de rotació continu i els capçals es poden moure de la part més externa del disc a la part més interna, cosa que permet un accés directe a qualsevol posició del disc.

Són els dispositius d'emmagatzematge secundari més importants en qualsevol computador i constitueixen la base de qualsevol sistema de memòria externa.

Són els dispositius de memòria externa que proporcionen més capacitat d'emmagatzematge i els que tenen unes prestacions més elevades. La capacitat dels discos magnètics és de l'ordre de Tbytes, el temps d'accés mitjà és de pocs mil·lisegons i poden arribar a velocitats de transferència de l'ordre del Gbyte per segon.

5.1.1. RAID

Un sistema RAID[1] consisteix en la utilització d'una col·lecció de discos que treballen en paral·lel amb l'objectiu de millorar el rendiment i la fiabilitat del sistema d'emmagatzematge.

(1)RAID són les sigles de *redundant array of independent disks*, en català matriu redundant de discos independents.

Un conjunt d'operacions d'E/S pot ser tractat en paral·lel si les dades a les quals s'ha d'accedir en cada operació són en diferents discos; també una sola operació d'E/S pot ser tractada en paral·lel si el bloc de dades al qual cal accedir es troba distribuït entre diversos discos.

Un RAID està format per un conjunt de discos que són vistos pel sistema operatiu com un sol disc lògic.

Les dades es poden distribuir entre els discos físics segons configuracions diferents. S'utilitza informació redundant per a proporcionar capacitat de recuperació en el cas de fallades en algun disc.

La classificació de tipus de RAID original inclou 7 nivells, del RAID 0 al RAID 6, en què cadascun necessita un nombre diferent de discos i utilitza diferents sistemes de control de la paritat i de detecció i correcció d'errors.

El control d'un sistema RAID es pot fer per programari o per maquinari, amb un controlador específic.

5.2. Cinta magnètica

La cinta magnètica és un dispositiu que utilitza una tecnologia d'emmagatzematge semblant a la dels discos magnètics; la diferència bàsica és que la superfície magnètica sobre la qual es guarda la informació es troba sobre una cinta de polièster. Com que és una cinta, s'utilitza un mètode d'accés seqüencial.

Són dispositius lents i s'utilitzen per a fer còpies de seguretat de grans volums de dades o per a emmagatzemar dades a les quals s'accedeix amb poca freqüència.

5.3. Memòria flaix

Les tendències actuals inclouen dispositius d'emmagatzematge construïts a partir de circuits de memòria flaix. L'objectiu és substituir els discos magnètics oferint característiques semblants pel que fa a temps d'accés, taxa de transferència de dades i capacitat d'emmagatzematge.

Com que les memòries flaix no tenen parts mecàniques ni superfícies magnètiques, són més tolerants a fallades, i més adequades per a ser utilitzades en entorns en què la fiabilitat és molt important. També hi ha dispositius d'aquest tipus per al gran públic, però que actualment no rivalitzen amb els discos magnètics, especialment pel que fa a capacitat.

Usos de la memòria flaix

La memòria flaix s'utilitza habitualment en diferents tipus de dispositius d'emmagatzematge extern: targetes de memòria (Compact Flash, Secure Digital, etc.), memòries USB (*pen-drive*) i unitats d'estat sòlid (SSD).

5.4. Disc òptic

Els discos òptics són unitats d'emmagatzematge que utilitzen llum làser per a fer operacions de lectura i escriptura sobre un suport extraïble. Les unitats poden ser internes (connectades a un bus intern del computador) o externes (connectades per un bus extern).

Hi ha bàsicament tres tipus de suports: CD, DVD i Blu-ray (BD). La seva capacitat màxima varia segons els tipus de suport; és de l'orde dels centenars de Mbytes en el cas del CD, de l'ordre Gbytes en el cas dels DVD i de desenes de Gbytes en el cas dels Blu-ray.

El tipus d'operacions que es poden fer sobre el disc depèn del tipus: hi ha discos de només lectura (CD-ROM, DVD-ROM, BD-ROM), discos que es poden escriure només un cop (CD-R, DVD+R, DVD-R, BD-R) i discos que es poden escriure diversos cops (CD-RW, DVD+RW, DVD-RW, BD-RE).

Habitualment una mateixa unitat és capaç de treballar amb suports de tipus diferents. Per exemple, una gravadora de DVD és capaç de llegir CD-ROM, DVD-ROM i de llegir i escriure CD-R, DVD+R, DVD-R, CD-RW, DVD+RW i DVD-RW.

La velocitat és inferior als discos magnètics. Estan dissenyats bàsicament per a fer operacions de lectura, ja que escriure-hi implica un procés de gravació relativament lent, de l'ordre de minuts, depenent de la quantitat de dades que es vol emmagatzemar.

5.5. Xarxa

Utilitzant els recursos de xarxes LAN o Internet, es pot disposar d'emmagatzematge de gran capacitat. N'hi ha de tipus diferents, com SMB, NFS o SAN. Es pot accedir a qualsevol dada emmagatzemada en un ordinador connectat a la xarxa, sense límit de capacitat, i per a ampliar la capacitat només cal afegir ordinadors nous a la xarxa amb capacitat d'emmagatzematge.

És habitual mesurar la velocitat de transferència en bits per segon, i està limitada per l'amplada de banda de la xarxa.

Té sentit utilitzar aquest tipus de sistema quan volem gestionar grans volums de dades, els suports físics que s'utilitzen per a emmagatzemar les dades són discos magnètics; si l'objectiu és només fer còpies de seguretat s'acostumen a utilitzar cintes magnètiques.

Resum

En aquest mòdul, primer s'ha fet una introducció al sistema de memòria d'un computador i s'han explicat les característiques principals dels diferents tipus de memòries.

- Localització
- Capacitat
- Mètodes d'accés
- Organització de les dades d'una memòria
- Temps d'accés i velocitat
- Cost
- Característiques físiques

S'ha introduït el concepte de *jerarquia de memòries* amb l'objectiu d'aconseguir que el processador, quan accedeix a una dada, aquesta es trobi en el nivell més ràpid i així aconseguir tenir una memòria amb un cost moderat, una velocitat propera al nivell més ràpid i la capacitat del nivell més gran.

La jerarquia de memòries d'un computador s'organitza en nivells diferents:

- Registres

- Memòria interna
  - Memòria cau
  - Memòria principal

- Memòria externa

Hem vist el concepte pel qual la jerarquia de memòries funciona, la proximitat referencial, i n'hem distingit dos tipus:

- Proximitat temporal
- Proximitat espacial

S'han explicat detalladament les característiques i el funcionament de la memòria cau:

- L'organització de la memòria cau

- El concepte d'*encert* i *fallada* i els índexs per a mesurar el rendiment

- Les diferents polítiques d'assignació que hi ha a l'hora de portar un bloc de memòria principal a la memòria cau:

  - assignació directa
  - assignació completament associativa
  - assignació associativa per conjunts

- Els diferents algorismes de reemplaçament que es poden utilitzar quan cal substituir el contingut d'una línia de la memòria cau:
  - FIFO
  - LFU
  - LRU
  - Aleatori

- Comparativa entre els diferents sistemes de memòria cau

- Com gestionar les escriptures en una memòria cau

A continuació s'han explicat els diferents tipus de memòria interna i com es poden classificar segons la seva perdurabilitat en memòries volàtils i no volàtils, i els diferents tipus que podem trobar dins de cada categoria.

Finalment, en el darrer apartat, s'ha fet una descripció dels dispositius més habituals que conformen la memòria externa d'un computador i que són gestionats pel sistema operatiu mitjançant el sistema d'E/S.