


La gerarchia delle memorie

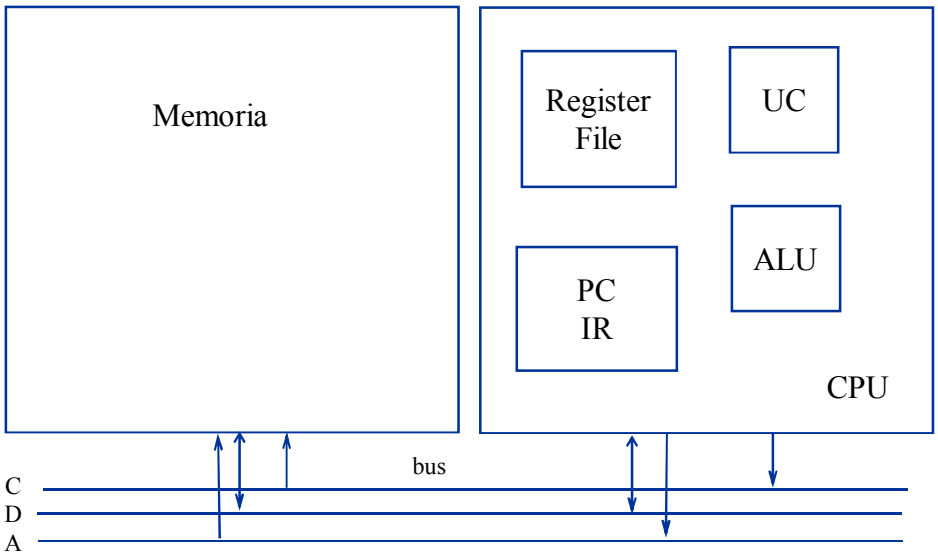
Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimento Patterson: Sezioni 5.1, 5.2

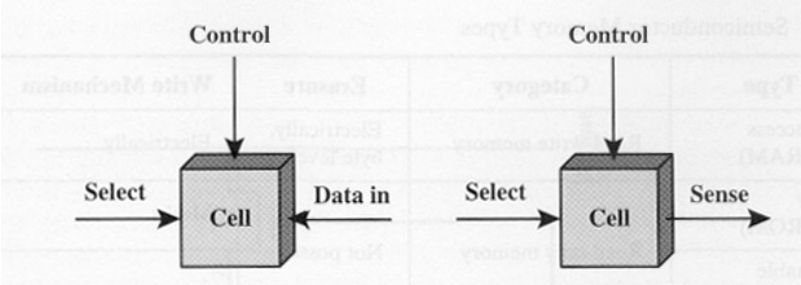

Sommario

Gli attori principali di un'architettura



Cella di memoria

La memoria è suddivisa in celle, ciascuna delle quali assume un valore binario stabile.
Si può scrivere il valore 0/1 in una cella.
Si può leggere il valore di ciascuna cella.

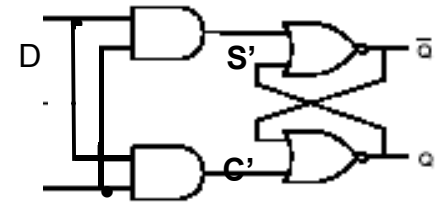


Quale struttura di memoria abbiamo già incontrato?

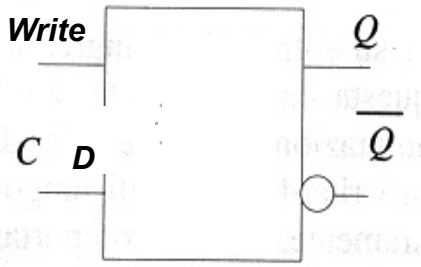
Control (lettura – abilitazione; scrittura)
Select (cf. dataport)
Data in & Sense (Data in & Data out).

Cella SRAM

E' trasparente quando Write = 1
Se Write = 1 $Q_{t+1} = D$
Se Write = 0 $Q_{t+1} = Q_t$

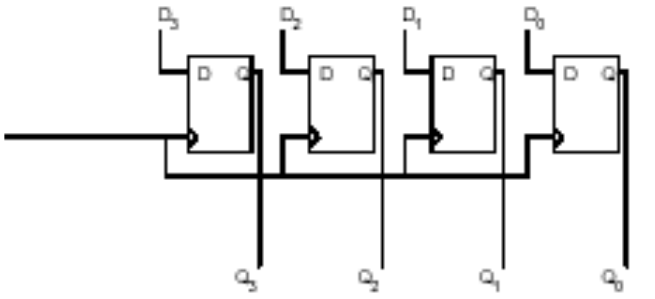


Clk = Write



Selezione (porta di lettura e porta di scrittura)
Lettura - sempre disponibile in uscita
Scrittura – segnale esplicito (in AND con il clock in caso di cella sincrona).

Registri

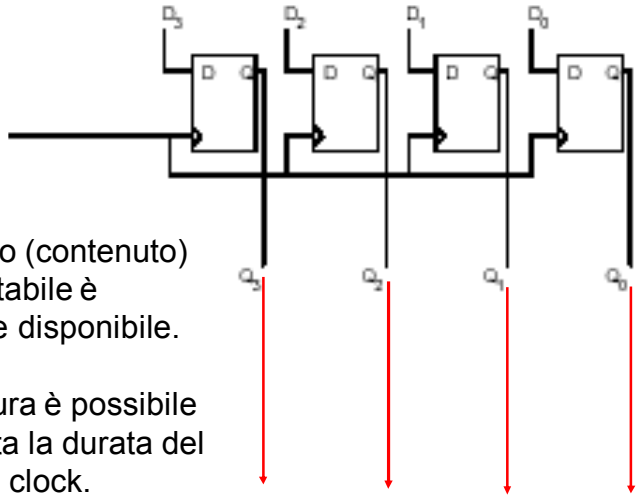


Un registro a 4 bit.
Memorizza 4 bit.

Latch di tipo D

NB Non è un registro a scorrimento (shift register!)

Lettura di un registro

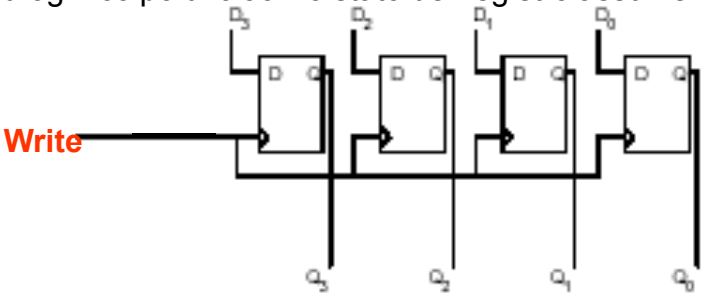


Lo stato (contenuto) del bistabile è sempre disponibile.

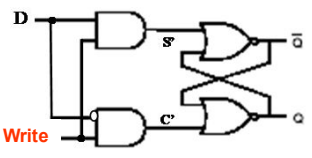
La lettura è possibile per tutta la durata del ciclo di clock.

Scrittura di un registro

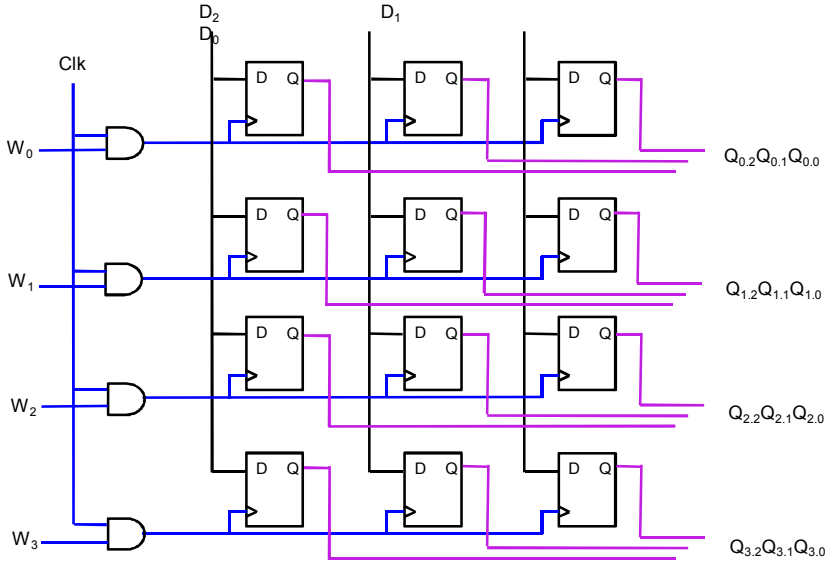
Ad ogni colpo di clock lo stato del registro assume il valore dell'i



Cosa occorre modificare perchè il registro venga scritto quando serve?
Introdurre una sorta di *"apertura del cancello (chiusura circuito)"* .
Può essere sincronizzata o meno con il clock.



Un banco 4 registri x 3bit

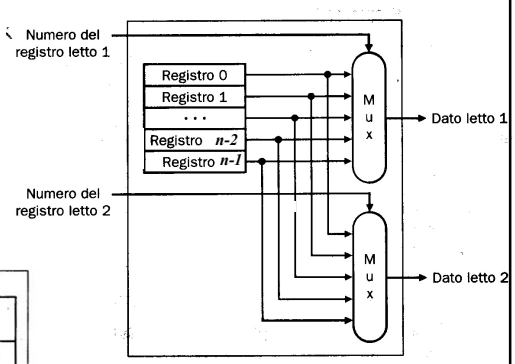


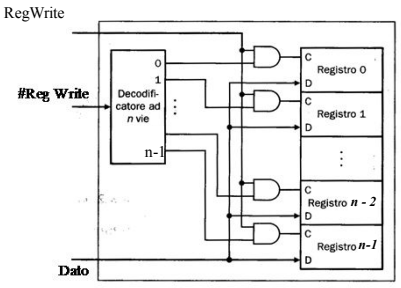
Register file

Il tempo di lettura dipende dal cammino critico dei Mux.

Il tempo di scrittura dipende dal cammino critico del Decoder.

Numero_registro = selettore.





Selezione – #registro
Lettura .- sempre disponibile in uscita (dopo tempo di commutazione del MUX)
Scrittura – segnale esplicito (in AND con il clock in caso di cella sincrona).

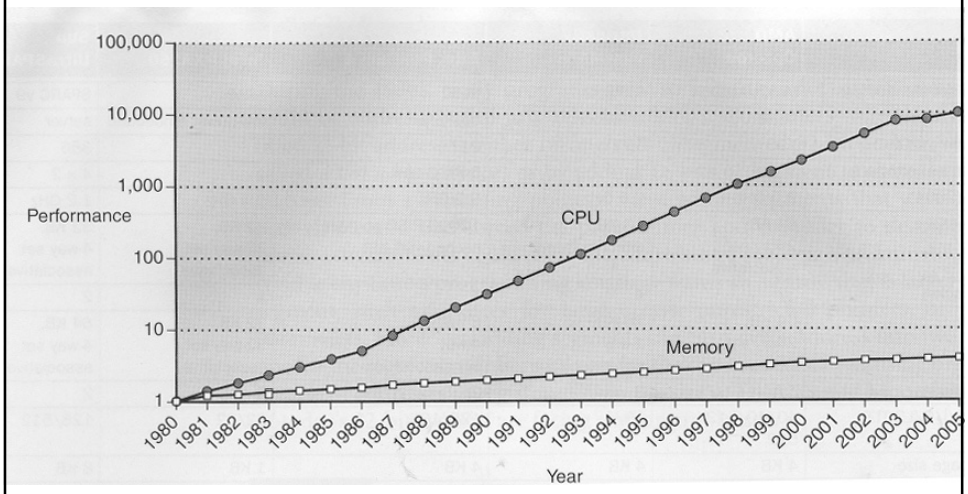
Principio di progettazione di una memoria

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.

Prestazioni processore vs memoria



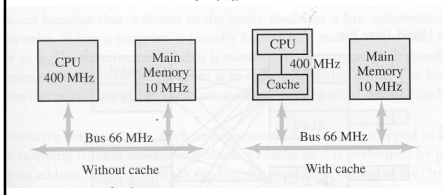
Gerarchia di memorie

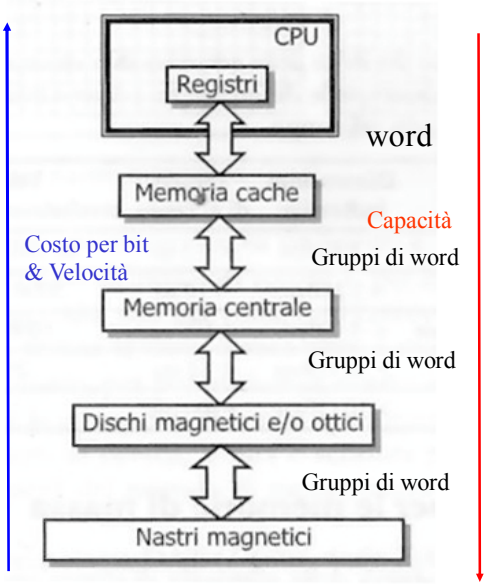
Livelli multipli di memorie con diverse dimensioni e velocità.

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

Ciascun livello vede il livello inferiore e viceversa.

Pentium





Altre memorie della gerarchia oltre ai registri

RAM

Dischi


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	> 100 Gbyte	< 10ms	< 200 (Seagate Cheetah SCSI)
Nastri	> 1 Tbyte (Pbyte)	> 100ms	1


Gerarchie di cache

L – Livelli di cache

L1 interna

L2 interna

L3 esterna

Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KB each for instructions/data per core	64 KB each for instructions/data per core
L1 cache associativity	4-way (I), 8-way (D) set associative	2-way set associative
L1 replacement	Approximated LRU replacement	LRU replacement
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L1 hit time (load-use)	Not Available	3 clock cycles
L2 cache organization	Unified (instruction and data) per core	Unified (instruction and data) per core
L2 cache size	256 KB (0.25 MB)	512 KB (0.5 MB)
L2 cache associativity	8-way set associative	16-way set associative
L2 replacement	Approximated LRU replacement	Approximated LRU replacement
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	Not Available	9 clock cycles
L3 cache organization	Unified (instruction and data)	Unified (instruction and data)
L3 cache size	8192 KB (8 MB), shared	2048 KB (2 MB), shared
L3 cache associativity	16-way set associative	32-way set associative
L3 replacement	Not Available	Evict block shared by fewest cores
L3 block size	64 bytes	64 bytes
L3 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L3 hit time	Not Available	38 (?)clock cycles

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).

Split cache

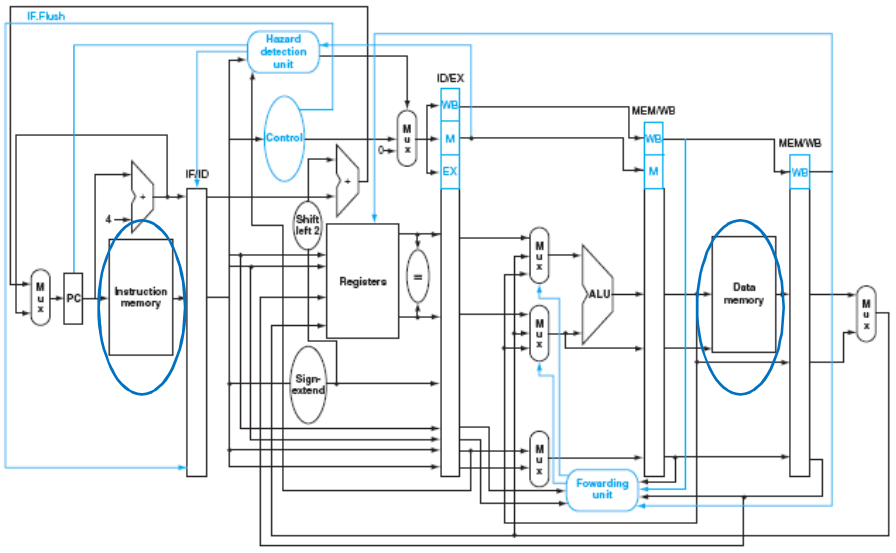
Split-cache: Cache dati e cache istruzioni.

Vantaggi. Possibilità di analizzare le istruzioni in coda (contenute nella cache istruzioni) mentre si eseguono altre istruzioni (che lavorano su dati contenuti nella cache dati), senza dovere competere per l'accesso alla cache. Efficiente per le architetture superscalari.

Svantaggi. Minore hit rate, perchè non si sfrutta al meglio la memoria cache. Si potrebbe riempire un'unica cache maggiormente con dati od istruzioni a seconda del frammento di codice correntemente in esecuzione.

Il register spilling e le miss sono inevitabili ➔ Come fare vedere al processore una memoria sufficientemente veloce?

CPU con pipeline



Principi di località

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

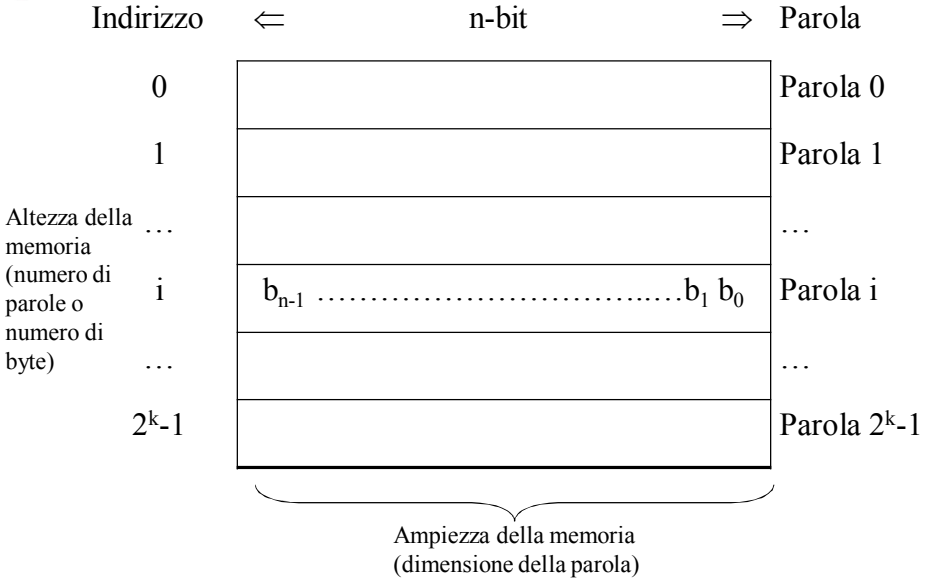
Regola pratica: un programma spende circa il **90%** del suo tempo di esecuzione per solo il **10%** del suo codice.

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….).

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 (Memoria Principale 1 byte; Register file: 1 word); cache: K word).

Altezza della memoria. Numero di elementi base della memoria.

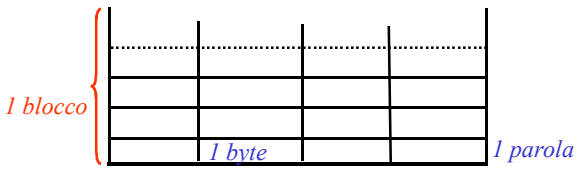
Parola di memoria. E' l'unità naturale in cui la memoria viene organizzata (e.g. 32 bit in MIPS).

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

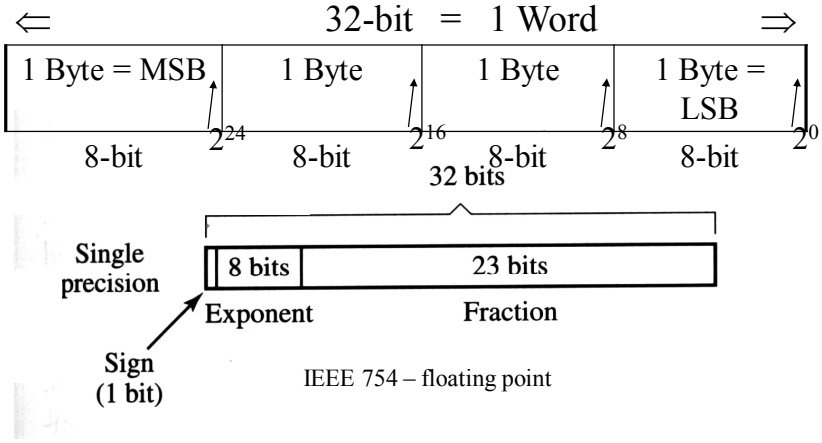
Unità di trasferimento:

- Blocco
- Parola



Indirizzamento dei byte all'interno della parola

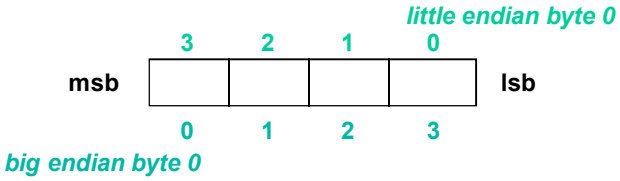
MIPS utilizza un **indirizzamento al byte**, cioè l'indice punta ad un byte di memoria, byte consecutivi hanno indirizzi consecutivi indirizzi di parole consecutive (adiacenti) differiscono di un fattore 4 (8-bit x 4 = 32-bit): ad ogni indirizzo è associato un byte.



IEEE 754 – floating point

Addressing Objects: Endianess

- Big Endian: address of most significant byte = word address (xx00 = Big End of word)
  - IBM 360/370, Motorola 68k, MIPS, Sparc, HP
- Little Endian: address of least significant byte = word address (xx00 = Little End of word)
  - Intel 80x86, DEC Vax, DEC Alpha (Windows NT)



Disposizione in memoria::little endian

Disposizione in memoria::big endian

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.

Criteri di progettazione

Cache primaria: massimizzo Hit rate.

Cache secondaria: minimizzo Miss penalty (massimizzo transfer rate).

Valutazione prestazioni memoria

Obbiettivo principale della gerarchia di memoria è incrementare le prestazioni => diminuire la velocità di accesso sia in caso di HIT che di MISS.

Cosa succede in caso di MISS?

HIT_TIME Tempo di accesso al livello superiore (che comprende anche il tempo necessario per determinare se l'accesso ha avuto successo oppure fallisce.

MISS_PENALTY è composto da:
TEMPO DI ACCESSO per accedere alla prima parola del blocco dopo che è stato rilevato il fallimento.
TEMPO DI TRASFERIMENTO per trasferire le altre parole del blocco al livello superiore.

MISS_TIME = > HIT_TIME + MISS_PENALTY

Gestione dei fallimenti di una cache

Hit – è quello che vorremmo ottenere, il funzionamento della CPU non viene alterato.

Miss – in lettura devo aspettare che il dato sia pronto in cache -> stallo.

Passi da eseguire in caso di Miss (fase Mem):
1) Ricaricare la pipeline (indirizzo dell'istruzione (PC-> PC-4), decodifica, esecuzione, memoria…)
2) Leggere il blocco di memoria dalla memoria principale.
3) Trasferire il blocco in cache, aggiornare i campi validita' e tag.
4) Avviare la fase di fetch, decodifica, esecuzione, memoria dell'istruzione con i dati corretti. .

NB Il programma non può continuare!!

In scrittura?

Sommario

Struttura di un sistema di memoria memoria

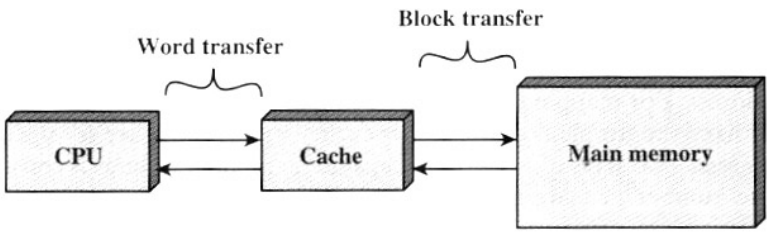
Cache a mappatura diretta

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.

La memoria è un sottosistema a sè che cerca di mantenere in cache i dati di cui ha bisogno il processore. Meccanismi di speculazione analoghi a quelli utilizzati dalla CPU. Per speculazioni errate, il MISS_PENALTY può essere elevato e la CPU può andare in stallo per un tempo considerevole.
Una cache "disaccoppia" i dati utilizzati dal processore da quelli memorizzati nella Memoria Principale.

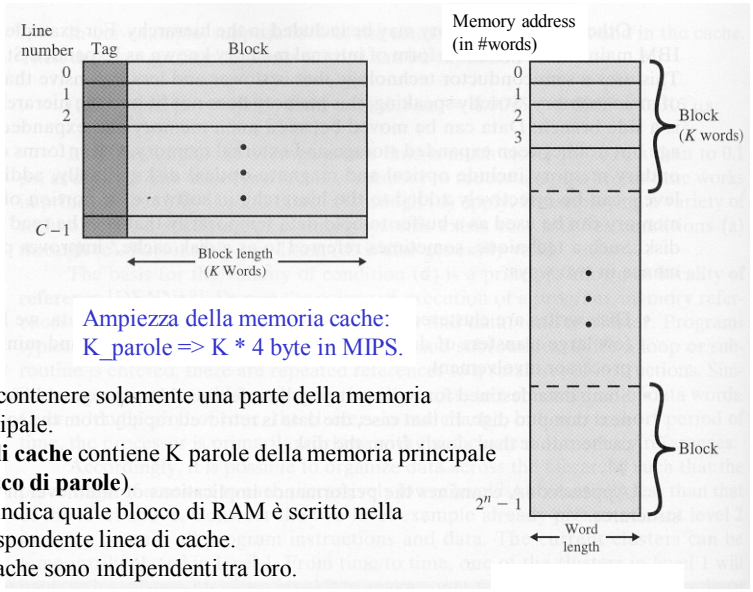


Word transfer: Data transfer or Instruction transfer. In MIPS = 1 parola.

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

Contenuto della cache

Altezza della memoria cache: # di linee

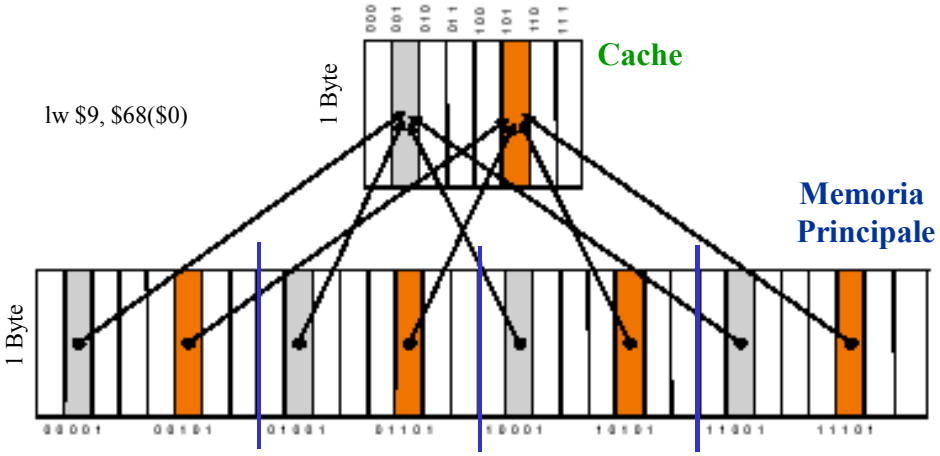


Ampiezza della memoria cache: K_parole => K * 4 byte in MIPS.

- 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.
- Le parole di cache sono indipendenti tra loro.

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?

Corrispondenza diretta

lw $9, $68($0)

Supponiamo di avere una cache con 8 linee di un byte => Capacità_Cache = 8

Indirizzo = N_blocchi * Capacità_Cache + offset_linea
68 = ?? * 8 + ??

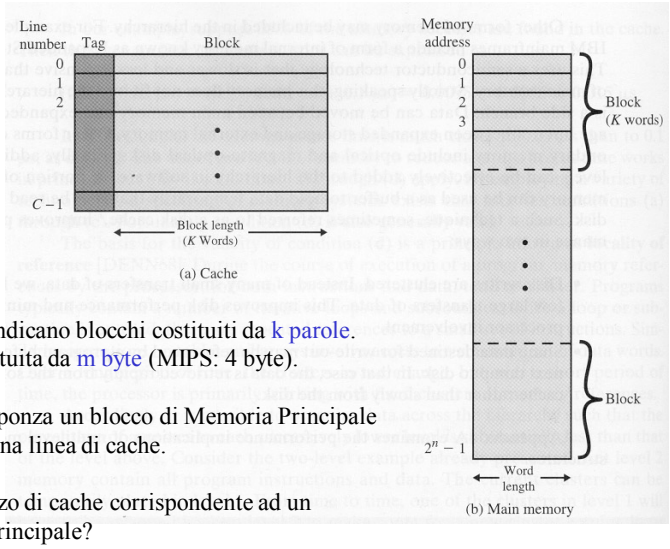
Indirizzo / Capacità_Cache = N_blocchi
Resto = N_linea

68 / 8 = 8 occore "saltare" N_blocchi, il dato appartiene al blocco 9 della memoria principale

R = 4 occorre "saltare" 4 posizioni all'interno della cache, il dato si trova nel 5° byte della cache.

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? (Metodo delle divisioni successive)

lw $t0, 14($t1) - carico in cache tutto il blocco associato all'indirizzo $t1 + 14.

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 (# TAG).
Resto1 = posizione in cache.

Posizione_linea = Resto1 (in #byte) *modulo* #byte_linea.
Utilizzo come range di conteggio in Cache, la linea di cache (# linea).
Resto2 = posizione all'interno della linea.

Posizione_byte_linea = Resto 2 (in # byte) *modulo* #byte_word.
Utilizzo come range di conteggio nella linea la dimensione della parola (# parola).
Resto3 = posizione del byte all'interno della parola.

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_byte_x_linea = 4 parole * 4 byte = 16 byte.
La capacità della cache sarà C = **8 linee** * n = **8** * 16 = 128 byte.

•lw $t0, 72($zero) – 72 è il 9° byte della 5ª linea (72 / 16) = 4, **∈ 5ª linea della cache**.

Indirizzo_cache = Indirizzo_Memoria principale *modulo* blocco_cache
Indirizzo_cache = 72 / 16 = 4 → resto 8. {Il resto indica la posizione del primo byte della parola all'interno della linea di cache (blocco) ➔ 9° byte = 1° byte della 3ª parola.}
La word letta è costituita dal byte 72, e dai byte 73, 74, 75, contenuti nella 3ª parola della 5ª linea della cache.

•lw $t0, 204($zero) – 204 / 128 byte = 1 (resto = 76) => mappiamo il 2° blocco di RAM sulla cache.
204 è il 13° 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 ➔ 13° 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

Sommario

Struttura di un sistema di memoria

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**.

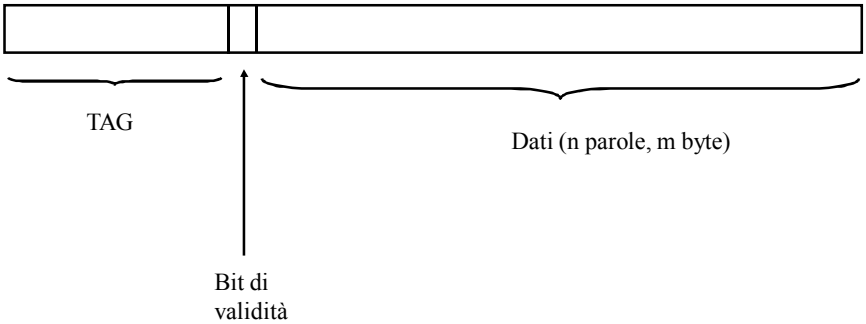
Il tag contiene i bit che costituiscono la parte più significativa dell'indirizzo e rappresenta il numero di blocco di RAM in cui il dato di cache è contenuto.
Esso è costituito da K bit:

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

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



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

Quanti blocchi di RAM avremo delle dimensioni della 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