


La gerarchia delle memorie

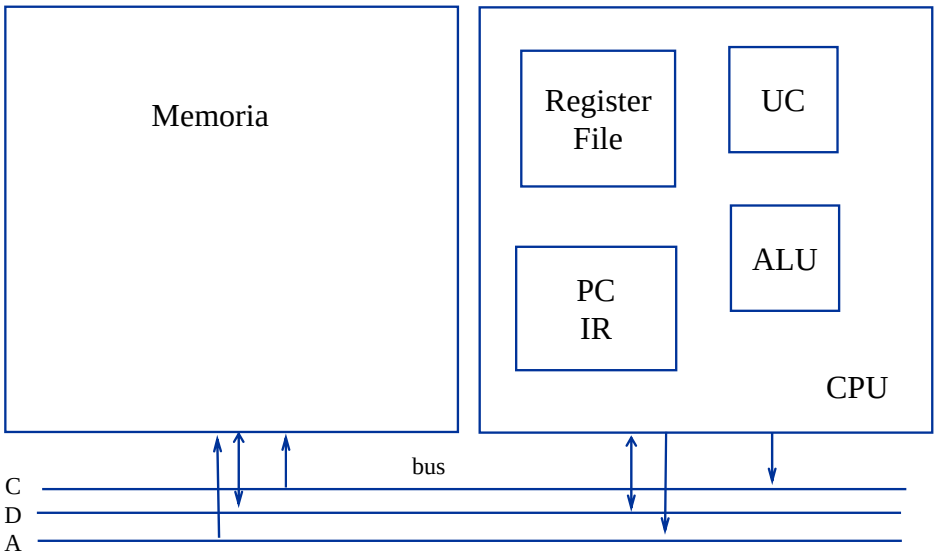
Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Università degli Studi di Milano

Riferimento Patterson: Sezioni 7.1, 7.2


Sommario

Gli attori principali di un'architettura



Caratteristiche della memoria

Posizione della memoria:

- Processore
- Interna (cache)
- Esterna (cache + Memoria Principale)
- Disco

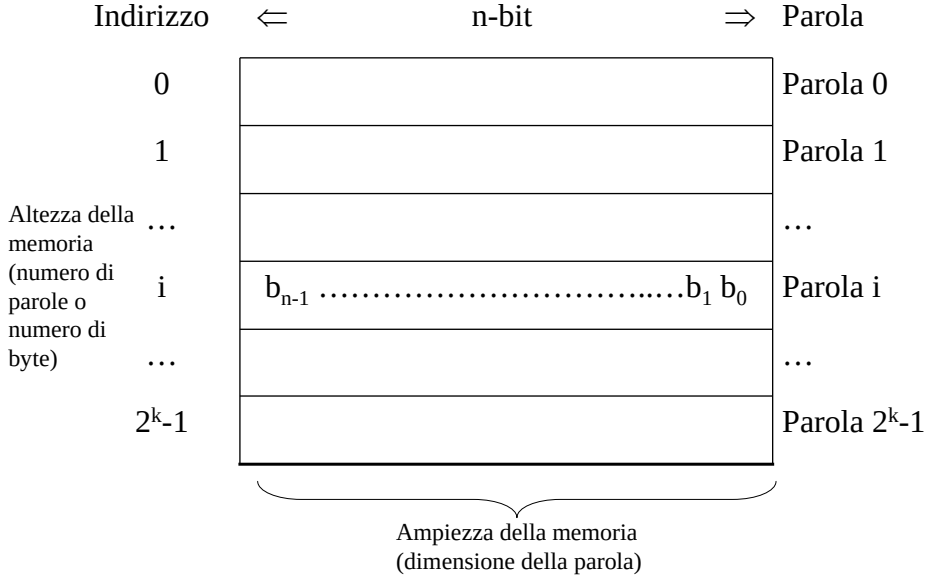
Metodo di accesso:

- Sequenziale (e.g. Nastri).
- Diretto (posizionamento + attesa, e.g. Dischi).
- Random Access (circuito di lettura / scrittura HW,tempo indipendente dalla posizione e dalla storia, e.g. Cache e Memoria principale).
- Associativa (Random Access, il contenuto viene recuperato a partire da un sottoinsieme incompleto dello stesso).

Caratteristiche fisiche:

- Nelle memorie volatili (E.g. Cache), l'informazione sparisce quando si toglie l'alimentatore (memorie a semiconduttore).
- Nelle memorie non-volatili, l'informazione è duratura (un emepio di memoria volatile è la memoria magnetica dei dischi e dei nastri). Esistono memorie a semiconduttore non-volatili (ROM).

Capacità della memoria



Capacità della memoria C = #_parole x dim_parola

Misura della capacità di una memoria

Ampiezza della memoria. Minimo numero di bit consecutivi che possono essere indirizzati.

Altezza della memoria. Numero di elementi base della memoria.

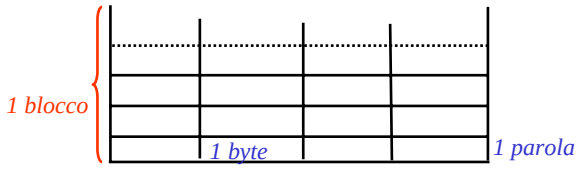
Parola di memoria. E' l'unità naturale in cui la memoria viene organizzata (e.g. 32 bit in MIPS).

Unità indirizzabile. E' il minimo numero di *unità contigue* indirizzabili. In quasi tutti i sistemi si tratta del byte.

Spazio dell'indirizzamento della memoria è: $N = \log_2$ Capacità

Unità di trasferimento:

- Blocco
- Parola



Sommario

Caratteristiche di un sistema di memoria

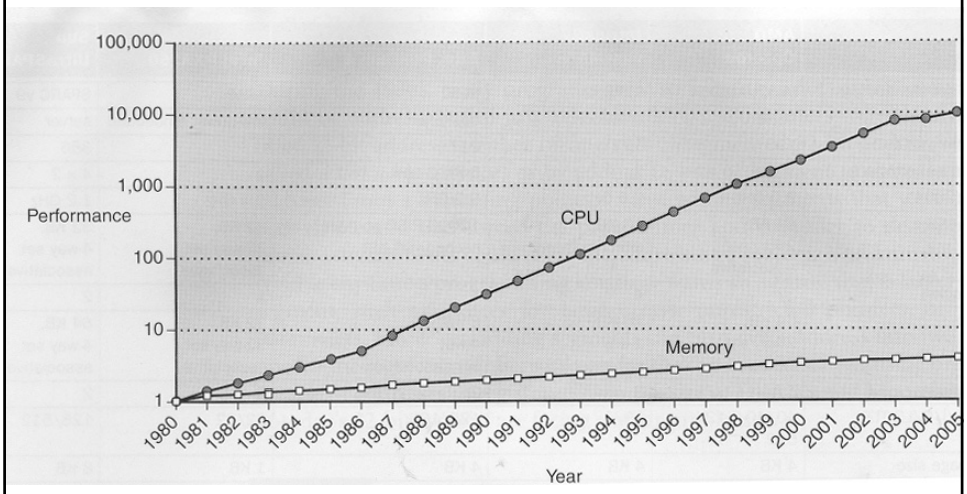
Struttura di una memoria

Principio di funzionamento di una memoria cache

Cache a mappatura diretta

Il campo tag di una cache

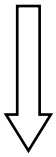
Prestazioni processore vs memoria



Principio di progettazione di una memoria

Quanta memoria?
Quanto deve essere veloce?
Quanto deve costare?

Maggiore è la velocità di accesso, maggiore il costo per bit.
Maggiore è la capacità, minore il costo per bit.
Maggiore è la capacità, maggiore è il tempo di accesso.



Memorie piccole e veloci.
Memorie grandi e lente.

Principi di località

I programmi riutilizzano dati e istruzioni che hanno usato di recente.

Regola pratica: un programma spende circa il **90%** del suo tempo di esecuzione per solo il **10%** del suo codice.

Basandosi sul passato recente del programma, è possibile predire con ragionevole accuratezza quali dati e istruzioni userà nel prossimo futuro.

Località temporale: elementi ai quali si è fatto riferimento di recente saranno utilizzati ancora nel prossimo futuro.
Località spaziale: elementi i cui indirizzi sono vicini, tendono ad essere referenziati in tempi molto ravvicinati.

Si possono organizzare programmi e dati in modo da sfruttare al massimo il principio di località (e.g. scrittura di blocchi di dati nei dischi, salti locali….).

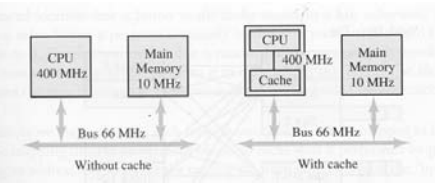
Gerarchia di memorie

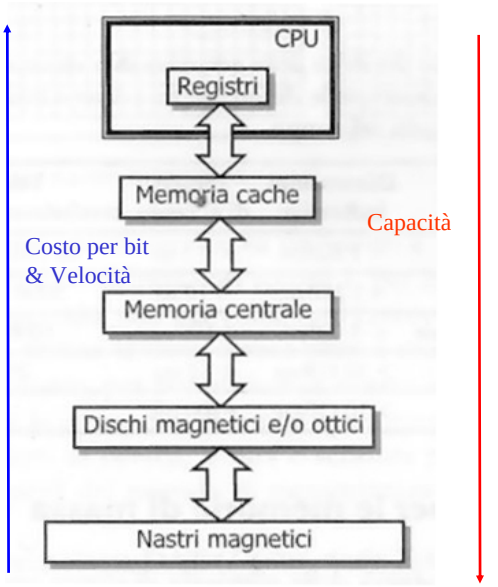
Livelli multipli di memorie con diverse dimensioni e velocità.

Nel livello superiore troviamo un sottoinsieme dei dati del livello inferiore.

Ciascun livello vede il livello inferiore e viceversa.

Pentium





Gerarchia di memorie - caratteristiche

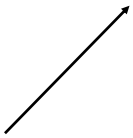
Livello	Dimensioni indicative	Tempo di Accesso	Velocità di Trasferimento (Mbyte/s)
Registri	< 1 Kbyte	< 0,01 ns	400,000 (32 byte)
Cache Primaria (Pentium 4, 3Ghz) ExecTrace cache	8kbyte 12kbyte	0.16ns	192,000 (32/64 byte)
Cache Secondaria (Pentium 4, 3Ghz)	256-512 kbyte	0.3ns	96,000 (32 byte in parallelo)
Memoria centrale (RAM, DRAM)	< 4 Gbyte	< 3-5 ns (233Mhz / 320Mhz)	1,600 – 3,000 di picco (DDSRAM – doppia lettura)
Bus PCI Bus PCI 64	8 byte 64 byte	133 Mhz 100 Mhz	133 (8 byte) 1064 (64 byte)
Dischi	> 50 Gbyte	< 10ms	< 200 (Seagate Cheetah SCSI)
Nastri	> 100 Gbyte	> 100ms	1

Principio di località in pratica

Operazione di prodotto su 2 matrici 500x500, risultato sommato a x.
Le matrici sono memorizzate in memoria principale per colonne.

```
for (i=0; i<500; i++)
{    for (j=0; j<500; j++)
     { for (k=0; k<500; k++)
            x[i][j]=x[i][j]+y[i][k]*z[k][j];
     }
}
```

```
for (k=0; k<500; k++)
{    for (j=0; j<500; j++)
     {    for (i=0; i<500; i++)
               x[i][j]=x[i][j]+y[i][k]*z[k][j];
     }
}
```



Devo continuare a caricare i nuovi valori di y[i][k] dalla memoria principale.

Raddoppio la velocità di esecuzione. Percorro x e y per colonne, z[.], cambia ogni 500 valori, assieme a y e x, ho tempo di caricare i nuovi blocchi.

Prestazione di una memoria

Tempo di accesso.

Per una memoria a random-access, è il tempo richiesto per eseguire un'operazione di lettura / scrittura. Più precisamente è il tempo che intercorre dall'istante in cui l'indirizzo si presenta alla porta di lettura della memoria all'istante in cui il dato diventa disponibile.

Per una memoria non random-access, è il tempo richiesto per posizionare il dispositivo di lettura / scrittura in corrispondenza dell'informazione da leggere / scrivere.

Memory cycle time: si applica ad una memoria random-access. E' il tempo di accesso più il tempo necessario perchè possa avvenire un secondo accesso a memoria.

Transfer rate: quantità di informazione trasferita nell'unità di tempo si misura in:

Random-access memory = 1 / Memory_cycle_time (unità di misura Hertz)

Not random-access memory = 1 / [T_A + N / R] con R trasfer rate (tempo di 1 ciclo di memoria in cui si trasferiscono N byte).

Numero_dati (in byte) / s.

Tassonomia del funzionamento

HIT Successo nel tentativo di accesso ad un dato: è presente al livello superiore della gerarchia.

MISS Fallimento del tentativo di accesso al livello superiore della gerarchia => il dato o l'indirizzo devono essere cercati al livello inferiore.

HIT_RATE Percentuale dei tentativi di accesso ai livelli superiori della gerarchia che hanno avuto successo.
HIT_RATE = Numero_successi / Numero_accessi_memoria

MISS_RATE Percentuale dei tentativi di accesso ai livelli superiori della gerarchia che sono falliti
MISS_RATE = Numero_fall. / Numero_accessi_memoria

HIT_RATE + MISS_RATE = 1

HIT TIME Tempo richiesto per verificare se il blocco è presente al livello attuale della memoria.

MISS_PENALTY Tempo richiesto per sostituire il blocco di memoria mancante al livello superiore + tempo necessario per inviarlo al processore.

Sommario

Caratteristiche di un sistema di memoria

Struttura di una memoria

Principio di funzionamento di una memoria cache

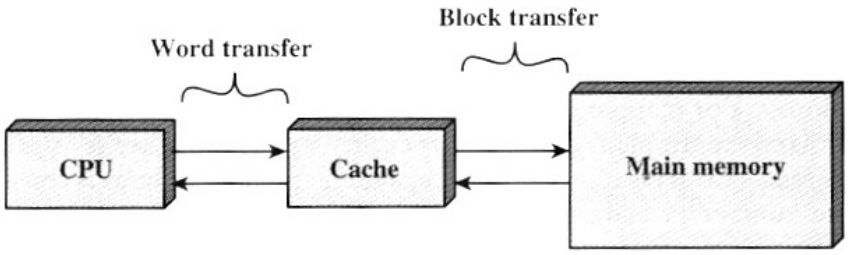
Cache a mappatura diretta

Il campo tag di una cache

Principio di funzionamento di una cache

Scopo: fornire alla CPU una velocità di trasferimento pari a quella della memoria più veloce con una capacità pari a quella della memoria più grande.

Una cache "disaccoppia" i dati utilizzati dal processore da quelli memorizzati nella Memoria Principale.



Word transfer: Data transfer or Instruction transfer. In MIPS = 1 parola.

La cache contiene una copia di parte del contenuto della memoria principale. Di che cosa?

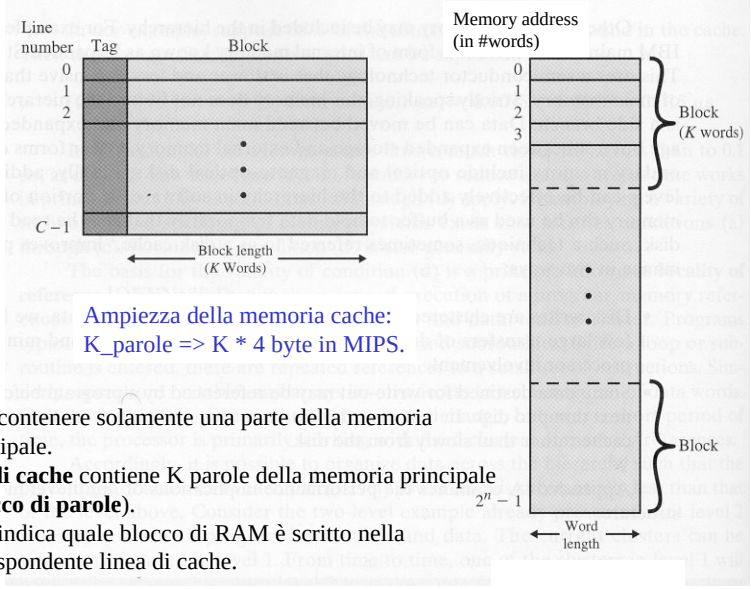
Sottosistema di memoria

Porta nella cache primaria i dati richiesti mentre il binomio processore-memoria sta lavorando.

1) Controlla se una parola è in cache (Hit).
2) Porta una parola (e quelle vicine) in cache, prelevandole dal livello inferiore (Miss):

Contenuto della cache

Altezza della memoria cache: # di linee



Ampiezza della memoria cache: K_parole => K * 4 byte in MIPS.

- La cache può contenere solamente una parte della memoria principale.
- Ogni **parola di cache** contiene K parole della memoria principale (**blocco di parole**).
- Il campo Tag indica quale blocco di RAM è scritto nella corrispondente linea di cache.

Sommario

Caratteristiche di un sistema di memoria

Struttura di una memoria

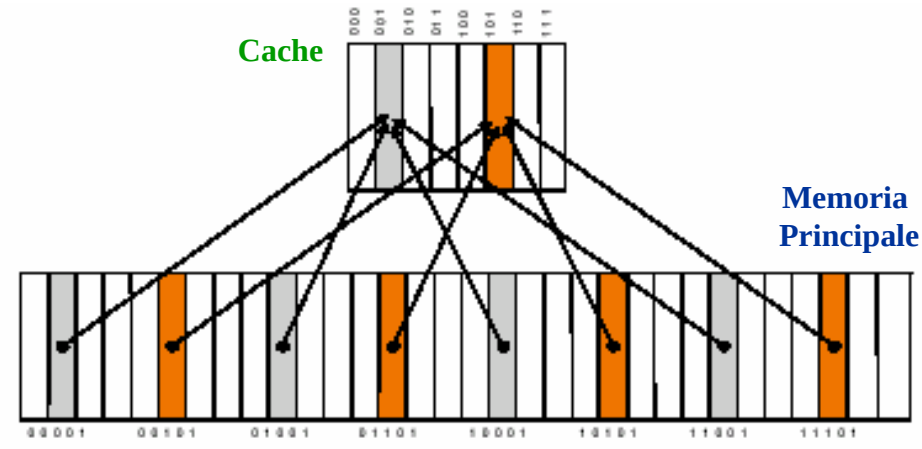
Principio di funzionamento di una memoria cache

Cache a mappatura diretta

Il campo tag di una cache

Corrispondenza diretta (direct mapped)

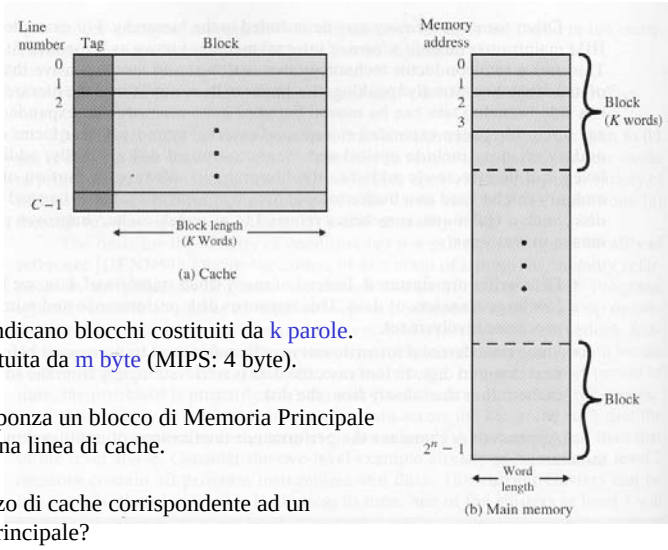
Ad ogni indirizzo di Memoria Principale corrisponde un indirizzo di cache.



Indirizzi diversi di Memoria Principale corrispondono allo stesso indirizzo di cache.
Quali indirizzi della memoria principale si considerano?

Determinazione della legge di corrispondenza

3 unità di misura:
Blocco di cache.
Parola.
Byte.



Le linee di una cache indicano blocchi costituiti da k parole.
Ciascuna parola è costituita da m byte (MIPS: 4 byte).

Posso mettere in corrisponza un blocco di Memoria Principale di n = k * m byte con una linea di cache.

Come ottengo l'indirizzo di cache corrispondente ad un indirizzo di memoria principale?

Come si ottiene l'indirizzo di cache?

lw $t0, 12($t1) - carico in cache tutto il blocco associato all'indirizzo $t1 + 12.

Identifico il blocco di Memoria principale a cui appartiene il byte da leggere / scrivere.
Associo il blocco di Memoria principale ad una linea di cache mediante operazione di modulo.

Posizione_byte_cache = indirizzo_Memoria principale (in #byte) *modulo* #byte_cache.

Utilizzo come range di conteggio in RAM, il blocco di cache.

Note:
L'operazione di modulo se la dimensione del blocco è multiplo della base di conteggio si ottiene scartando alcune delle cifre meno significative.
La divisione per un multiplo di 2 si ottiene come shift a sx, le cifre che "escono" rappresentano il resto.

Esempio

La cache con linee di ampiezza pari a 4 parole (blocco = 4 parole) ed altezza di 8 linee:
Il blocco di dati della memoria principale che può essere contenuto in ogni linea di cache, ha dimensioni:
n = 4 parole * 4 byte = 16 byte.
La capacità della cache sarà C = **8 linee** * n = **8** * 16 = 128 byte.

•lw $t0, 70($zero) – 70 è il 7º byte della 5ª linea (70 / 16) = 4, **∈ 5ª linea della cache**.

Indirizzo_cache = Indirizzo_Memoria principale *modulo* blocco_cache
Indirizzo_cache = 70 / 16 = 4 → resto 6. {Il resto indica la posizione del primo byte della parola all'interno della linea di cache (blocco) ➔ 7º byte = 3º byte della 2ª parola.}
La word letta è costituita dal byte 70, e dai byte 68, 69, 71, contenuti nella 2ª parola della 5ª linea della cache.

•lw $t0, 204($zero) – 204 / 128 byte = 1 (resto = 76) => mappiamo il 2º blocco di RAM sulla cache.
204 è il 13º byte della 13ª linea (204 / 16) = 12, **∈ 13ª linea (8 + 5)**. La memoria cache ha solo 8 linee.

Indirizzo_cache = Indirizzo_memoria principale (**relativo all'inizio del blocco**) *modulo* capacità_blocco → 76 / 16 = 4 → resto 12.{Il resto indica la posizione del primo byte della parola all'interno della linea di cache ➔ 13º byte.}
Il dato viene letto (trasferito nella CPU) assieme ai byte 205, 206, 207) nella stessa linea 5ª della cache.
Il dato viene recuperato (dalla RAM) assieme alle word di indirizzo 200, 196, 192) nella stessa linea 5ª della cache.

Indirizzamento scartando i bit più significativi

Indirizzo cache	Indirizzo decimale RAM	Indirizzo binario RAM
111	112-127, 240-255, 368-383,...	00 0111 0000 - 00 0111 1111
110	96, 224, 352	00 0110 0000 - 00 0110 1111
101	80-95, 208, 336...	00 0101 0000 - 00 0101 1111
100	64-79, 192-207, 320-335, 448-461,...	00 0100 0000 - 00 0100 1111
011	48-63, 176, 304...	00 0011 0000 - 00 0011 1111
010	32-47, 160, 288...	00 0010 0000 - 00 0010 1111
001	16-31, 144, 272...	00 0001 0000 - 00 0001 1111
000	0-15, 128, 256, 384,...	00 0000 0000 - 00 0000 1111

NB: La capacità della cache è di: 8 * 16 byte = 128 byte = x000 0000 – x111 1111

Indirizzamento scartando i bit più significativi (2° blocco di RAM)

Indirizzo cache	Indirizzo decimale RAM	Indirizzo binario RAM
111	112-127, 240-255, 368-383,...	00 1111 0000 - 00 1111 1111
110	96, 224, 352	00 1110 0000 - 00 1110 1111
101	80-95, 208, 336...	00 1101 0000 - 00 1101 1111
100	64-79, 192-207, 320-335, 448-461,...	00 1100 0000 - 00 1100 1111
011	48-63, 176, 304...	00 1011 0000 - 00 1011 1111
010	32-47, 160, 288...	00 1010 0000 - 00 1010 1111
001	16-31, 144, 272...	00 1001 0000 - 00 1001 1111
000	0-15, 128, 256, 384,...	00 1000 0000 - 00 1000 1111

NB: La capacità della cache è di: 8 * 16 byte = 128 byte = x1000 0000 – x1111 1111

Indirizzamento scartando i bit più significativi (3° blocco di RAM)

Indirizzo cache	Indirizzo decimale RAM	Indirizzo binario RAM
111	112-127, 240-255, 368-383,...	01 0111 0000 - 01 0111 1111
110	96, 224, 352	01 0110 0000 - 01 0110 1111
101	80-95, 208, 336...	01 0101 0000 - 01 0101 1111
100	64-79, 192-207, 320-335, 448-461,...	01 0100 0000 - 01 0100 1111
011	48-63, 176, 304...	01 0011 0000 - 01 0011 1111
010	32-47, 160, 288...	01 0010 0000 - 01 0010 1111
001	16-31, 144, 272...	01 0001 0000 - 01 0001 1111
000	0-15, 128, 256, 384,...	01 0000 0000 - 01 0000 1111

NB: La capacità della cache è di: 8 * 16 byte = 128 byte = x1 0000 0000 – x1 0111 1111

Parsing dell'indirizzo

0000 0000 0000 00(00) ⟹ 0000 0000 0111 11(11)

128 indirizzi diversi di RAM (32 parole sequenziali di 4 byte)

•Questi 128 byte vengono messi in corrispondeza con i 128 byte di una cache che è organizzata come 8 x 16: 4 parole per linea.

•Gli ultimi 2 bit a destra indicano i byte interni alla parola. Non vengono considerati nei trasferimenti da / per la memoria. Viene considerata la *Word*.

•I 2 bit seguenti indicano la posizione della parola ricercata all'interno della linea della cache.

•I 3 bit seguenti indicano la linea della cache nella / dalla quale si scrive / legge.

•I bit seguenti indicano il numero del blocco della memoria principale messo in corrispondenza con la cache.

Metodo di parsing dell'indirizzo

0000 0000 0000 00(00) ⟹ 0000 0000 0111 11(11)

128 indirizzi diversi di RAM (32 parole sequenziali di 4 byte)

Questo vale per:
- Cache di 8 linee.
- Linee di 4 parole.
- Parole di 4 byte.

Prendiamo un indirizzo ed operiamo per divisioni successive nel caso più generale:

Capacità della cache

M / [(#byte / parola) * (#parole / linea) * #linee] = Numero_blocco di RAM.
Il resto, R_1, rappresenta l'offset in byte all'interno della cache a partire dal byte 0 di cache.

R_1/ [(#byte / parola) * (#parole / linea)] = Numero di linea (di blocco) di cache.
Il resto, R_2, rappresenta l'offset in byte all'interno della linea di cache.

R_2/ [(#byte / parola)] = Numero di parola all'interno della linea di cache.
Il resto, R_3, rappresenta l'offset in byte all'interno della parola.

Esempio di parsing dell'indirizzo

0000 0000 0000 00(00) ⟹ 0000 0000 0111 11(11)

128 indirizzi diversi (32 parole di 4 byte)

La cache con linee di 4 parole (ampiezza) ed altezza di 8 linee:
Il blocco di dati contenuto in ogni linea di cache è di dimensioni: n = 4 * 4 byte = 16 byte.
La capacità della cache è di 8 * 16 byte = 128 byte.

lw $t0, 196($zero)

196 / [4 * 4 * 8] = 1 (2º blocco di RAM) con resto R_1 = 196 – 1 * 128 = 68.
Il resto, R_1, rappresenta l'offset in byte all'interno della cache.

68/ [4 * 4] = 4 (5ª linea della cache) con resto R_2 = 68 – 4 * 16 = 4.
Il resto, R_2, rappresenta l'offset in byte all'interno della linea di cache.

4/ 4 = 1 (2ª parola della cache) con resto R_3 = 4 – 1*4 = 0.
Il resto, R_3, rappresenta l'offset in byte all'interno della parola.

Esempio di parsing dell'indirizzo

0000 0000 0000 00(00) ⟹ 0000 0001 1111 11(11) 512 indirizzi diversi (128 parole, 4 byte)

La cache con linee di 8 parole (ampiezza) ed altezza di 16 linee:
Il blocco di dati contenuto in ogni linea di cache è di dimensioni: n = 8 * 4 byte = 32 byte.
La capacità della cache è di 16 * 32 byte = 512 byte.

lw $t0, 600($zero)

600 / [16 * 8 * 4] = 1 (2° blocco di RAM) con resto R_1 = 600 – 1 * 512 = 88.
Il resto, R_1, rappresenta l'offset in byte all'interno della cache.

88 / [8 * 4] = 2 (3ª linea della cache) con resto R_2 = 88 – 2 * 32 = 24.
Il resto, R_2, rappresenta l'offset in byte all'interno della linea di cache.

24/ 4 = 6 (7ª parola della linea di cache) con resto R_3 = 24 – 6*4 = 0.
Il resto, R_3, rappresenta l'offset in byte all'interno della parola.

Sommario

Caratteristiche di un sistema di memoria

Struttura di una memoria

Principio di funzionamento di una memoria cache

Cache a mappatura diretta

Il campo tag di una cache

Come si può sapere se un dato è presente in cache?

Aggiungiamo a ciascuna delle linee della cache un campo **tag**.

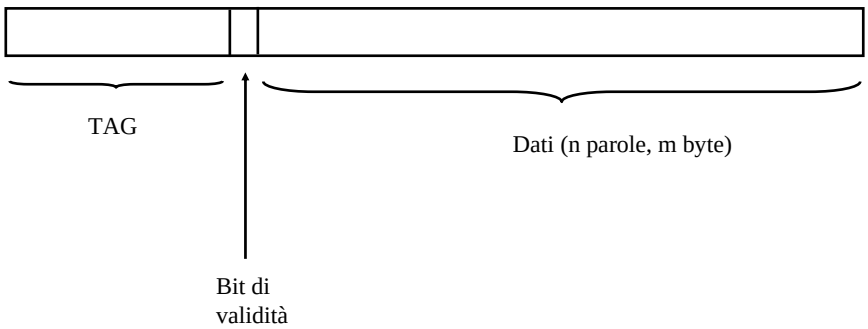
Il tag contiene i bit che costituiscono la parte più significativa dell'indirizzo e rappresenta il numero di blocco di RAM in cui il dato di cache è contenuto.
Esso è costituito da K bit:

$$K = M - \sup(\log_2 \text{Capacità\_cache (byte)})$$

Nell'esempio precedente: $K = 32 - \sup(\log_2 512) = 23$ bit.

Occorre inoltre l'informazione data_valid / data_not_valid: bit di validità.

Struttura di un blocco di cache



Quanti blocchi di RAM avremo delle dimensioni della cache?

Nel caso precedente, avremo linee di cache di lunghezza:
25 (lunghezza_campo_TAG) + 1 + 4 (parole) * 4 (byte/parola) * 8(bit/byte) = 154bit.

Come leggere / scrivere su cache

Individuare la linea della cache dalla quale leggere / scrivere (operazione analoga all'indirizzamento del register file).

Confrontare il campo tag con il blocco RAM in cui risiede il dato.

Controllare il bit di validità.

Leggere (scrivere) il dato.

Per blocchi più ampi di una parola, occorre individuare una parola tra le k presenti nella linea di cache.

Sommario

Caratteristiche di un sistema di memoria

Struttura di una memoria

Principio di funzionamento di una memoria cache

Cache a mappatura diretta

Il campo tag di una cache