UNIVERSITÀ DEGLI STUDI DI PARMA

Corso di Laurea in

"Ingegneria Informatica, Elettronica e delle Telecomunicazioni"

Architettura dei Calcolatori Elettronici

Memorie

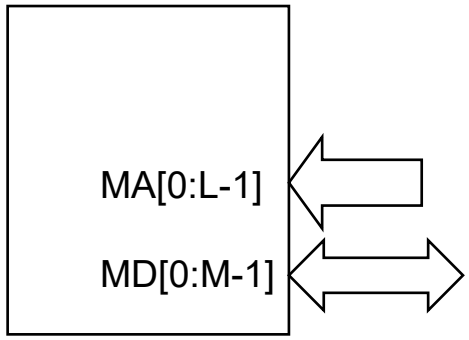
Andrea Prati

Memorie (1/4)

- **Memoria**: unità logica di memorizzazione di dati nel calcolatore.

- Esistono diversi tipi di memorie che possono essere classificate in base a:

1. **Capacità**
2. **Caratteristiche fisiche**
3. **Organizzazione**
4. **Modalità di Accesso**
5. **Prestazioni**



1. Capacità:
   - numero di parole, N, accedute con L linee con L = $\log_2$ N
   - dimensione delle parole M
     - Normalmente consideriamo M=8 o M=1
     - Spesso la dimensione delle parole è maggiore per la presenza di bit di ridondanza (es: bit di parità)

Memorie (2/4)

2. Caratteristiche fisiche:
- Tipo :
  - a semiconduttore,
  - a superficie magnetica,
  - Ottica,…
- Consumo:
  - Dipende dalla tecnologia (può comportare la necessità di sistemi di raffreddamento)
- Affidabilità:
  - è di solito misurata dal tempo medio tra 2 guasti (Mean Time Between Failure – **MTBF**).
- Alterabilità:
  - cancellabile/non-cancellabile
- Durevolezza:
  - volatile/non-volatile

3. Organizzazione:
  - posizione nella gerarchia di memorie
  - parallelismo, interallacciamento, ...

Memorie (3/4)

4. Modalità di Accesso:

- A seconda della modalità di accesso si hanno diverse prestazioni e diversi tempi di accesso:
- **Sequenziale:**
  - per accedere ad un dato in una fissata posizione (locazione) della memoria è necessario accedere a tutti i dati memorizzati in posizioni precedenti (es. i tape)
- **Diretto**:
  - è possibile avere un accesso direttamente alla locazione di memoria voluta senza dovere accedere alle precedenti. In molti casi però il tempo di accesso è dipendente dalla posizione e dalla locazione letta o scritta precedentemente (es. Hard Disk in cui il seek time – tempo di ricerca – dipende dalla posizione della testina)
- **Casuale**:
  - è un accesso diretto in cui il tempo di accesso è costante e non dipende dall'indirizzo nè dalla precedente locazione acceduta. (RAM, ROM,...)
- **Associativo**:
  - per accedere ad un dato bisogna associare l'indirizzo al contenuto, ossia parte dell'indirizzo o una funzione dell'indirizzo (chiave hash) è contenuta nella memoria e bisogna accedere a tutte le locazioni presenti per verificare la presenza del dato.

Memorie (4/4)

5. Prestazioni:

- **Tempo di accesso** (**Tacc**): il tempo necessario per una operazione di lettura (o scrittura). Per le memorie ad accesso casuale è il tempo che intercorre da quanto vengono forniti gli indirizzi a quando sono pronti i dati. Per le memorie sequenziali comprende anche il tempo per raggiungere i dati (e quindi dall'indirizzo stesso del dato).

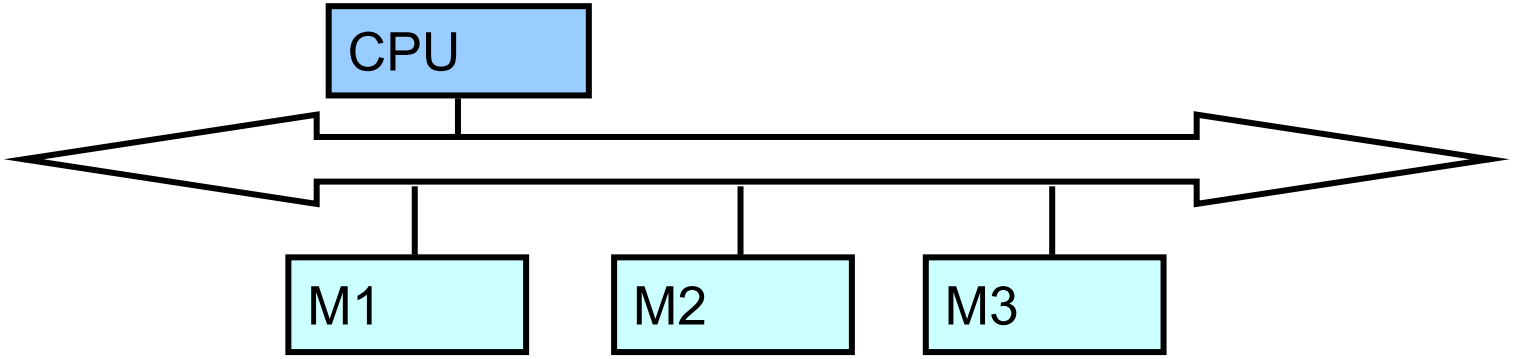
- **Tempo di ciclo**: è il tempo di accesso più il tempo per terminare l'operazione prima di poter compiere un altro accesso. Di solito si calcola in lettura (**Trc**)

- **Velocità di trasferimento** (**bit rate**): per le memorie ad accesso casuale è il reciproco del tempo di accesso, considerando il numero di bit trasferiti

- Di solito sono dati che vengono imposti dalla CPU a cui la memoria o si adegua o introduce ritardi.

Organizzazione delle memorie

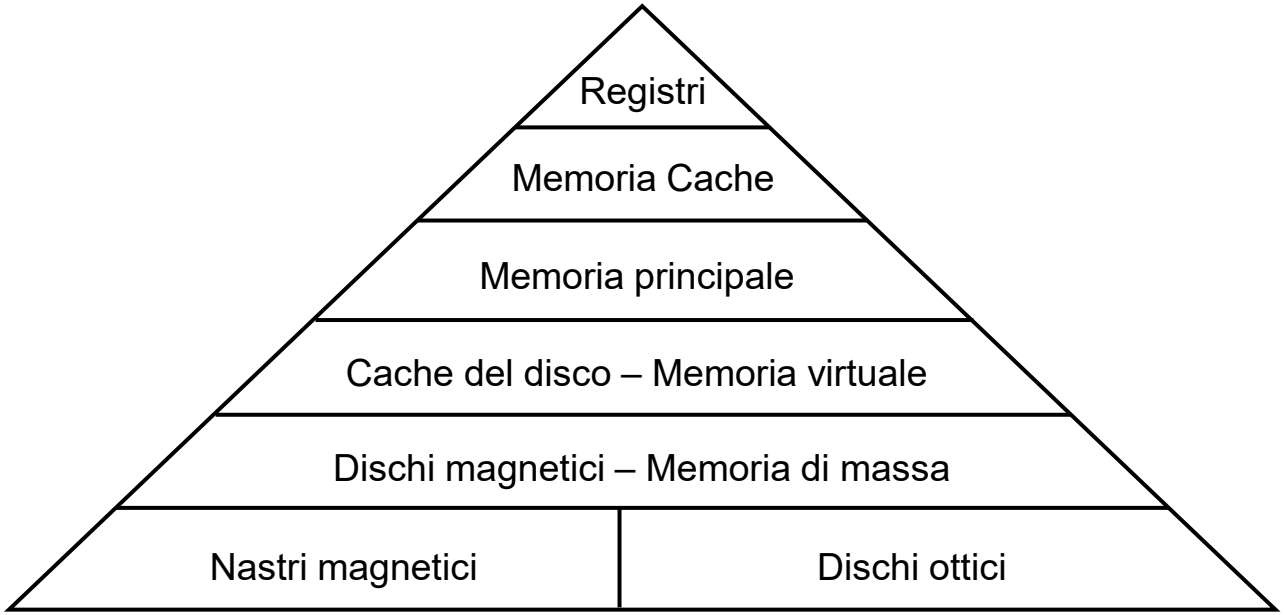
- *La memoria ideale dovrebbe essere a capacità infinita, tempo di accesso nullo, e costo e consumo nullo*.
- In pratica, l'organizzazione del calcolatore comprende diversi tipi di memorie con caratteristiche diverse
- Alcune memorie sono dedicate, ossia sono progettate per contenere specifici tipi di dati (es: la EPROM o la flash di bootstrap)
- Altre memorie sono general-purpose e possono essere accedute mediante specifici metodi di indirizzamento per leggere o scrivere dati di qualsiasi tipo. Il loro impiego dipende dalle loro caratteristiche e da come il dato viene impiegato durante il funzionamento del calcolatore



Gerarchia di memoria

- **Gerarchia di memorie:** la memoria del calcolatore è organizzata gerarchicamente in modo da comprendere poche memorie a bassa capacità ed alti costi, ma molto veloci, e molta memoria più lenta e di capacità maggiore.

- L'obiettivo è di organizzare le politiche di piazzamento ed accesso dei dati in modo tale da avere i dati più frequentemente usati, o in generale più "utili" virtualmente sempre nelle memorie più veloci.



Memoria

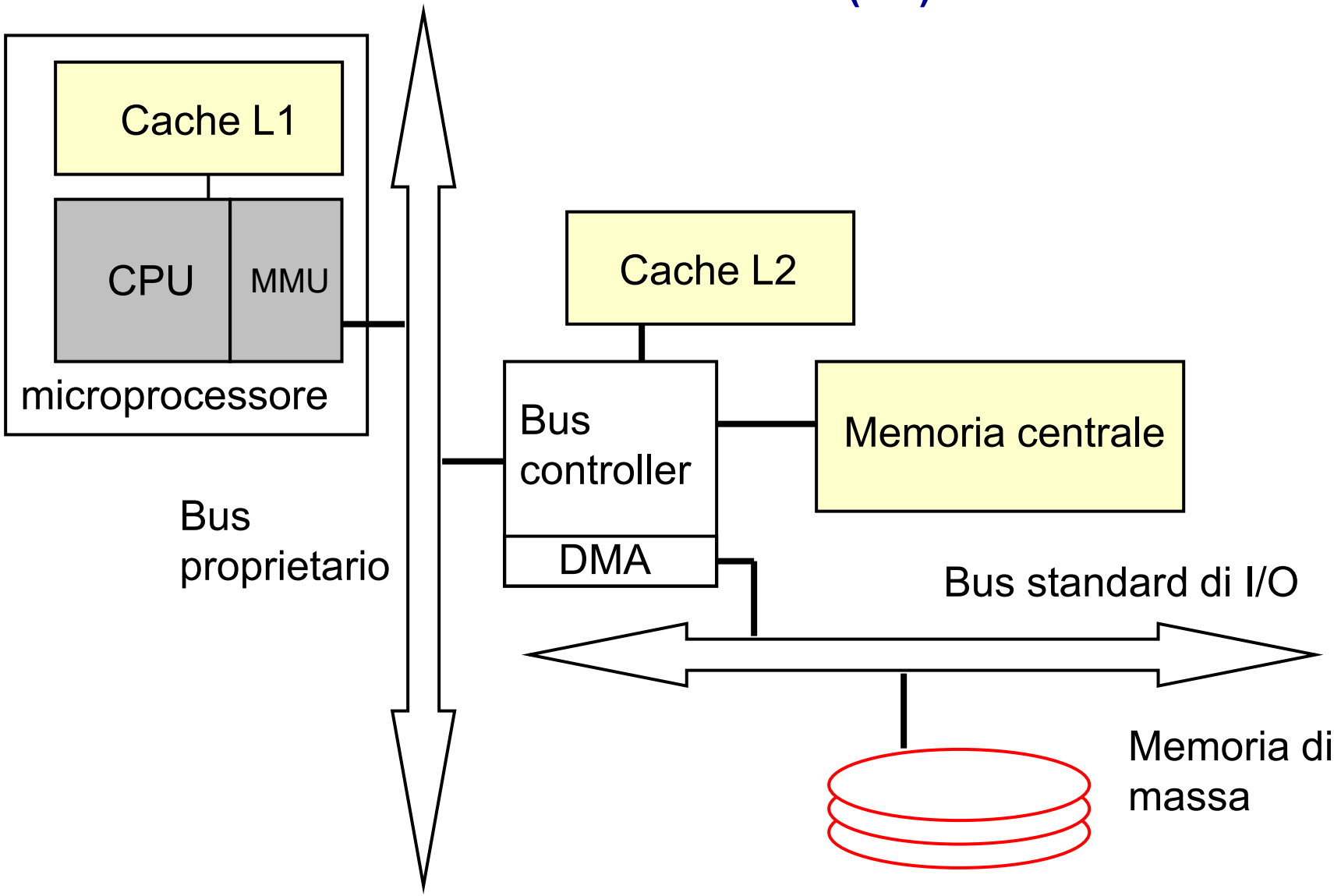
- La **memoria interna** alla CPU è costituita dai registri ed è caratterizzata da: alta velocità (comparabile con quella del processore) e limitate dimensioni (al più qualche migliaio di byte).

- La **memoria centrale o principale** è caratterizzata da dimensioni molto maggiori della memoria interna alla CPU (fino a qualche Gbyte), ma tempi di accesso più elevati. È accessibile in modo diretto tramite indirizzi.
- Nei sistemi attuali almeno un livello di memorie **cache** è stato inserito tra CPU e memorie centrali

- **Memorie secondarie**:

  In un calcolatore esistono diverse memorie secondarie (o memorie di massa) ad alta capacità, bassi costi e non volatili

Gerarchia di memorie (1/2)



Gerarchia di memorie (2/2)

Regole base dell'efficienza è rendere di massima velocità il caso comune: sfruttamento della località spaziale e temporale

Principio di località: un programma in ogni istante utilizza una porzione limitata dello spazio di indirizzamento:

- **Località spaziale**: accedendo ad un dato è assai probabile che si debba accedere ad altri dati localizzati "vicino" nello spazio di indirizzamento (es: array e tabelle)
- **Località temporale:** accedendo ad un dato è assai probabile che si debba riaccedere ad esso in un tempo "vicino" (es: cicli)

- Dati che inizialmente si trovano ad un livello più basso
  - conviene spostarli in memorie più veloci (loc. temporale)
  - conviene spostare anche i dati vicini (loc. spaziale)
- I programmi NON vedono la gerarchia, ma referenziano i dati come se fossero sempre in memoria centrale (a parte per i registri che sono nominati esplicitamente) perciò bisogna mantenere il più vicino possibile alla CPU dati utilizzati più recentemente
- La gerarchia delle memorie deve prevedere in successione memorie sempre più larghe e più lente per mantenere i dati nei livelli più alti proporzionatamente alla previsione della frequenza d'uso

Blocchi di locazione di memoria

Gerarchia di memoria: definizioni

- **Blocco**: unità di informazione minima scambiata fra livelli (es: 32 bytes)
- **Hit**: un accesso alla memoria che trova l'informazione cercata nel livello superiore
- **Hit Rate**: frequenza di accessi trovati nel livello superiore (h)
- **Miss** Rate: (1 - h)
- **Hit Time**: tempo di accesso al livello superiore Th
- **Miss Penalty**: tempo per rimpiazzare un blocco nel livello superiore più il tempo per fornirlo alla CPU. Il "miss penalty" può anche essere scomposto nel tempo per reperire il primo dato di un blocco più il tempo per trasferire gli altri.

- $Tacc = h\,Th + (1-h)Tmp = h\,Th + (1-h)\,(Th + Tmiss)$

- $Tacc = Th + (1-h)\,Tmiss$

Gerarchia: esempio

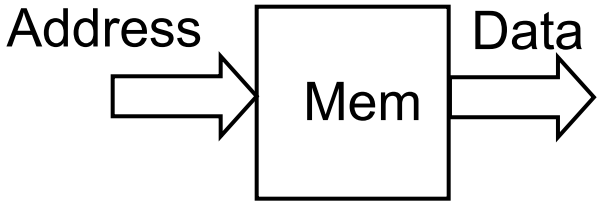
- **Esempio**: Supponiamo un sistema a due livelli in cui per accedere al primo livello (memoria centrale) il tempo di accesso sia 0.1 microsec per ogni dato e per accedere al secondo(ad esempio, l'hard disk) sia di 0.1 msec.
- Per accedere al secondo livello il dato deve essere copiato nel primo. Ipotizziamo che le probabilità di trovare il dato al primo livello siano del 95%.
- Trascuriamo il tempo necessario al processore per capire dove si trova il dato.

Quanto è il tempo di accesso medio?

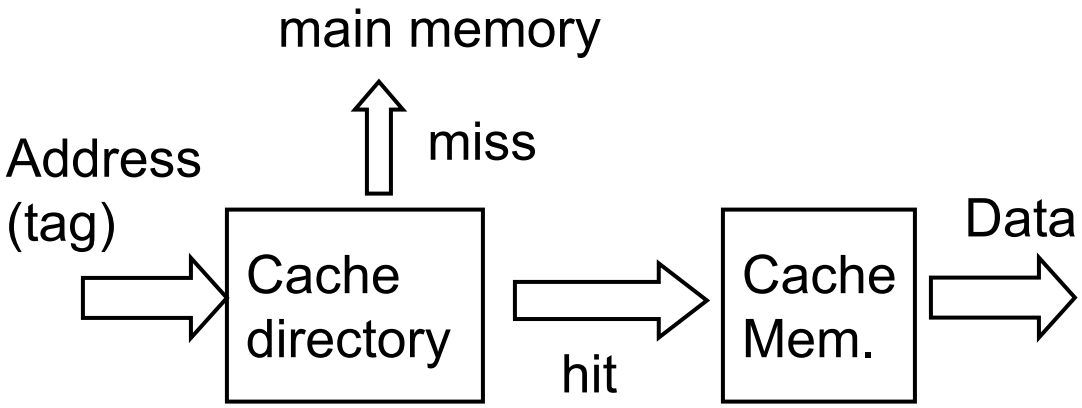
In microsec: Tacc= 0.95 0.1 + 0.05 (0.1 +100)= 0.095+5.5= 5.596 microsec

Basta un 5% di penalizzazione perchè tutto l'accesso medio sia fortemente rallentato

Memoria centrale e cache



In memoria centrale il dato è individuato dall'indirizzo:
Tacc costante (es: Tacc = 80 ns)



In cache il dato è individuato da parte dell'indirizzo:
- se hit, Tacc=Thit (es: Thit=5 ns)
- se miss, Tacc=Tmiss

Gerarchia di memorie

Parametri di progettazione di una gerarchia di memoria:

- Quanti livelli ha la gerarchia
- Che dimensione e velocità per ogni livello
- Cosa metto nei vari livelli

Ogni tipo di memoria si definisce in base al

1) **Piazzamento** del blocco (o funzione di traduzione o **mapping**): dove può essere allocato il blocco al livello corrente
2) **Identificazione** del blocco: come si può ritrovare il blocco a livello corrente
3) **Rimpiazzamento** del blocco: come si sceglie quale blocco sostituire
4) Strategia di **scrittura**

Ad esempio **a livello di registri**:

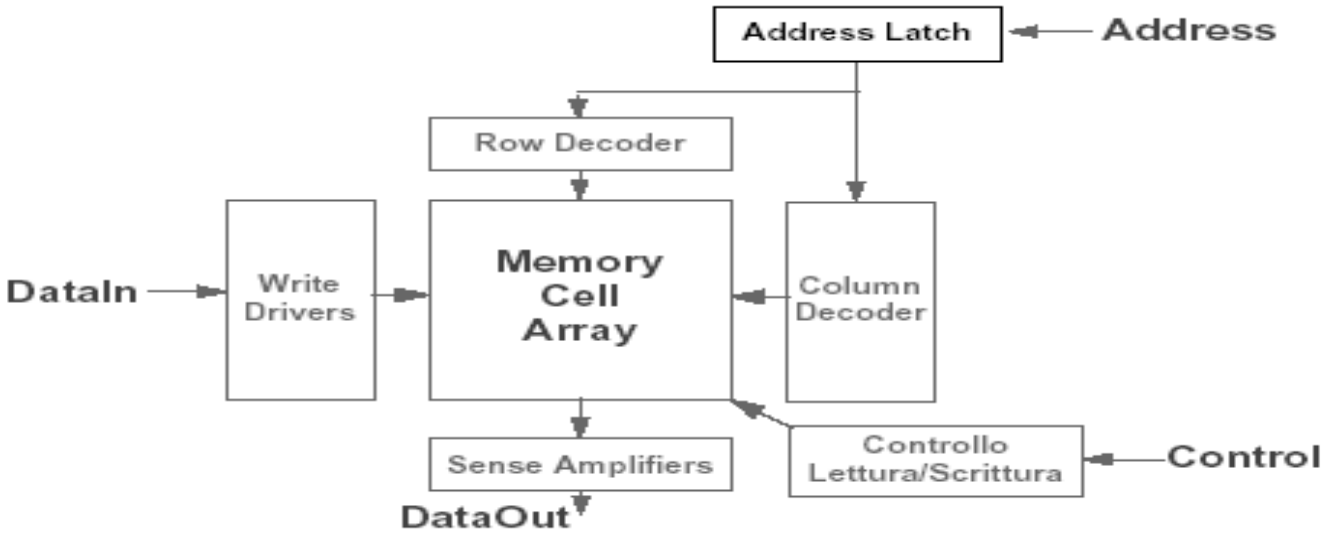
- **l'identificazione** è nominale (mov ax,15)
- **il piazzamento e rimpiazzamento** è definito dal compilatore

Memorie nel calcolatore

Architettura delle memorie a semiconduttore.

Memoria centrale:

1) **Piazzamento** del blocco: deciso dall'indirizzo nell'istruzione
2) **Identificazione** del blocco: indirizzo (codice o dal sistema operativo)
3) **Rimpiazzamento** del blocco: deciso dal codice
4) Strategia di **scrittura:** deciso dal codice



- DataIn e DataOut sono poi multiplexati su uno stesso bus selezionato dal segnale di lettura/scrittura

Memorie RAM

- **RAM** (**Random Access Memory)**: memorie volatili a lettura e scrittura in cui il tempo di accesso è costante per ogni locazione di memoria indirizzata.

- **SRAM (Static RAM)**: sono memorie molto veloci realizzate in diverse tecnologie che permettono al dato di rimanere staticamente memorizzato fino a che il dispositivo è alimentato. Sono composte da
  - tanti FF quanti i bit da memorizzare
  - logica combinatoria di selezione del FF (o dei FF se la parola ha parallelismo maggiore di 1 bit) a partire dagli indirizzi
  - logica combinatoria di selezione di lettura e scrittura

- **DRAM (Dynamic RAM)**: non usano FF, ma il bit è memorizzato in un condensatore che può essere caricato o scaricato; però necessita di essere rinfrescato (**refresh**) ogni decina di msec e quindi è necessaria una logica aggiuntiva di refresh; più lente, ma a più alta capacità delle RAM statiche.

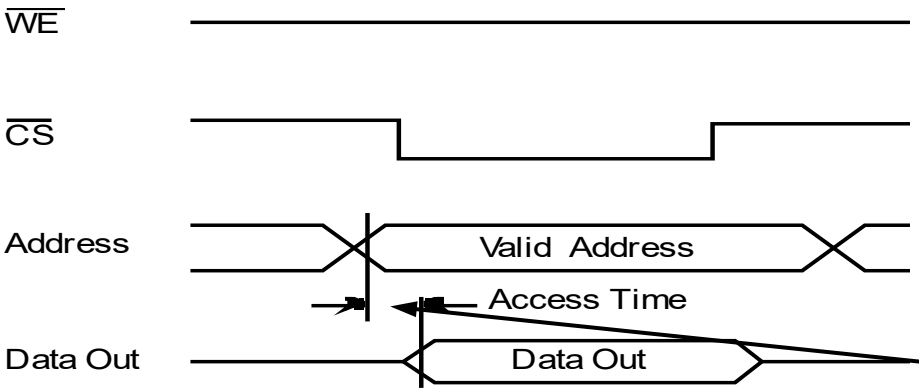
Memorie ROM

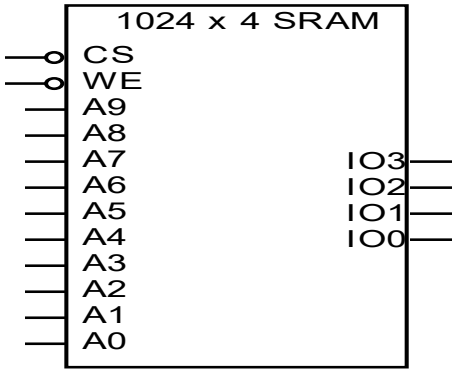
- **ROM** (**Read Only Memory)**: memorie di sola lettura; non hanno la logica per la scrittura, ma la scrittura avviene con altra tecnologia
  - PROM: programmabili una volta impiegando un pin speciale (ad elevato voltaggio)
  - EPROM: cancellabili in toto con luce ultravioletta
  - EEPROM: sono cancellabili elettricamente senza doverle estrarre dalla scheda, più lente e con meno capacita'
  - FLASH: leggibili e scrivibili a blocchi senza estrarle dal circuito con alta capacità, ma cancellabili un n. limitato di volte

Type	Category	Erasure	Byte alterable	Volatile	Typical use
SRAM	Read/write	Electrical	Yes	Yes	Level 2 cache
DRAM	Read/write	Electrical	Yes	Yes	Main memory
ROM	Read-only	Not possible	No	No	Large volume appliances
PROM	Read-only	Not possible	No	No	Small volume equipment
EPROM	Read-mostly	UV light	No	No	Device prototyping
EEPROM	Read-mostly	Electrical	Yes	No	Device prototyping
Flash	Read/write	Electrical	No	No	Film for digital camera

SRAM

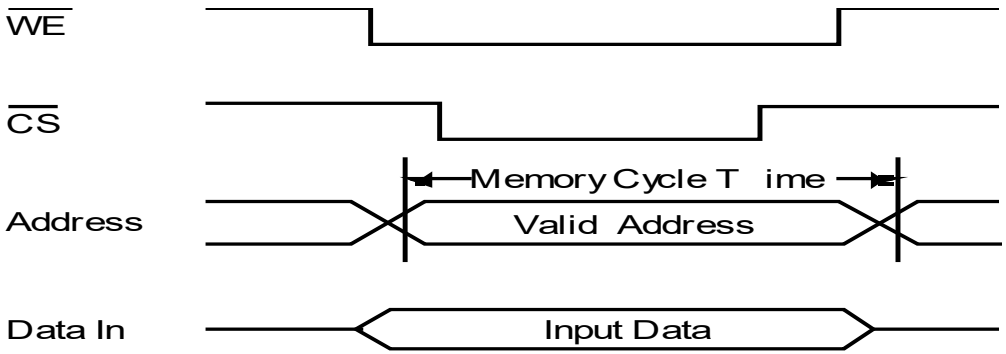
- Esempio di RAM statica 1024 x 4 bit
- in CS (attivo basso) seleziona il dispositivo e WE indica se si ha una lettura o scrittura
- **ciclo di lettura**

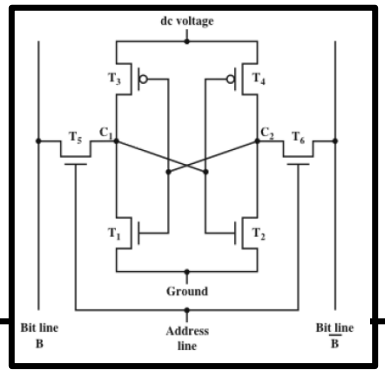




Il tempo di accesso si calcola da quando i segnali di controllo ed indirizzi sono validi a quando i dati sono forniti in uscita dalla memoria

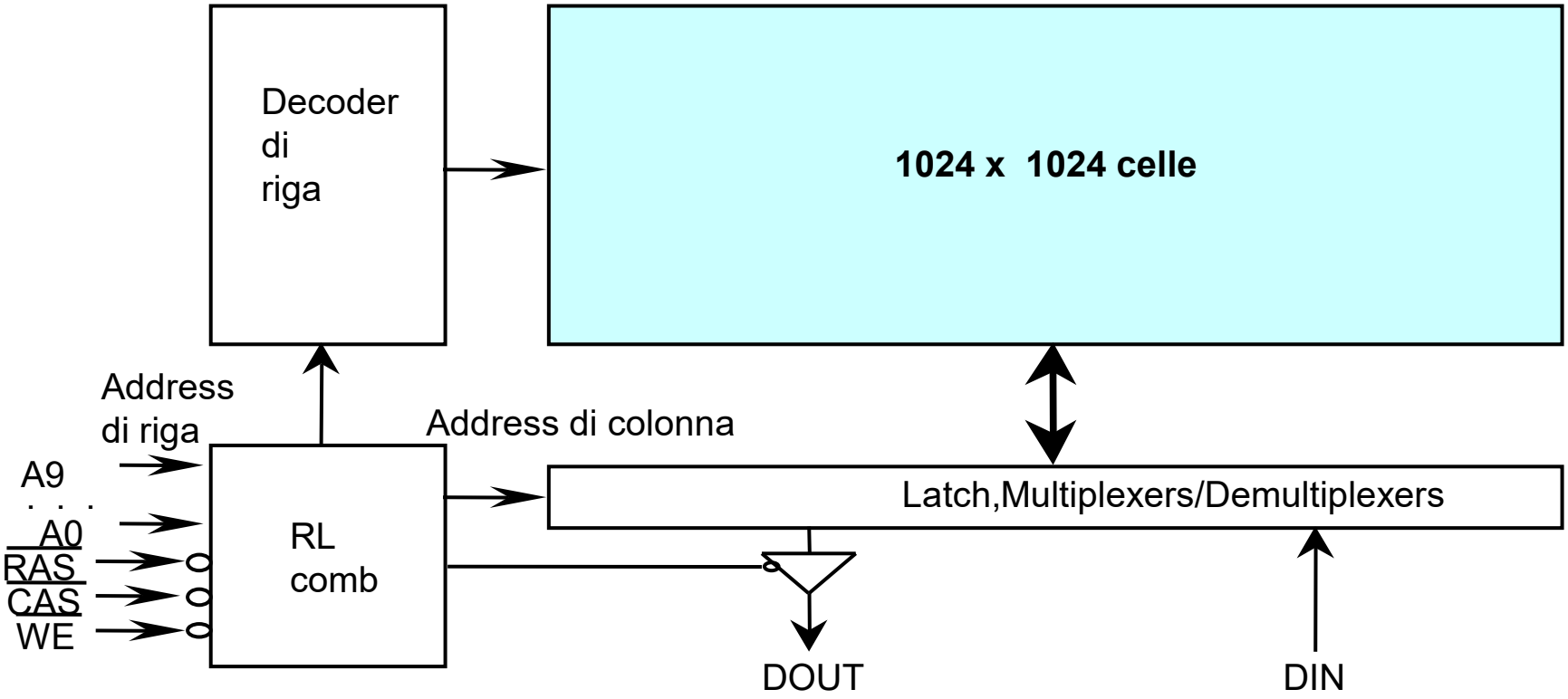
- **ciclo di scrittura**





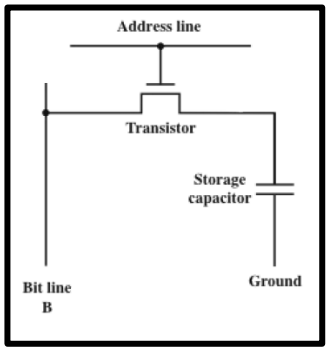
RAM dinamiche (1/3)

- Ad elevata capacità (ad esempio un 1 Mbit) con bus multiplexato
- Il tempo di accesso deve tener conto sia del segnale di RAS (Row Address Strobe) che di CAS (Column Address Strobe)
- Occorre poi considerare il tempo del rinfresco ad esempio ogni 16ms devono essere lette e scritte tutte le 1024 righe
- E servono 1024 cicli di RAS



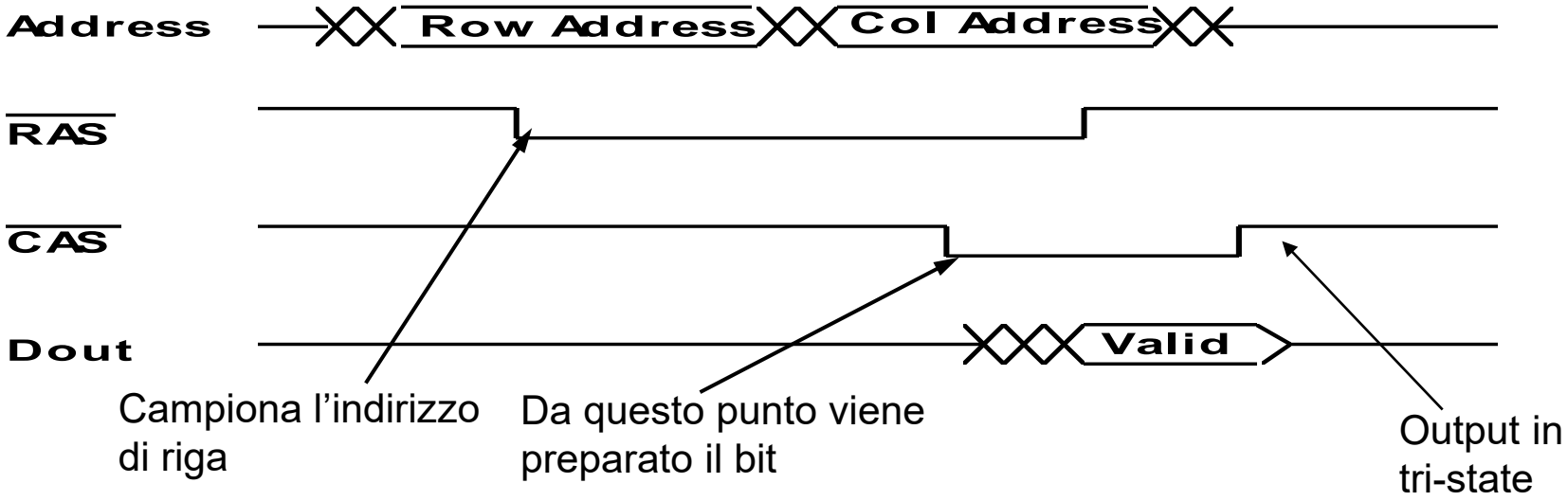
RAM dinamiche (2/3)

- DRAM da 16Mb 4M locazioni per parole di 4 bit

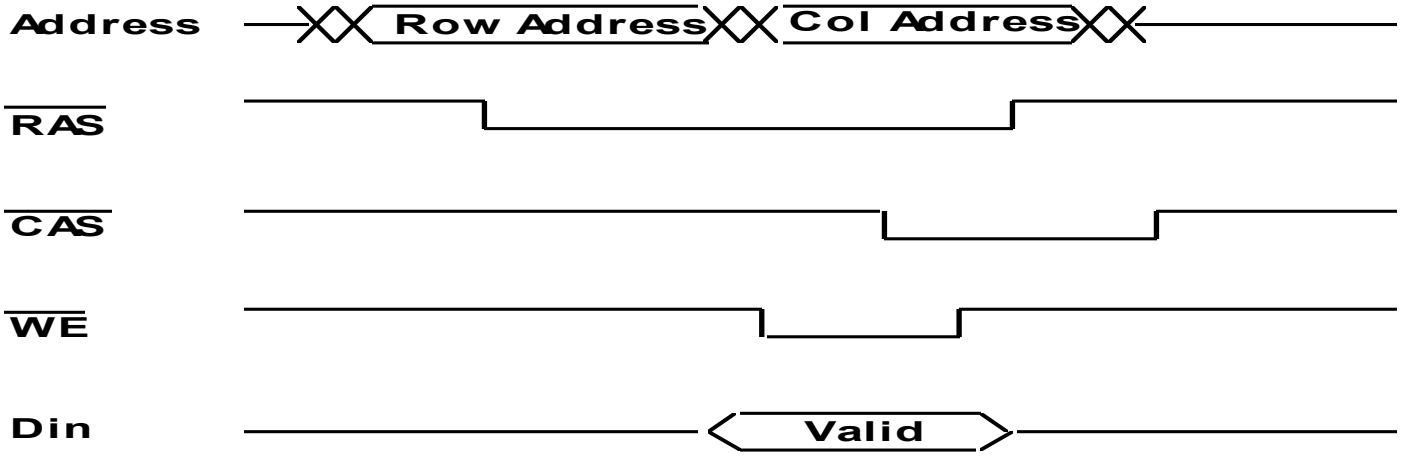


RAM dinamiche (3/3)

- **Temporizzazione in lettura**



- **Temporizzazione in scrittura**



RAM sincrone

- Le DRAM sono il collo di bottiglia dei calcolatori moderni. Per questo sono state ideate soluzioni alternative:

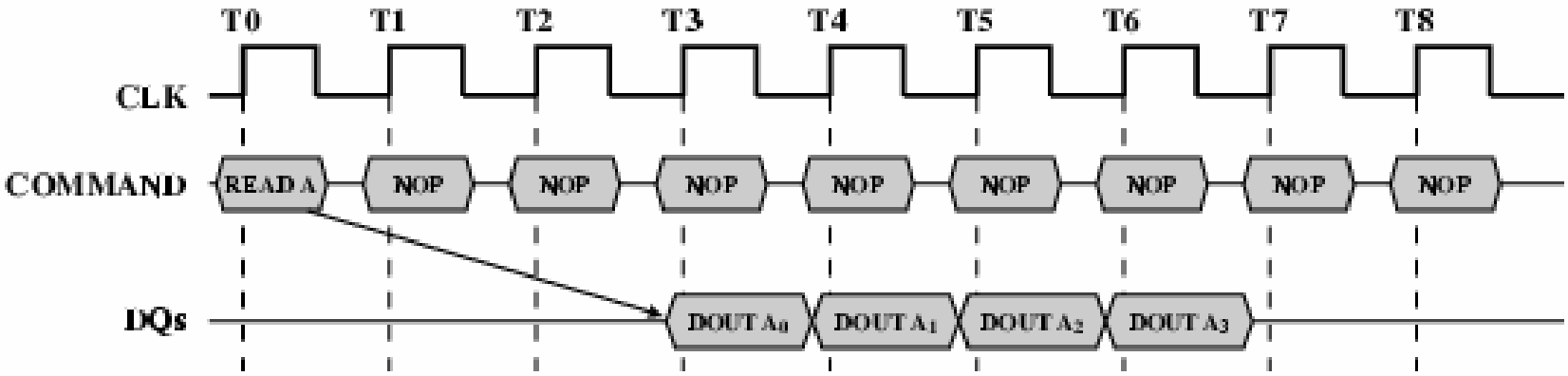
	Clock Frequency (MHz)	Transfer Rate (GB/s)	Access Time (ns)	Pin Count
SDRAM	166	1.3	18	168
DDR	200	3.2	12.5	184
RDRAM	600	4.8	12	162

- RAM sincrone:
  - hanno un ritardo (Tacc) sincrono con un clock dato
  - SDRAM Syncronous Dynamic RAM
    - Sono generalmente più lente delle corrispondenti memorie asincrone
    - hanno un ritardo predicibile (n Tck) in cui la CPU può attendere (con segnale di rdy#) o far partire un'altra attività
- CPU manda indirizzo
- Dopo un ritardo n Tck la memoria è pronta a rilasciare una o più parole
- Se la memoria può rilasciare una sequenza di parole può essere usata in trasferimenti a burst

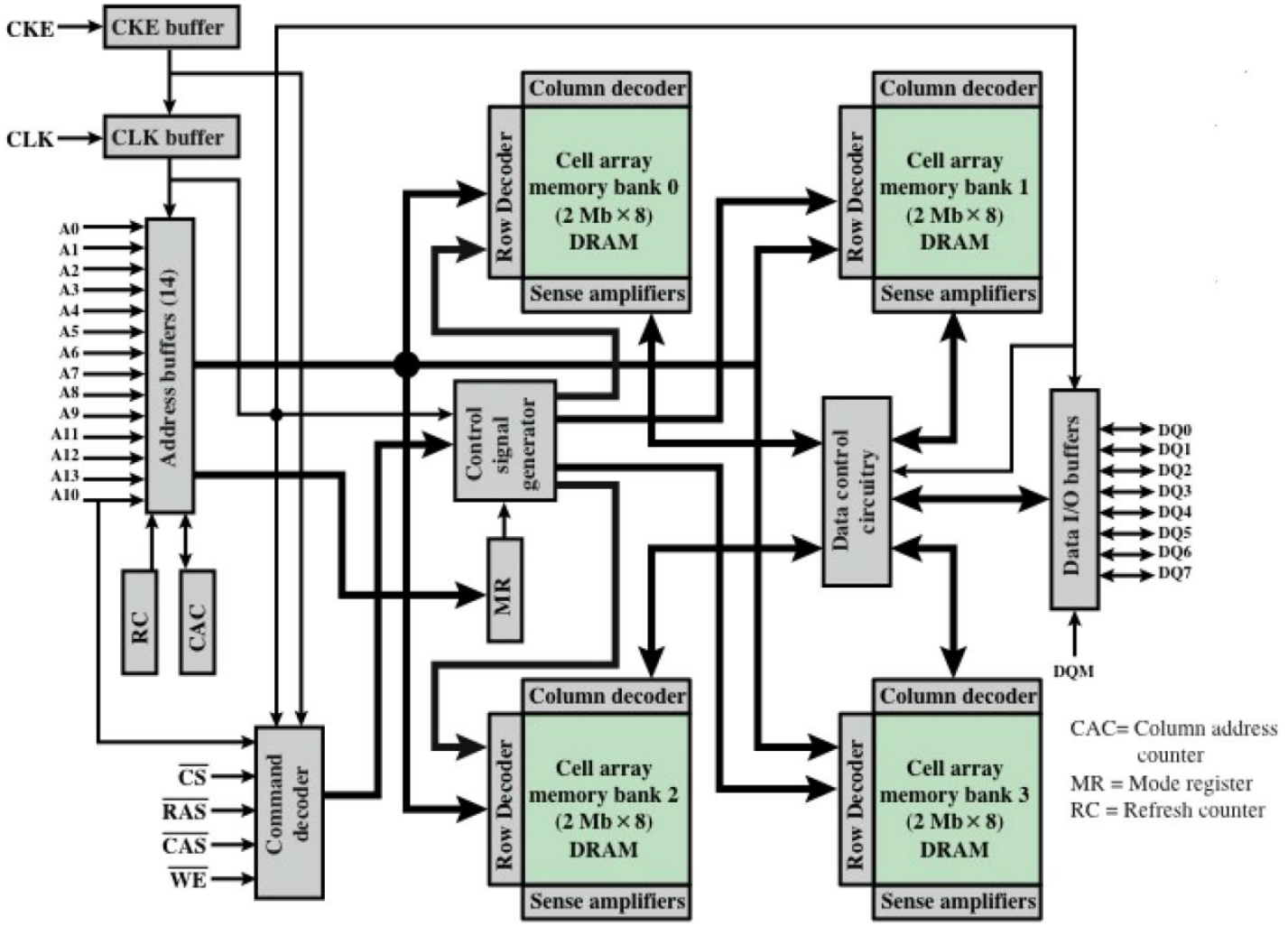
Trasferimento a burst con SDRAM

- Le memorie devono adeguarsi ai tempi richiesti dalla CPU
- La CPU definisce quanti cicli di clock sono necessari in un ciclo di bus per
  - **Trasferimento singolo** (1 sola parola larga al massimo come il data bus)
  - **Trasferimento a BURST:** La CPU ha una modalità (con un segnale di controllo) per gestire il trasferimento a burst. Utile se deve leggere o scrivere più parole di seguito, ad esempio per riempire la cache interna

- Esempio di memoria sincrona con 2 cicli di ritardo dopo il primo indirizzo: ciclo 4-1-1-1



Architettura SDRAM



Esempio Bus Pentium

- Il trasferimento di un dato avviene in 2 cicli di clock: uno per gli indirizzi ed uno per i dati + i cicli di wait.
- **Trasferimento a burst** vengono trasferiti più dati ad indirizzi sequenziali

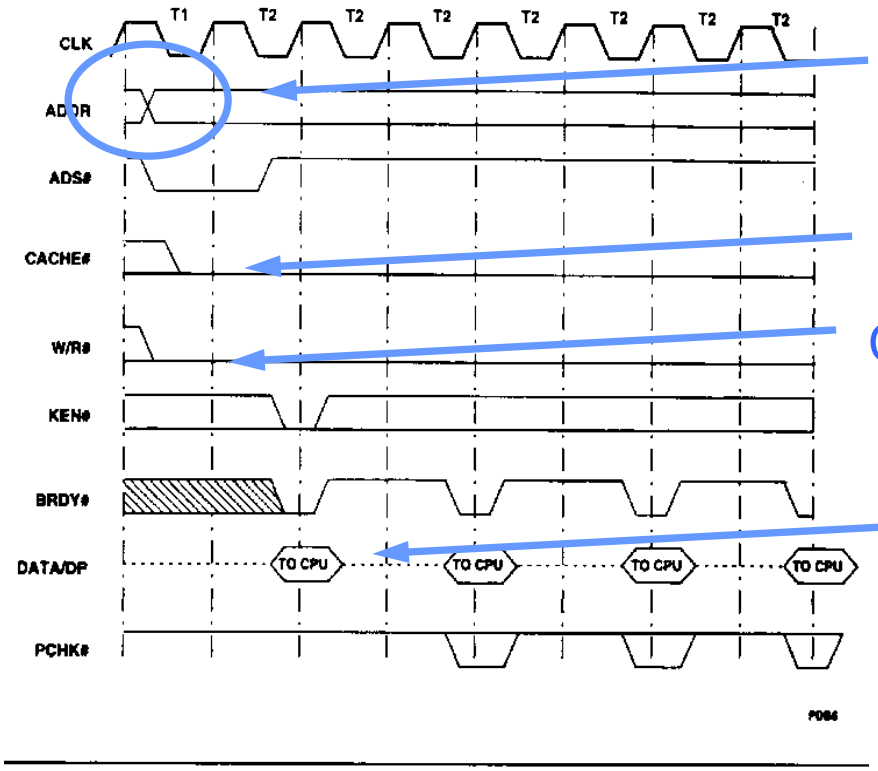


Figure 6-9. Slow Burst Read Cycle

La CPU manda l'indirizzo

Trasferimento a burst

Ciclo di lettura

Trasferimento di 4 parole da 64 bit

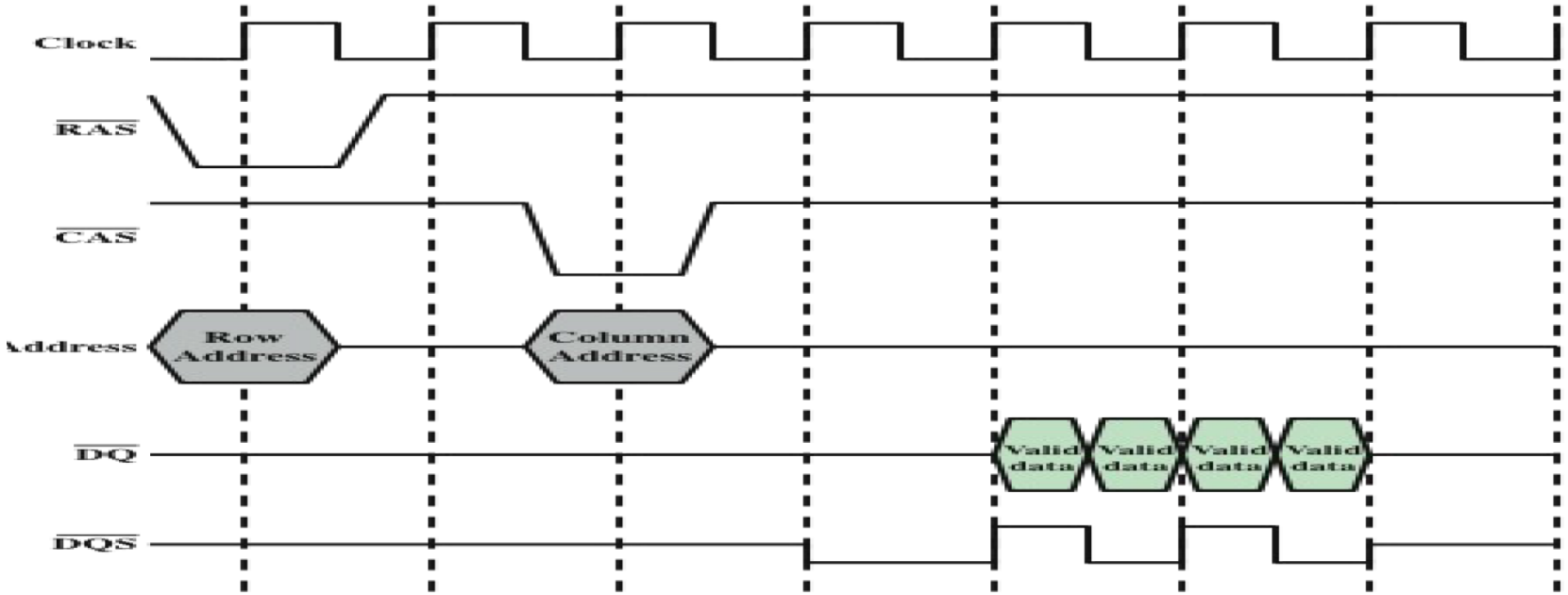
Nuove tecnologie di memorie

- Memorie DRAM di nuova generazione: nate per supportare in modo efficiente i trasferimenti a burst:
  - **FPM-DRAM (Fast Page Mode):** permettono di tenere il segnale di RAS# attivo e variare solo CAS# in modo da rendere più lento solo il primo trasferimento. Permettono trasferimenti asincroni tramite appositi segnali di sincronizzazione aggiuntivi. Tipicamente i trasferimenti possibili sono 6-3-3-3 o 5-3-3-3.

    Ad esempio consideriamo un clock a 66Mhz (di bus) (Tck=15ns) e DRAM a tempo di accesso di 70ns. Il primo dato necessita di 4 clock di wait (4+1)x15=75. Poi però non avendo bisogno di cambiare il RAS è possibile aspettare meno in 5-3-3-3.
  - **EDO-DRAM (extended data out).** Permette di risparmiare sui trasferimenti successivi perchè l'uscita non viene disabilitata ad ogni lettura. Tipicamente i tempi di burst sono 5-2-2-2.
  - **SDRAM (synchronous DRAM)** sono memorie con una logica sequenziale e che richiedono un clock. RAM ibrida (parte statica, parte dinamica). Non necessita di segnali di controllo (CAS, RAS, …). Molto simili alle EDO ma con una logica per accedere in modo veloce a dati consecutivi fornendo gli indirizzi consecutivi (senza controller esterno). Tipicamente 5-1-1-1 a 100Mhz.
  - **Double data rate (DDR) DRAM**: ulteriore miglioramento della SDRAM. Tutti i tipi di SDRAM usano come segnale di clock un'onda quadra. La tradizionale SDRAM si attiva sul fronte crescente del segnale ed ignora quello opposto. La DDR opera in entrambi gli istanti, a volte raddoppiando la quantità di dati trasferiti nello stesso tempo.Lo standard DDR si sta evolvendo da DDR (2,5 V) a DDR2 (1,8 V) a DDR3 (tra 1,2 e 1,5 V).
  - **RDRAM (rambus DRAM)** e **DRDRAM (direct Rambus DRAM)** memorie asincrone con controllore integrato.
    Costruite da specifiche di un consorzio Rambus che assieme a Intel fornisce un controllore dedicato di memorie integrato nel chipset per premettere trasferimenti elevatissimi per processori sopra il GHz. Permette fino a 1 GB/s di trasferimento tanto da sostituire le cache in applicazioni speciali (es: video controller)

Double data rate (DDR) DRAM

- Double Data Rate SDRAM
  - JEDEC Solid State Technology Association
- Le SDRAM "standard" spediscono un blocco per ciclo
- Le DDR SDRAM sfruttano sia fronte salita che discesa

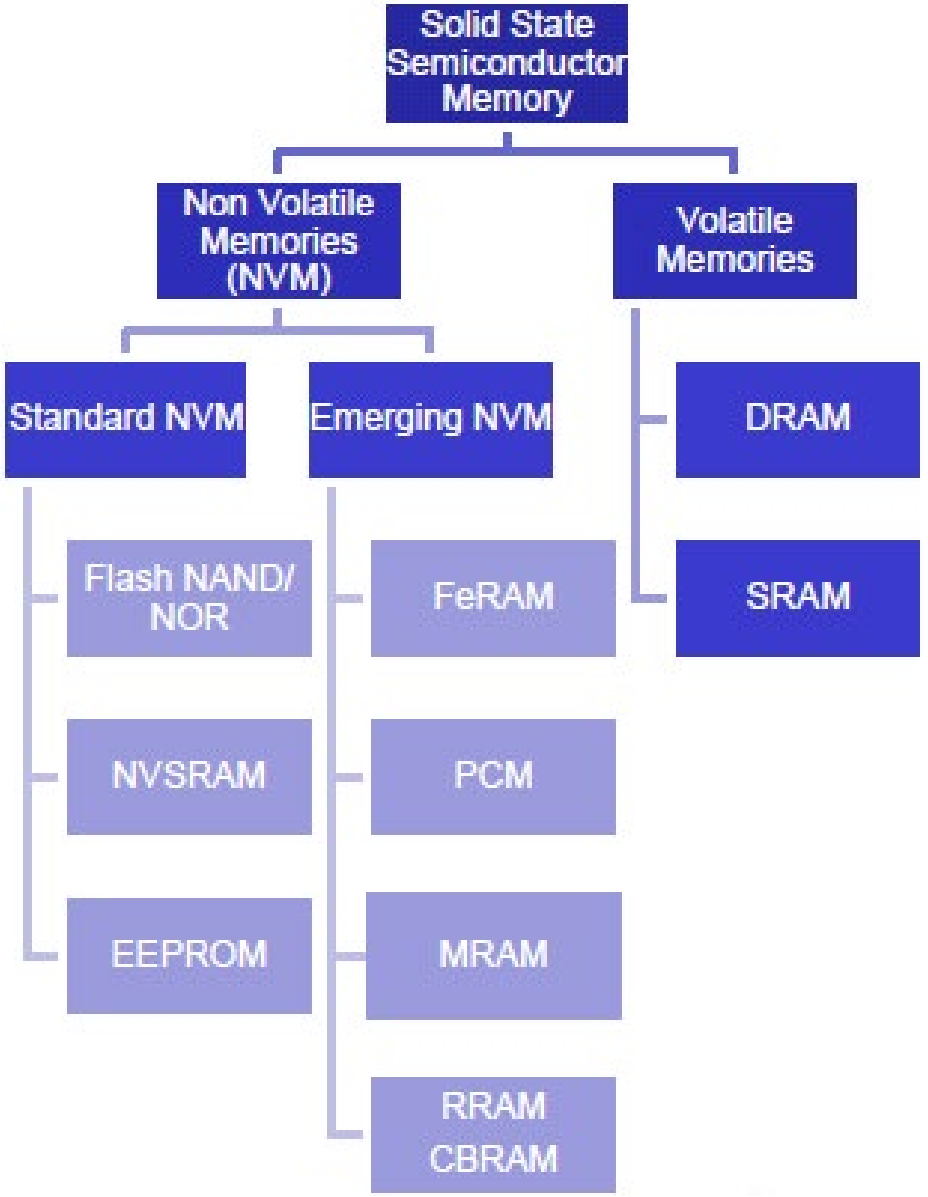


Rambus DRAM

- Rambus DRAM
  - Tecnologia Intel
- Bus specifico → RAMBUS
  - Fino a 320 chip RDRAM
  - 1,6 Gbyte/s
  - Protocollo orientato ai blocchi
  - Chip con pin da un solo lato
  - Limiti lunghezza fisica bus

Memorie permanenti (1/6)

- Per i nostri scopi ci concentriamo sulle memorie permanenti o **memorie non volatili (NVM)**
- Le tecnologie "standard" e già molto diffuse includono:
  - **Memorie FLASH** (in tecnologia NAND o NOR): utilizzate per chiavette USB, ma anche per **dischi SSD (Solid State Disk)** con tecnologia NAND
  - **NVSRAM (Non Volatile Static Random Access Memory)**: come SRAM ma non volatili
  - **EEPROM** viste in precedenza



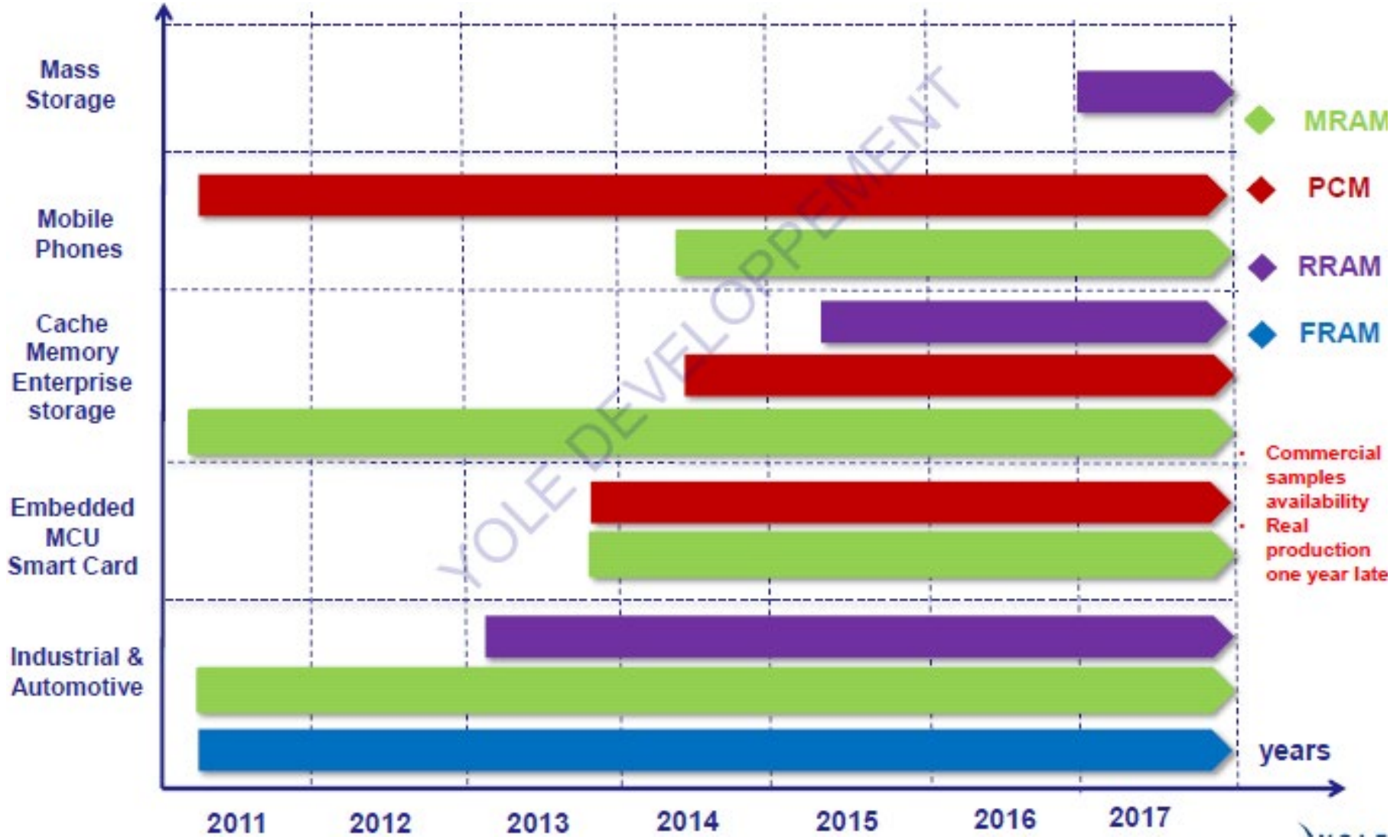
Memorie permanenti (2/6)

- Esistono varie nuove tecnologie di NVM tra cui:
  - **FeRAM** o **Ferroelectric RAM (**o **FRAM** oppure **F-RAM)** è simile nel processo costruttivo alla memoria DRAM, ma usa uno strato di materiale ferroelettrico per ottenere la proprietà di non-volatilità. Malgrado il mercato delle memorie non volatili sia attualmente dominato dalla tecnologia Flash, la FeRAM offre numerosi vantaggi rispetto alla Flash: minor consumo, velocità di scrittura più elevata e un numero di cicli di scrittura/cancellazione maggiore (oltre i 10^{16} per i dispositivi da 3,3 V).
  - **PCM (Phase-change memory** o **memoria a cambiamento di fase**) è una NVM a stato solido di nuova generazione che è composta da un materiale in grado di cambiare fase (cristallina o amorfa) in modo reversibile e controllato per mezzo di una corrente di programmazione. Il concetto è simile a quello dei CD/DVD. In particolare una regione amorfa presenta bassa riflettività (0 logico), mentre una regione cristallina presenta alta riflettività (1 logico).

Memorie permanenti (3/6)

- **MRAM (Magnetoresistive RAM)** è una tipologia di memoria che sfrutta l'effetto magnetoresistivo. A differenza di una comune memoria RAM, le MRAM non memorizzano le informazioni come una quantità di carica elettrica ma come un campo magnetico. La lettura dell'informazione viene ottenuta tramite la misura della resistenza elettrica della cella. Normalmente se i due strati hanno la stessa polarità si considera il dato "0" mentre se le polarità sono opposte si considera lo strato "1".
- **RRAM/CBRAM (Conductive Bridging RAM)** è basata sulle proprietà di un elettrolita solido (generalmente solfuro di germanio drogato con rame) posto tra un elettrodo relativamente inerte (ad esempio tungsteno) e uno elettrochimicamente attivo (ad esempio argento o rame). In queste condizioni l'applicazione di un campo elettrico tra i due elettrodi provoca uno spostamento di ioni metallici nell'elettrolita con la conseguente formazione di “nano-fili” conduttivi. I principali vantaggi della tecnologia Cbram sono basso consumo, alta velocità di scrittura e lunga durata.

Memorie permanenti (4/6)

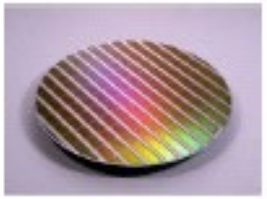


Memorie permanenti (5/6)

	Emerging memories				Established memories	
	FRAM	MRAM / STT MRAM	PCM	RRAM	DRAM	Flash NAND
Non volatile	YES	YES	YES	YES	NO	YES
Endurance (Nb cycles)	High (10^{12})	High (10^{15})	Medium (10^{8})	Medium (10^{8})	High (10^{15})	Low (10^{5})
2012 latest technological node produced (nm)	130 nm	130 nm	45 nm	R&D	30 nm	20 nm
Cell size (cell size in F^2)	Large (15-20)	Large /Medium (6-40)	Medium (6-12)	Medium (6-12)	Small (6-10)	Very small (4)
Write speed (ns)	Medium (100 ns)	High (10 ns)	Medium (75 ns)	Medium (75 ns)	High (10 ns)	Low (10 000 ns)
Power consumption	Low	High/low	Low	Low	Low	Very high
Cost ($/Gb)	High ($ 10 000 /Gb)	High ($ 1000 - 100 /Gb)	Medium (few $ /Gb)	R&D	Low ($ 1/Gb)	Very Low ($ 0,1/Gb)

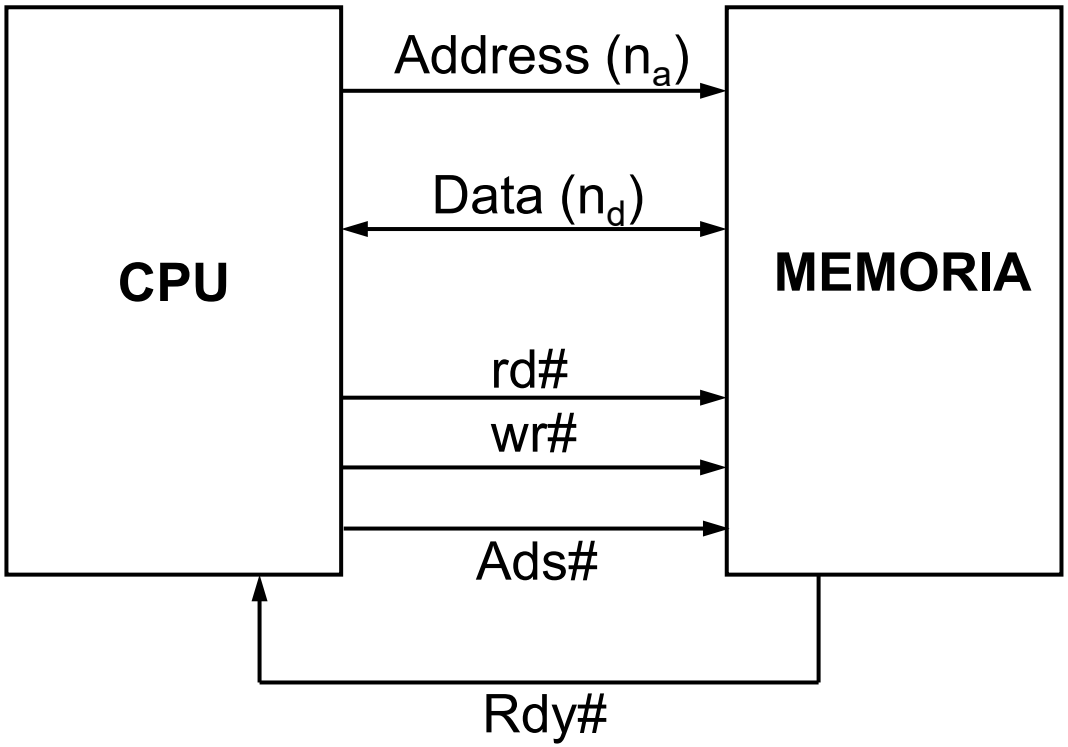
Memorie permanenti (6/6)

Emerging Non Volatile Memories applications

- **Industry & transportation**
  - 
    Industrial Automation
  - 
    Smart Meter
  - 
    Automotive
- **Embedded MCU for Smart Card**
  - 
    SIM Card
  - 
    ID Card
- **Cache Memory For Enterprise storage**
  - 
    Journal Memory
  - 
    Other Cache Memory applications
- **Mobiles Phones**
  - 
    Entry mobile phone
  - 
    Smart phones
- **Mass storage Memory**
  - 
    NAND Memories

Interfaccia generica tra CPU e Memoria

- Ads# (Address Strobe) indica quando inizia il ciclo (e sono pronti gli indirizzi), rd# e wr# indicano se lettura o scrittura, Rdy# indica se la memoria è pronta

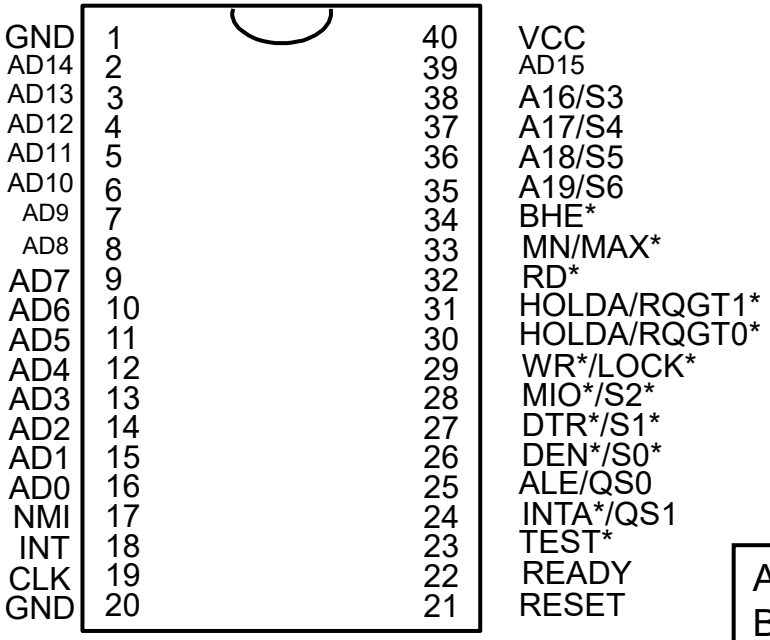


Bus di controllo

- Il bus di **controllo** contiene segnali non omogenei, in quanto ognuno ha la propria funzione e temporizzazione; sono diversi da casa costruttrice a casa costruttrice o a seconda dello standard; alcuni segnali sono comuni a tutti i bus
- Un segnale di **inizio della traslazione** (es: ADS# address strobe) che indica quando il ciclo ha inizio
- Un segnale di **lettura** (RD#) che indica quando i dati devono essere letti dalla CPU o in generale dal master
- Un segnale di **scrittura** (WR#) che indica quando i dati sul bus di dati sono pronti per essere scritti
- Un segnale di **destinazione** M/IO# che indica se la traslazione comprende oltre che la CPU anche la memoria o l'I/O. Tale segnale è previsto in quelle organizzazioni (come Intel ma non Motorola) che considerano uno spazio di indirizzamento separato tra memoria ed I/O.
- Uno o più segnali di sincronizzazione; nel caso di bus sincroni esiste un segnale di **pronto** (READY#) che indica che le periferiche o la memoria sono state sufficientemente veloci per completare il ciclo nei tempi fissati, ossia che il ciclo non deve essere ritardato ed un ciclo nuovo può iniziare.

Intel 8086: interfaccia

Interfaccia del processore Intel 8086

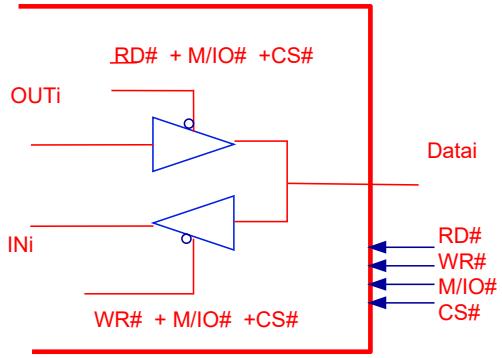


AD[0:15],A[16..19]	indirizzi e dati
BHE#	byte high enable
M/IO#	spazio di memoria o di I/O
RD#,WR#, DT/R#	scrittura e lettura
DEN#,ALE	data enable, address latch enable
HOLD/HOLDA	arbitraggio
READY	periferica pronta
INT,INTA#,NMI	gestione interruzioni

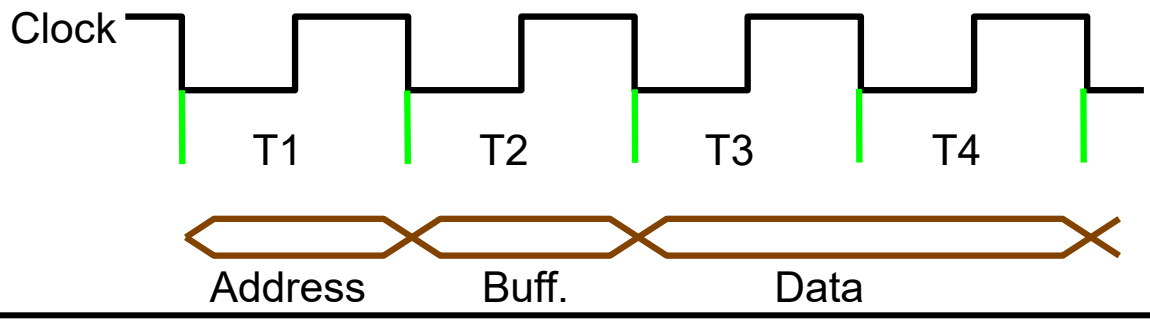
Processore con bus dati ed indirizzi multiplexato

Interfacce con i bus

- I dispositivi master emettono segnali sui bus. Sono collegati ad esso da porte che si chiamano **bus driver** come amplificatori con tri-state per poter essere disabilitati; i dispositivi che lavorano solo come slave hanno dei **bus receiver** per ricevere i dati se abilitati; se i segnali sono in entrambe le direzioni allora sono presenti dei **bus transceiver bidirezionali** e che possono essere disabilitati. Se abilitati i segnali passano sull'unico bus e si comportano logicamente come wired-or.



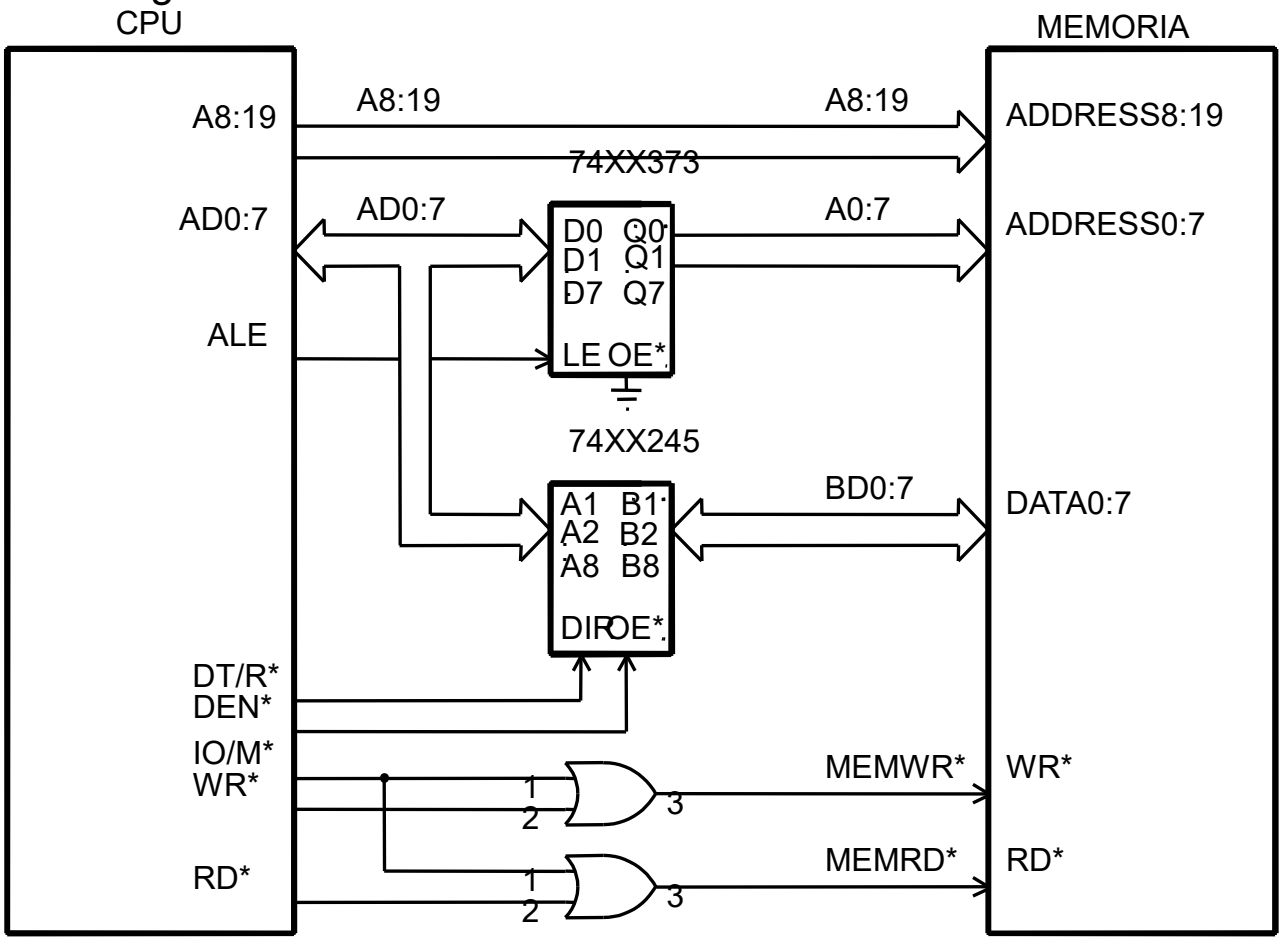
- alcuni bus possono essere **multiplexati** ossia coesistere sullo stesso supporto fisico (es, bus di indirizzi e dati). La loro funzionalità si attiva a multiplazione di tempo, ossia in alcuni fissati periodo del ciclo di bu si comportano da bus di indirizzi in altri casi da bus di dati.



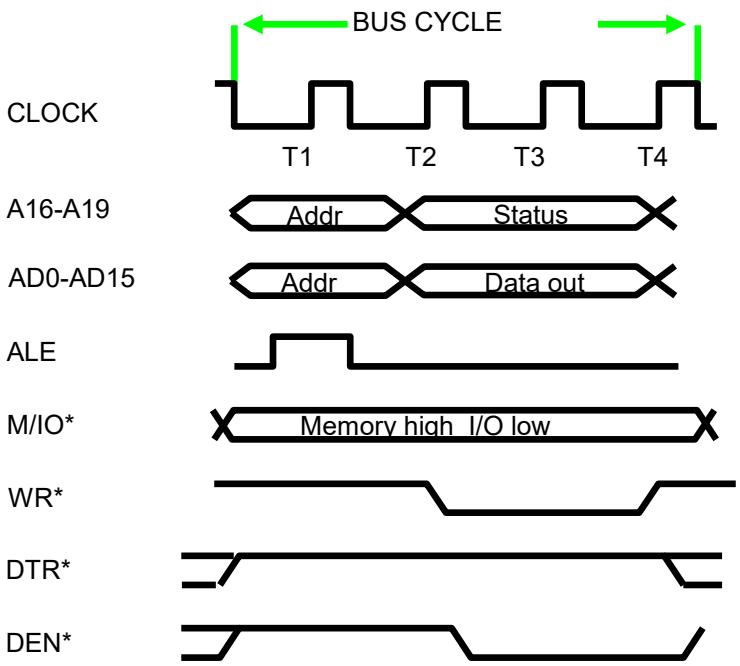
Interfaccia 8086

- Un registro di 8 latch campiona gli indirizzi quando sono pronti all'inizio del ciclo secondo il segnale ALE# e li mantiene per tutto il ciclo di bus; invece un transceiver esterno bidirezionale viene posto per bloccare i dati sul bus dei dati fissandone la direzione ed impedendone il passaggio nel periodo di tempo in cui sul bus multiplexato viaggiano i segnali d'indirizzo.

Nota: Caso 8088 e non 8086 per semplicità e chiarezza del disegno. Infatti, la differenza principale tra 8088 e 8086 è che l'8088 aveva un bus dei dati a 8 bit invece che a 16, quindi basta un 74XX373 (invece di 2) e un 74xx245 (invece di 2) per demultiplexare il bus.



Scrittura e lettura con 8086



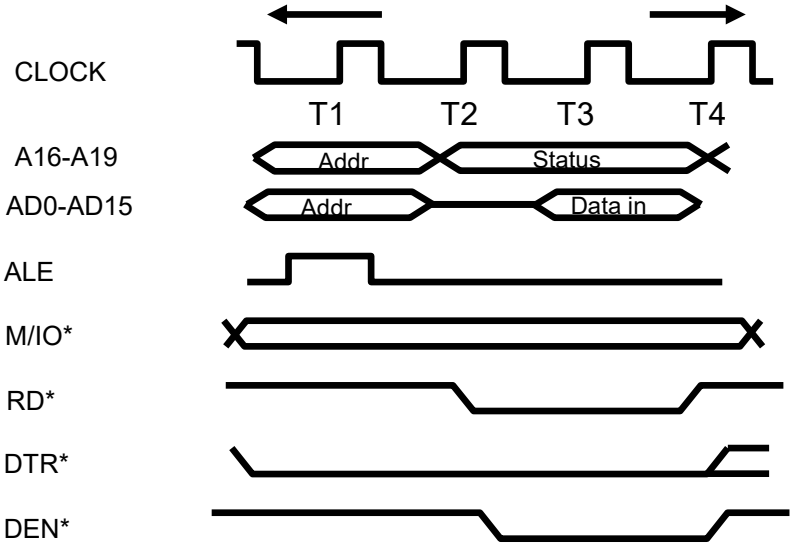
Scrittura

T1 ciclo di indirizzamento

T3 ciclo di trasferimento dati

T2 e T4 cicli idle

Lettura



Bus sincroni ed asincroni

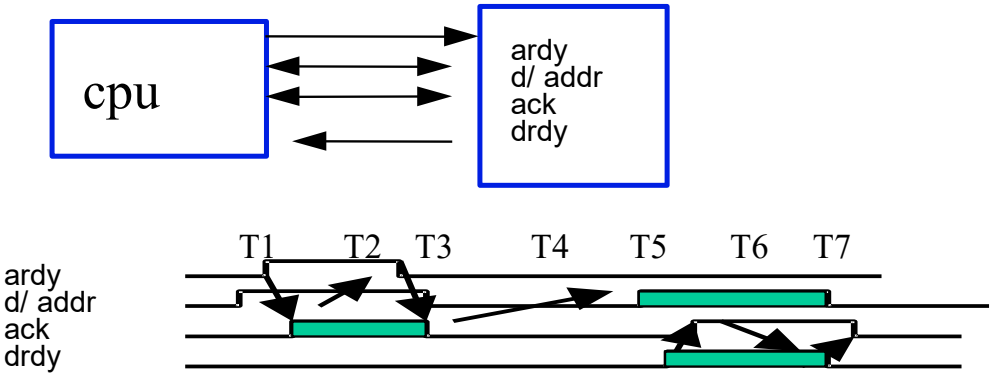
- I cicli di bus si definiscono **sincroni** od **asincroni** a seconda che il bus sia o meno controllato da un segnale di clock.
- L'automa a stati finiti che regola il ciclo di bus può essere costituito in modo sincrono o asincrono

- La temporizzazione indica il protocollo e le specifiche temporali che devono essere seguite da ogni componente che si voglia interconnettere al bus.

- Di solito nei bus di sistema o nei bus processori-memorie tali temporizzazioni sono descritte dal punto di vista della CPU che è master del bus e che gestisce tutti i segnali necessari al ciclo di bus.

Bus asincroni

- I bus asincroni sono quelli in cui non esiste un ciclo di clock, ma la temporizzazione è basata su segnali di **handshake** o interallacciato a due vie.

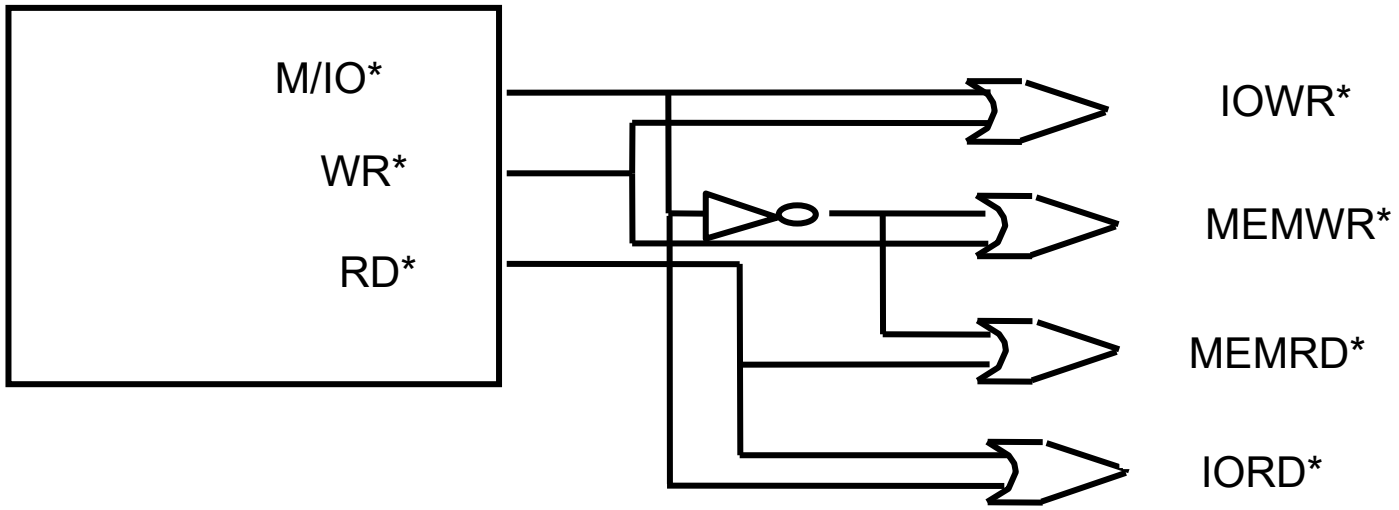


- La CPU (o il master) manda gli indirizzi e il segnale di indirizzi validi **ardy** con una latenza T1; l'unità interfacciata, es la memoria, manda il segnale di ricevuto **ack** con un ritardo T2; dopo un tempo T3 **ardy** va basso e quindi anche **ack** va basso; nel frattempo la memoria prepara il dato sul bus con un ritardo pari al tempo di accesso alla memoria T4 e con un ritardo T5 indica che il dato è pronto sul bus con **drdy**. La Cpu indica che ha letto il dato con il segnale **ack** dopo un ritardo T6 e la memoria rilascia drdy con un ritardo T7.

Interfaccia 8086 ed I/O

- Un registro di 8 latch campiona gli indirizzi quando sono pronti all'inizio del ciclo secondo il segnale ALE# e li mantiene per tutto il ciclo di bus; invece un transceiver esterno bidirezionale viene posto per bloccare i dati sul bus dei dati fissandone la direzione ed impedendone il passaggio nel periodo di tempo in cui sul bus multiplexato viaggiano i segnali d'indirizzo

- Se lo spazio separato di I/O e memoria sono necessari 4 segnali di abilitazione separati (di solito attivi bassi)



Memoria ed I/O (1/2)

- I calcolatori che dividono lo spazio di indirizzamento tra memoria ed I/O come i processori dell''Intel prevedono la presenza di un ulteriore segnale M/IO#, che indica il tipo di istruzione che viene eseguita. Le periferiche sono **mappate in I/O (isolated IO)** se insistono su uno spazio di indirizzamento separato. Ad esempio, la architettura Intel prevede una istruzione mov per i trasferimenti da e per memoria ed istruzioni in ed out per lavorare con l'input/output.

- L'architettura Pentium prevede uno spazio di 4 Gbyte di memoria con 32 bit di indirizzi, ma usa solo 64k indirizzi per l'I/O.

- Alcune case costruttrici (come ad esempio la Motorola) non prevedono spazi separati. L'architettura PowerPC non ha specifiche istruzioni ma ha spazi di indirizzamento separati e segnale corrispondente.

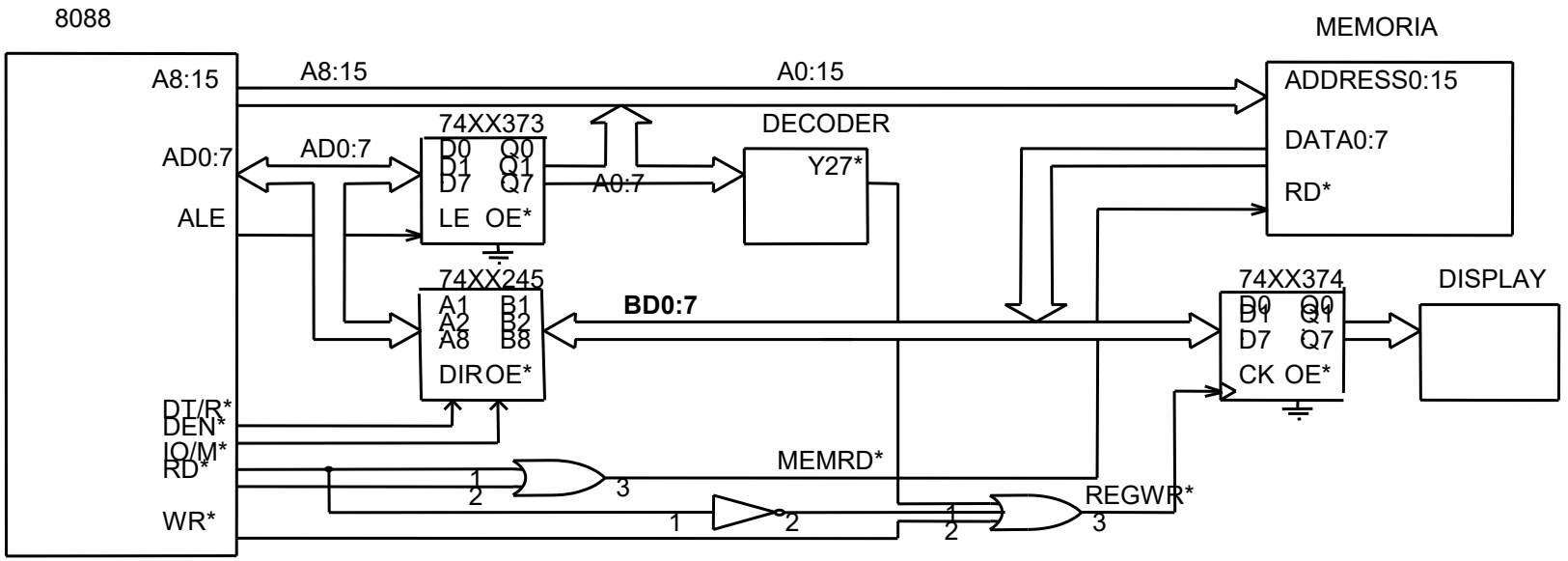
- Quando le periferiche sono nello spazio di indirizzamento della memoria si dicono mappate in memoria (**Memory mapped I/O**).

Memoria ed I/O (2/2)

Ad esempio: ipotizzando che: mov bx,100; mov ds,0; mov dx,100

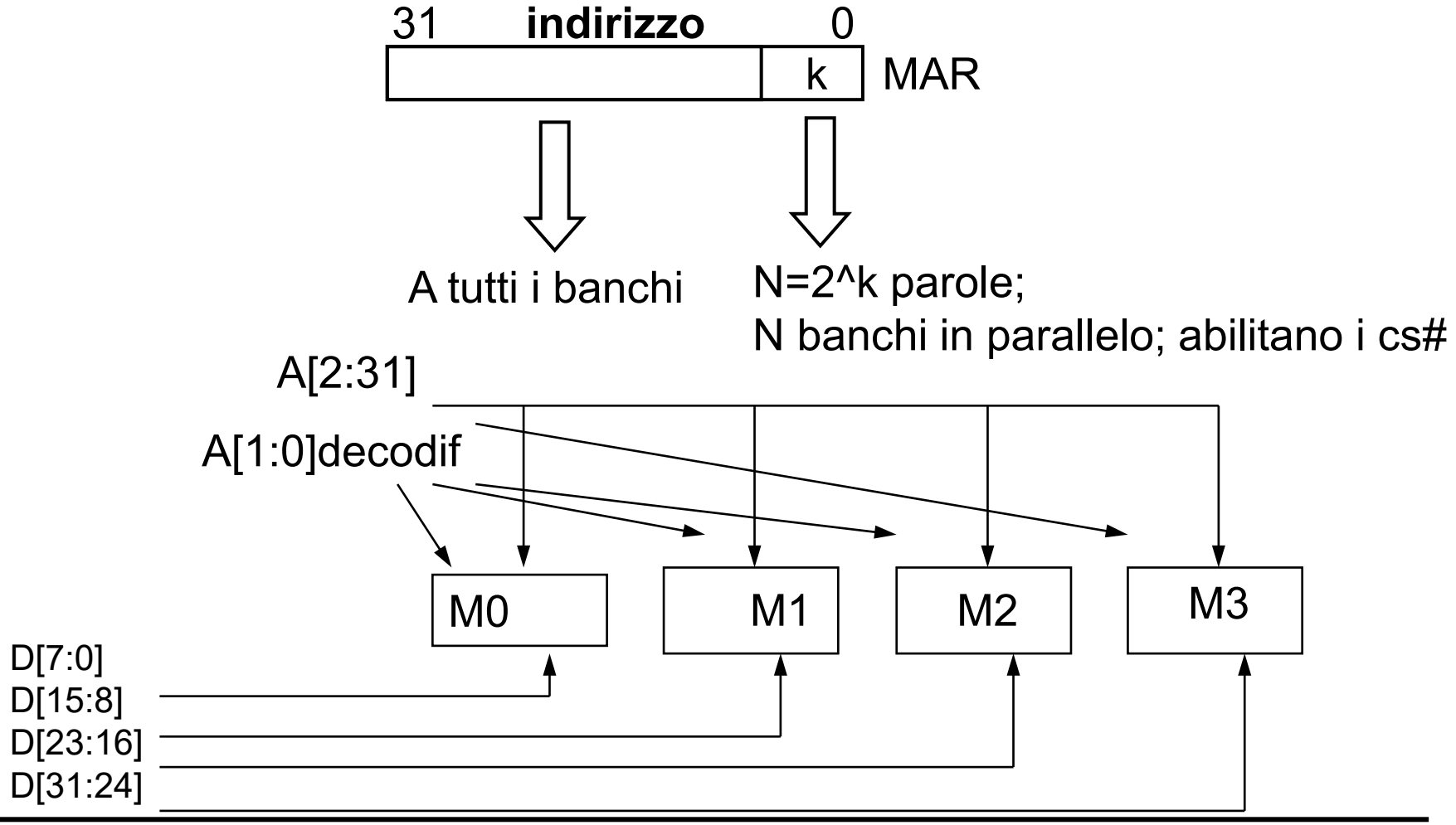
- Operazioni di trasferimento da e per memoria e da e per I/O

mov ax,[bx]	Leggi la locazione 100 di memoria verso il registro ax	IO/M#=0 RD#=0 WR#=1
mov [bx],ax	Scrivi il contenuto di ax nella locazione 100 di memoria	IO/M#=0 RD#=1 WR#=0
In ax,dx	Leggi la periferica di indirizzo 100 verso il registro ax	IO/M#=1 RD#=0 WR#=1
Out dx,ax	Scrivi il contenuto di ax nella periferica di indirizzo 100	IO/M#=1 RD#=1 WR#=0



Indirizzamento e accesso in moduli

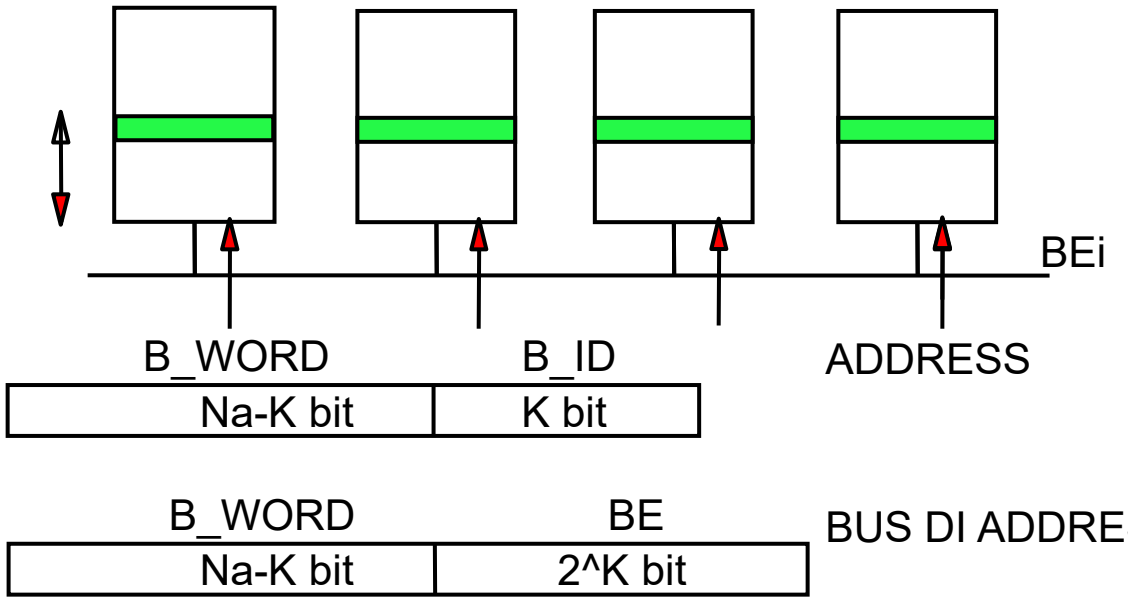
- La memoria ha un parallelismo di parola di byte
- Più memorie possono essere messe in parallelo per definire più byte
- Indirizzamento allineato



Accesso ai moduli

- In modo non allineato devo poter accedere a più banchi contemporaneamente.
- I segnali A[n:0] sono già decodificati nei processori
- Es: Pentium non ha segnali A[2:0] ma ha BE7#, BE6#.. BEi#, BE0#
- BEi# byte enable

NA = N. BIT BUS ADDRESS



accesso IN UN CICLO a k word ALLINEATE

Trasferimento dati a byte

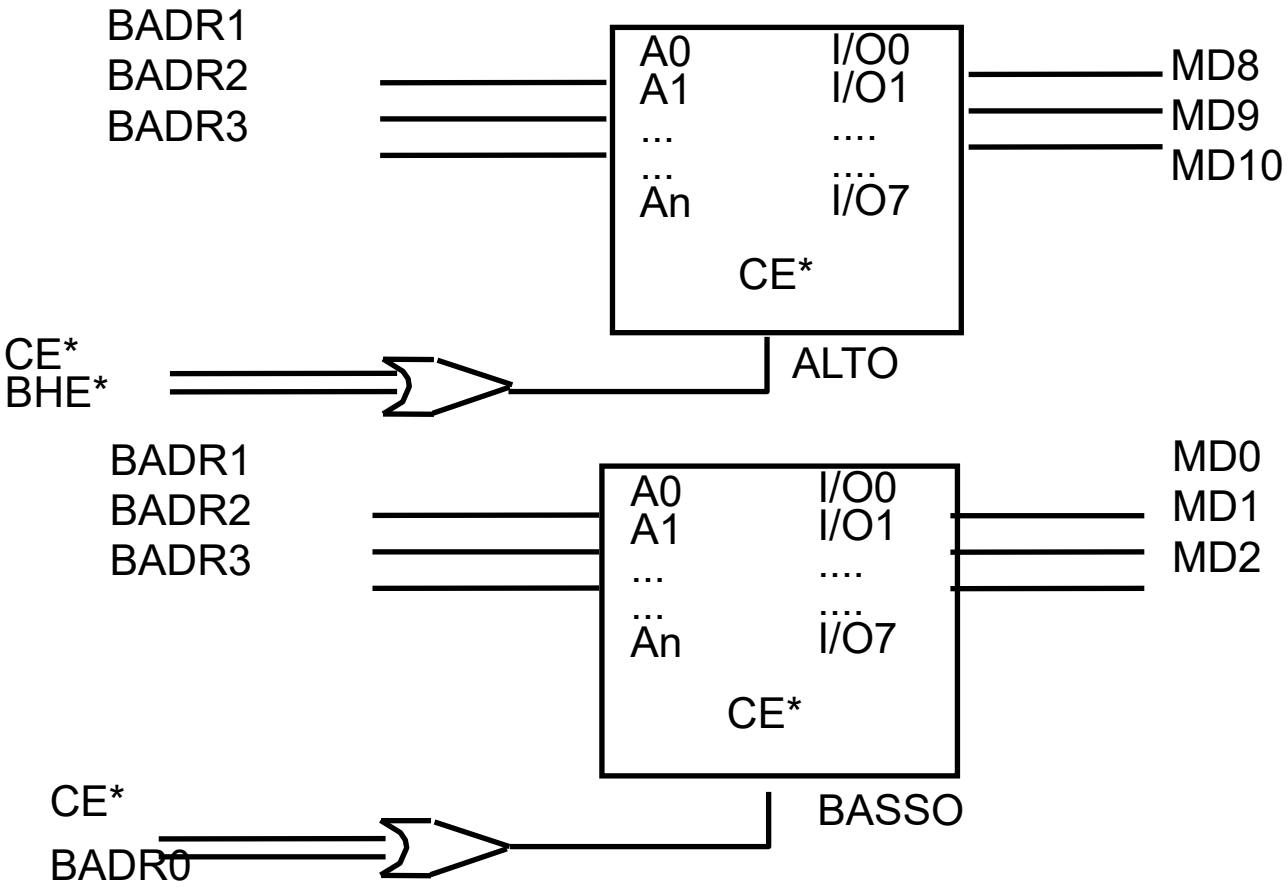
- I segnali di byte enable assieme ai segnali di indirizzo codificano quale parola e quale byte di tale parola
- ad **esempio:** processore a 32 bit di dato e 32 di indirizzo come richiede una parola a 16 bit ? Avrà i segnali D[0:31] e A[2:31] di indirizzo più BE0#, BE1#, BE2# e BE3# e asserira' allora solo BE#0 e BE1#
- Nell'8086 come in molti processori a 16 bit non allineati esistono due segnali di BHE# in questo caso si chiamano BE# e A0 anziché BE1# e BE0#

- Lo scambio byte alto esterno e byte basso avviene all'interno del microprocessore (8086)

BHE*	A0	
0	0	Word a 16 bit
0	1	Byte alto (ind dispari)
1	0	Byte basso (pari)
1	1	-

Interfaccia

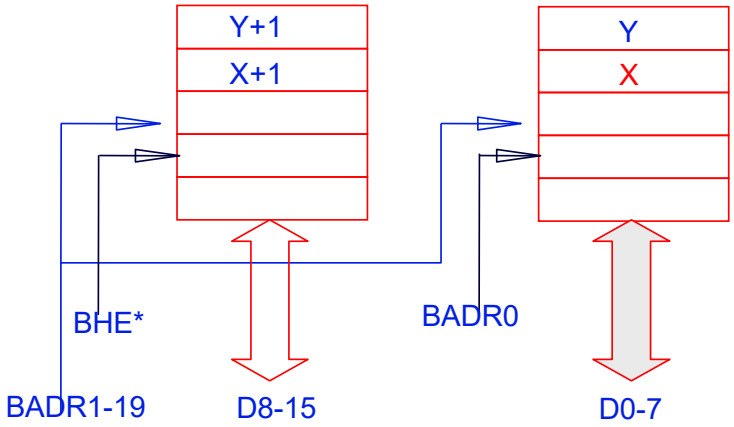
- Collegamento standard per una interfaccia con un bus a 16 bit come nell'8086



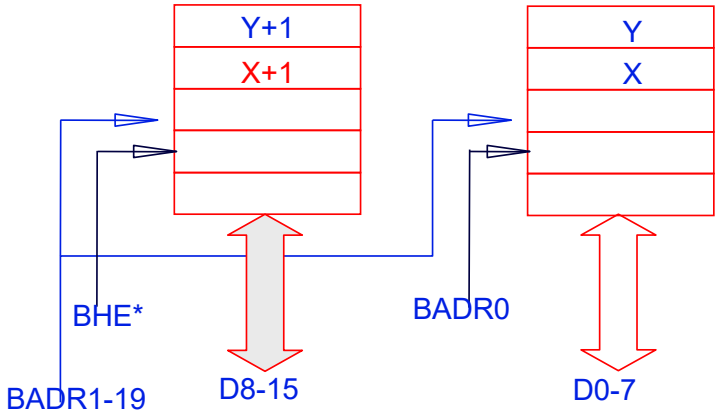
Interfaccia 8086 (1/2)

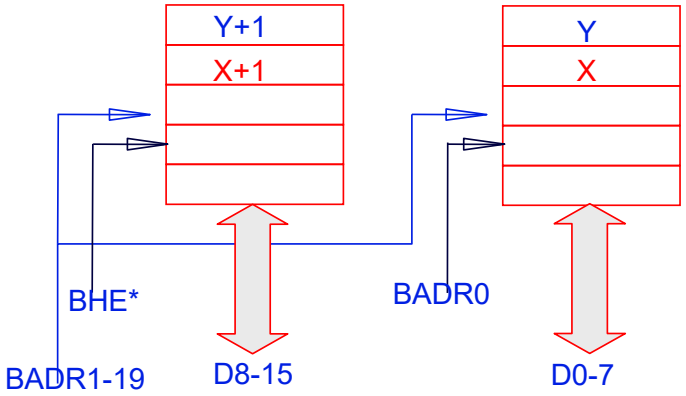
- Per trasferire un byte ad indirizzo pari o dispari

Trasferimento di byte a indirizzo pari



Trasferimento di byte a indirizzo dispari

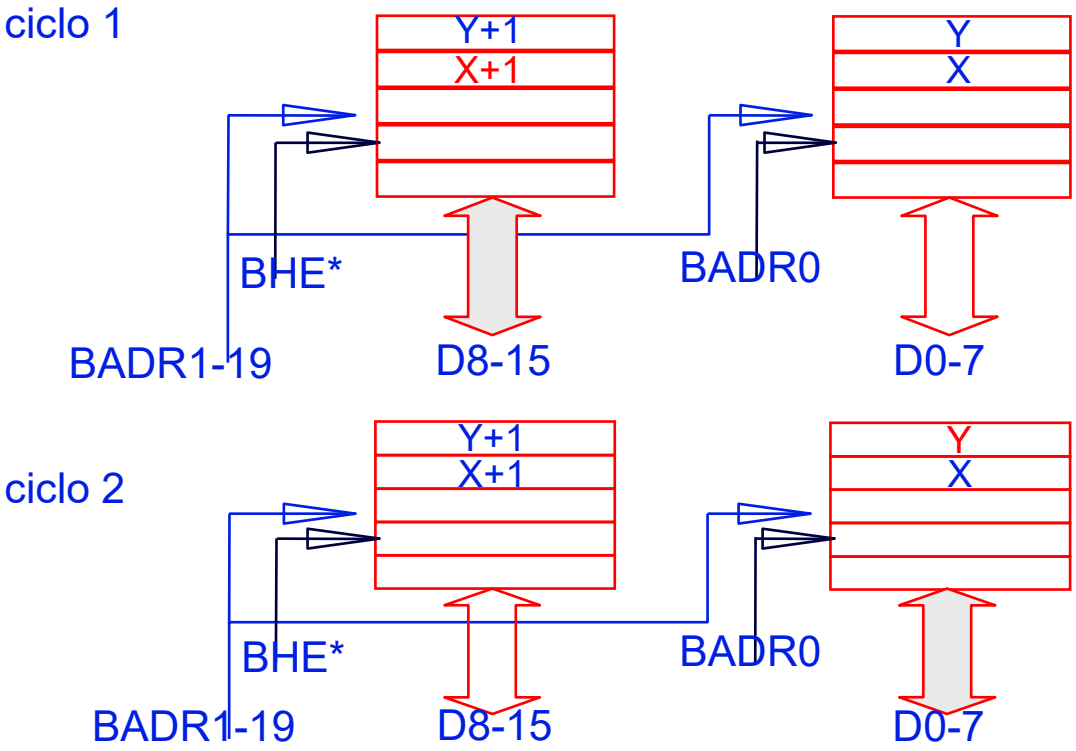




Trasferimento di word a indirizzo pari

Interfaccia 8086 (2/2)

- E per trasferire 16 bit ad indirizzo dispari?
- Non allineato, l'8086 esegue due cicli di bus

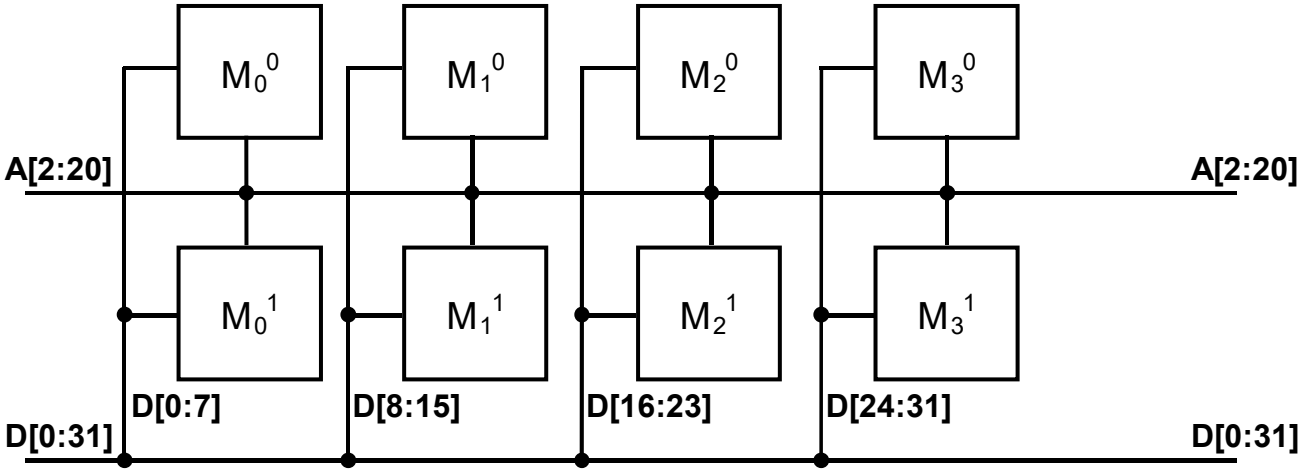


Trasferimento di word a indirizzo dispari

Esercizio Interfaccia Processore-Memorie

- Si progetti l'interfaccia tra un processore a 32 bit di indirizzamento e 32 bit di dati e avente gli stessi segnali di controllo del Pentium, con un blocco di memoria statica da 4M byte posto nella parte più bassa dello spazio di indirizzamento (avendo a disposizione banchi da 512 Kbyte)

- Bus a 32 bit:4 banchi da 8 bit (le memorie sono supposte con parallelismo M a 8 bit)
- Banchi da 512 KB: 4x512 KB = 2 MB
- Per raggiungere i 4 MB richiesti si devono utilizzare due blocchi da 4 banchi di memoria ciascuno, come nella figura seguente



Soluzione (1/2)

Vediamo il mapping dei blocchi di memoria nello spazio di indirizzamento.

(M_0^0 M_1^0 M_2^0 M_3^0)	da 0 a 2 Mbyte	(32 bit) da 00000000 a 001FFFFF
(M_0^1 M_1^1 M_2^1 M_3^1)	da 2 Mbyte a 4 Mbyte	(32 bit) da 00200000 a 003FFFFF

Quindi

	A_{31} …	A_{22}	A_{21}	A_{20}	A_{19}	… A_4	A_3	A_2	(A_1	A_0)	BE_0	BE_1	BE_2	BE_3
M_0^0	0 …	0	0	0	0	… 0	0	0			1			
				1	1	…… 1	1	1						
M_1^0	0 …	0	0	0	0	… 0	0	0				1		
				1	1	…… 1	1	1						
M_2^0	0 …	0	0	0	0	… 0	0	0					1	
				1	1	…… 1	1	1						
M_3^0	0 …	0	0	0	0	… 0	0	0						1
				1	1	…… 1	1	1						
M_0^1	0 …	0	**1**	0	0	… 0	0	0			1			
				1	1	…… 1	1	1						
M_1^1	0 …	0	**1**	0	0	… 0	0	0				1		
				1	1	…… 1	1	1						
M_2^1	0 …	0	**1**	0	0	… 0	0	0					1	
				1	1	…… 1	1	1						
M_3^1	0 …	0	**1**	0	0	… 0	0	0						1
				1	1	…… 1	1	1						

Soluzione (2/2)

Chip enable

$$CE\_M_0^0 = (\overline{A}_{31} \dots \overline{A}_{22}\ \overline{A}_{21}) \cdot BE_0$$
$$CE\_M_1^0 = (\overline{A}_{31} \dots \overline{A}_{22}\ \overline{A}_{21}) \cdot BE_1$$
$$CE\_M_2^0 = (\overline{A}_{31} \dots \overline{A}_{22}\ \overline{A}_{21}) \cdot BE_2$$
$$CE\_M_3^0 = (\overline{A}_{31} \dots \overline{A}_{22}\ \overline{A}_{21}) \cdot BE_3$$

$$CE\_M_0^1 = (\overline{A}_{31} \dots \overline{A}_{22}\ A_{21}) \cdot BE_0$$
$$CE\_M_1^1 = (\overline{A}_{31} \dots \overline{A}_{22}\ A_{21}) \cdot BE_1$$
$$CE\_M_2^1 = (\overline{A}_{31} \dots \overline{A}_{22}\ A_{21}) \cdot BE_2$$
$$CE\_M_3^1 = (\overline{A}_{31} \dots \overline{A}_{22}\ A_{21}) \cdot BE_3$$

Interfaccia Processore-Memorie priva di conflitti

- In un sistema basato su un **processore 8086** sono presenti i seguenti dispositivi:
  - 2 chip di EPROM da 128 KB agli indirizzi alti dello spazio di memoria;
  - 4 chip di RAM da 64 KB agli indirizzi bassi dello spazio di memoria;
  - 2 chip di RAM da 128 KB a partire dall'indirizzo 40000h dello spazio di indirizzamento della memoria

- Si scrivano le equazioni dei CS dei dispositivi:

  a) con la decodifica completa

  b) con la decodifica semplificata.

  La semplificazione al punto b) è operata assumendo che il programma generi soltanto indirizzi compresi negli intervalli assegnati; nel caso che il programma generi per errore altri indirizzi, è possibile che avvengano conflitti; quindi, c) si semplifichi la decodifica in modo tale che anche la generazione di indirizzi al **di fuori degli intervalli previsti per le memorie non causi conflitti sul bus**.

Soluzione (1/2)

a) Decodifica completa:

CS_EPROM_0 = A19 * A18 * $\overline{A0}$ * (M/IO#) * **$\overline{(RD\#)}$ * WR#**

CS_EPROM_1 = A19 * A18 * $\overline{BHE\#}$ * (M/IO#) * $\overline{(RD\#)}$ * WR#

CS_RAM0_0 = $\overline{A19}$ * $\overline{A18}$ * $\overline{A17}$ * $\overline{A0}$ * (M/IO#) * **$\overline{(RD\# * WR\#)}$**

CS_RAM0_1 = $\overline{A19}$ * $\overline{A18}$ * $\overline{A17}$ * $\overline{BHE\#}$ * (M/IO#) * $\overline{(RD\# * WR\#)}$

CS_RAM1_0 = $\overline{A19}$ * $\overline{A18}$ * A17 * $\overline{A0}$ * (M/IO#) * $\overline{(RD\# * WR\#)}$

CS_RAM1_1 = $\overline{A19}$ * $\overline{A18}$ * A17 * $\overline{BHE\#}$ * (M/IO#) * $\overline{(RD\# * WR\#)}$

CS_RAM2_0 = $\overline{A19}$ * A18 * $\overline{A0}$ * (M/IO#) * $\overline{(RD\# * WR\#)}$

CS_RAM2_1 = $\overline{A19}$ * A18 * $\overline{BHE\#}$ * (M/IO#) * $\overline{(RD\# * WR\#)}$

b) Decodifica semplificata, si tralasciano i segnali di controllo:

CS_EPROM = A19

CS_RAM0 = $\overline{A19}$ * $\overline{A17}$

CS_RAM1 = $\overline{A18}$ * A17

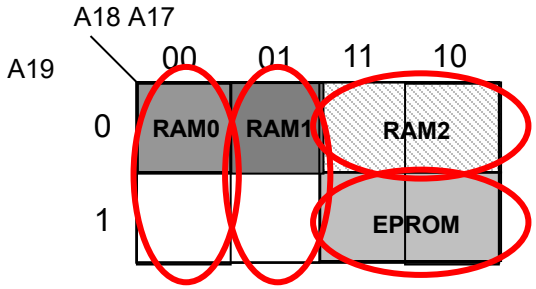
CS_RAM2 = $\overline{A19}$ * A18

Soluzione (2/2)

c) Decodifica semplificata senza conflitti, si tralasciano i segnali di controllo:

Per la decodifica semplificata b) esiste ad esempio conflitto per l'indirizzo A0000h per cui A19=1, A18=0 e A17=1 quindi sono abilitate sia la EPROM che la RAM1. Oppure per l'indirizzo 40000h per cui A19=0, A18=1 e A17=0 quindi sono abilitate sia RAM0 che RAM2.

La soluzione è utilizzare una specie di mappa di Karnaugh (reti logiche combinatorie) partendo dalla decodifica completa e cercare la copertura minima.



$CS\_EPROM = A19 * A18$

$CS\_RAM0 = \overline{A18} * \overline{A17}$

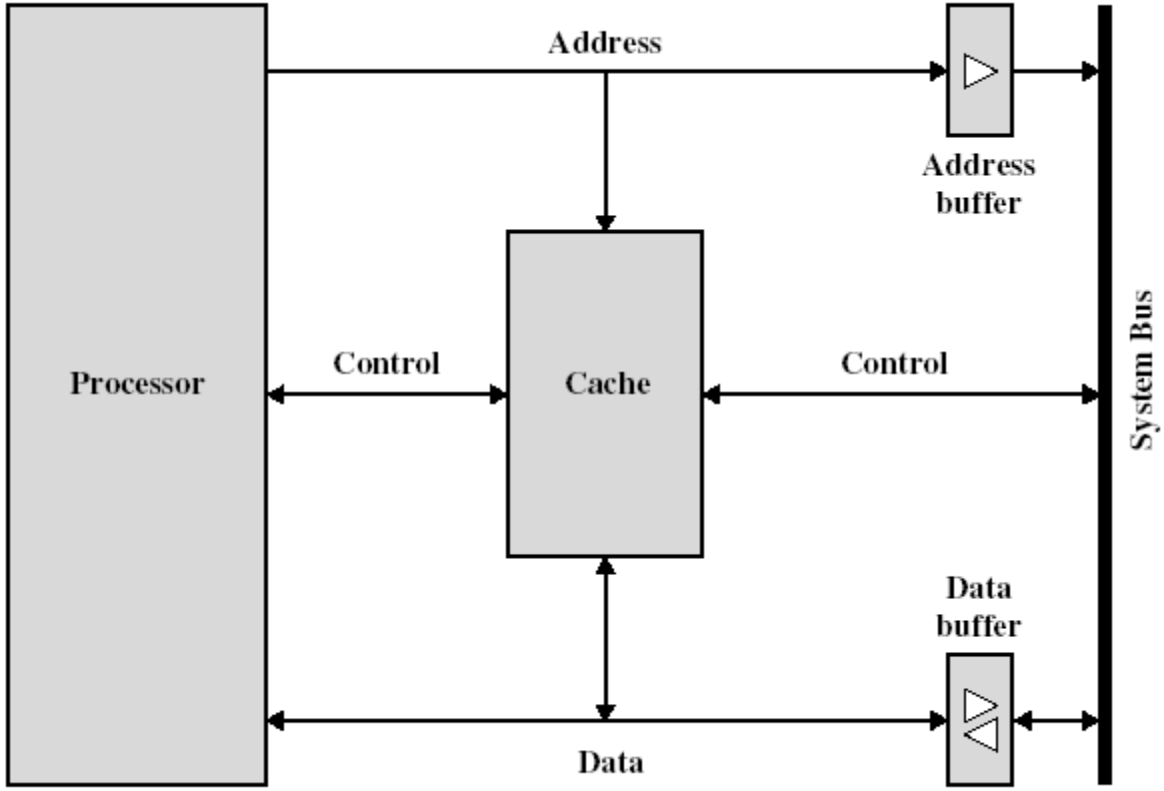
$CS\_RAM1 = \overline{A18} * A17$

$CS\_RAM2 = \overline{A19} * A18$

MEMORIE CACHE

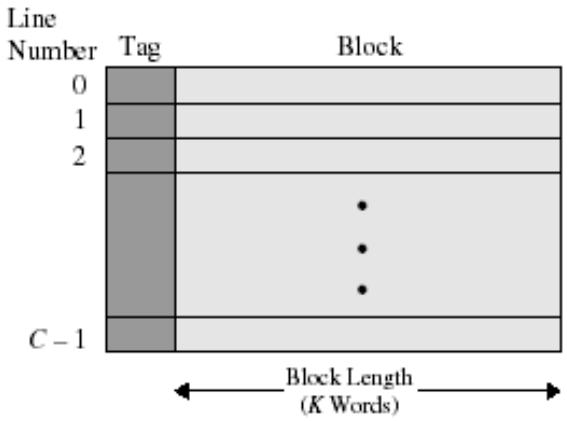
Memorie e cache

- Le memorie cache non sono visibili dal programmatore

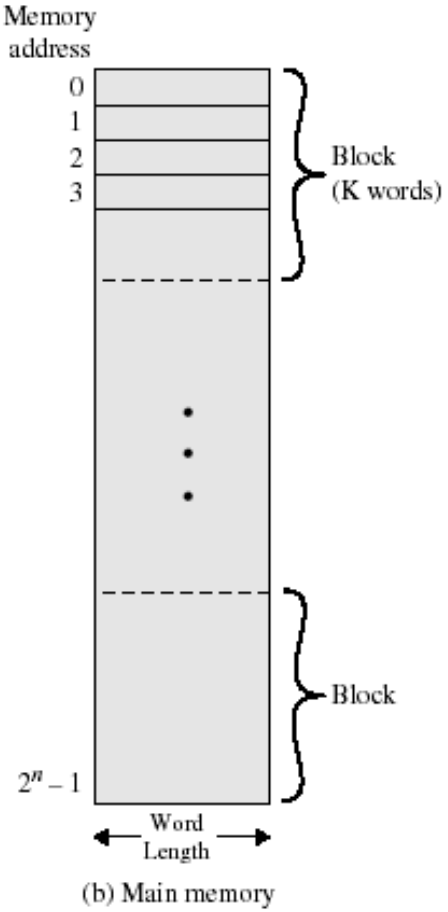


Cache

- In cache viene ricopiato un blocco di parole normalmente contenute in memoria centrale (supponiamo il parallelismo di parola = n byte così che ogni blocco B contiene nK byte)



(a) Cache



(b) Main memory

Memorie cache

Parametri di progettazione delle cache:

1) **Piazzamento** del blocco (o funzione di traduzione o mapping): dove può essere allocato il blocco al livello corrente.
2) **Identificazione** del blocco: come si può ritrovare il blocco a livello corrente
3) **Rimpiazzamento** del blocco: come si sceglie quale blocco sostituire
4) Strategia di **scrittura**

Classificazione delle miss

- **Inevitabili**: primo accesso a un blocco
- **Capacità**: miss che accadono quando si accede a un blocco sostituito
- **Conflitto**: miss che accadono poiché i blocchi sono scartati a causa di una specifica strategia di "set-mapping"

Piazzamento: Associatività (1/3)

- DOVE PUÒ ESSERE PIAZZATO UN BLOCCO?

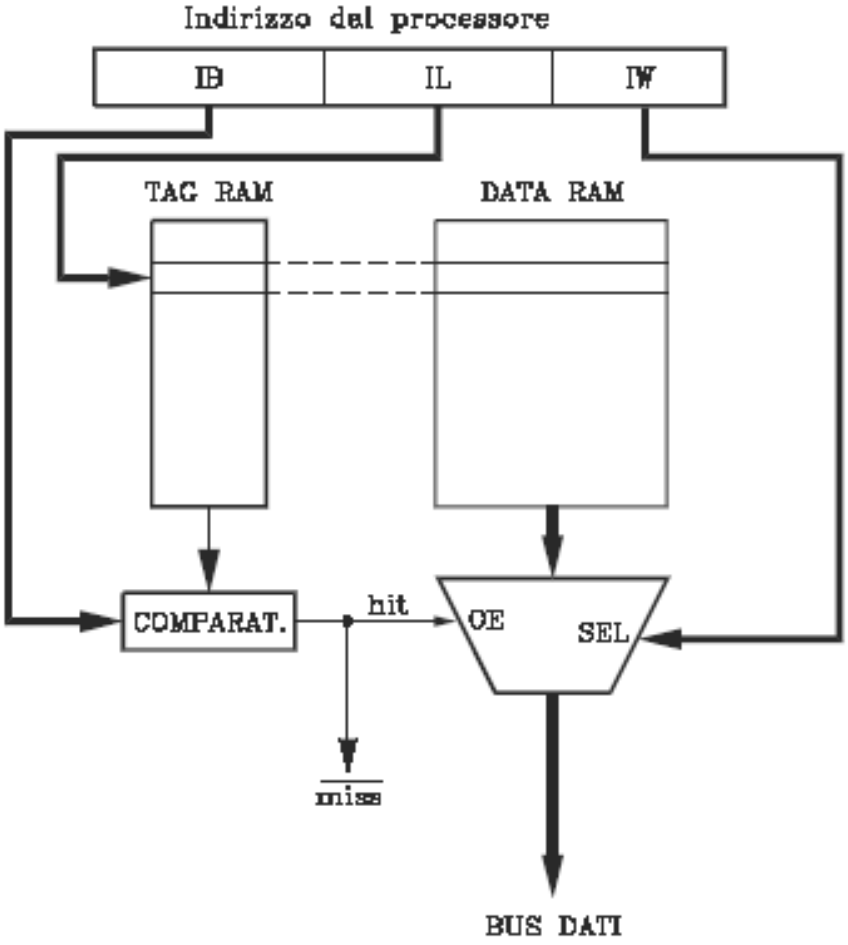
1) **DIRECT MAPPED**

Un blocco può essere posto in un solo indirizzo sulla cache

E' indicato dalla parte di indirizzo detta **index**; la parte di **offset** seleziona all'interno della linea, la rimanente parte di **tag** viene confrontata con il valore della cache directory

Baddr= (Index|Offset) mod Nb

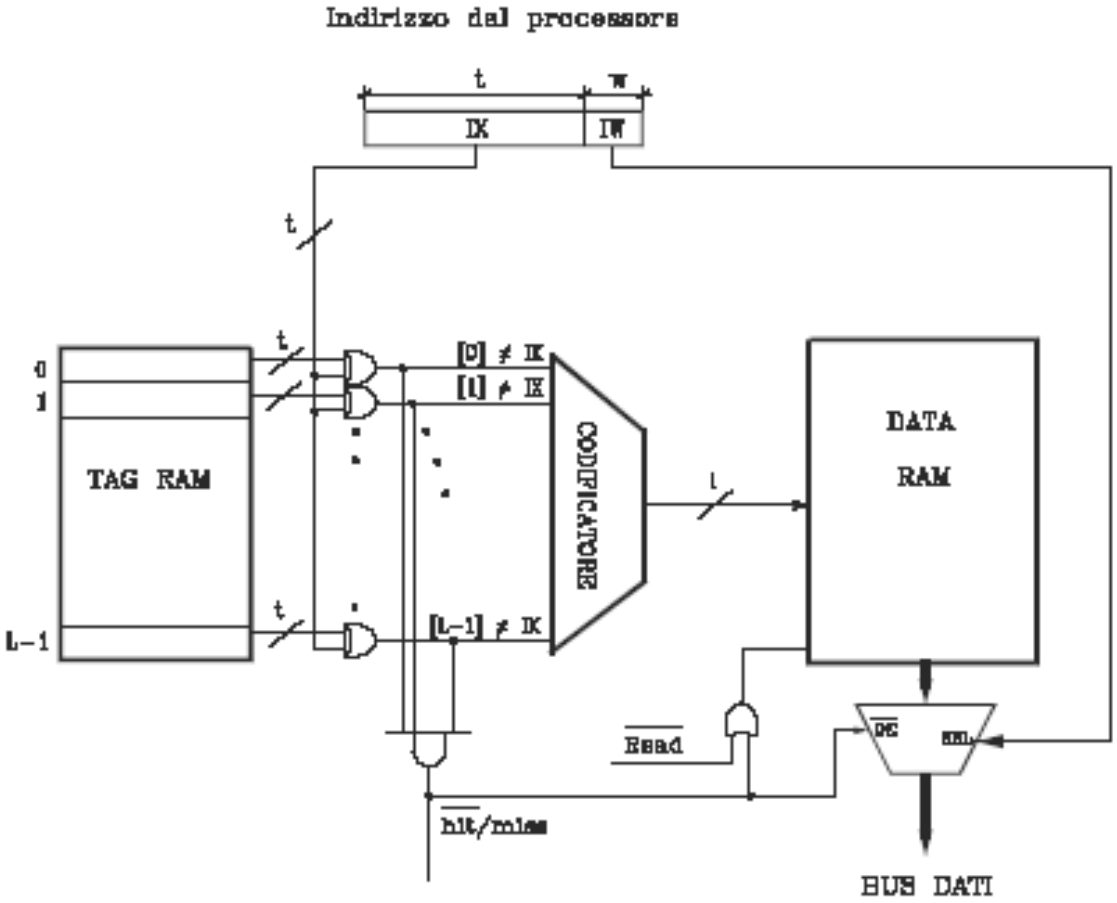
Es: 32 bit di indirizzo suddivisi in (tag | index | offset) (20|7|5)
Tutti i blocchi aventi uguali index e offset e diverso tag sono mappati allo stesso indirizzo su cache



Piazzamento: Associatività (2/3)

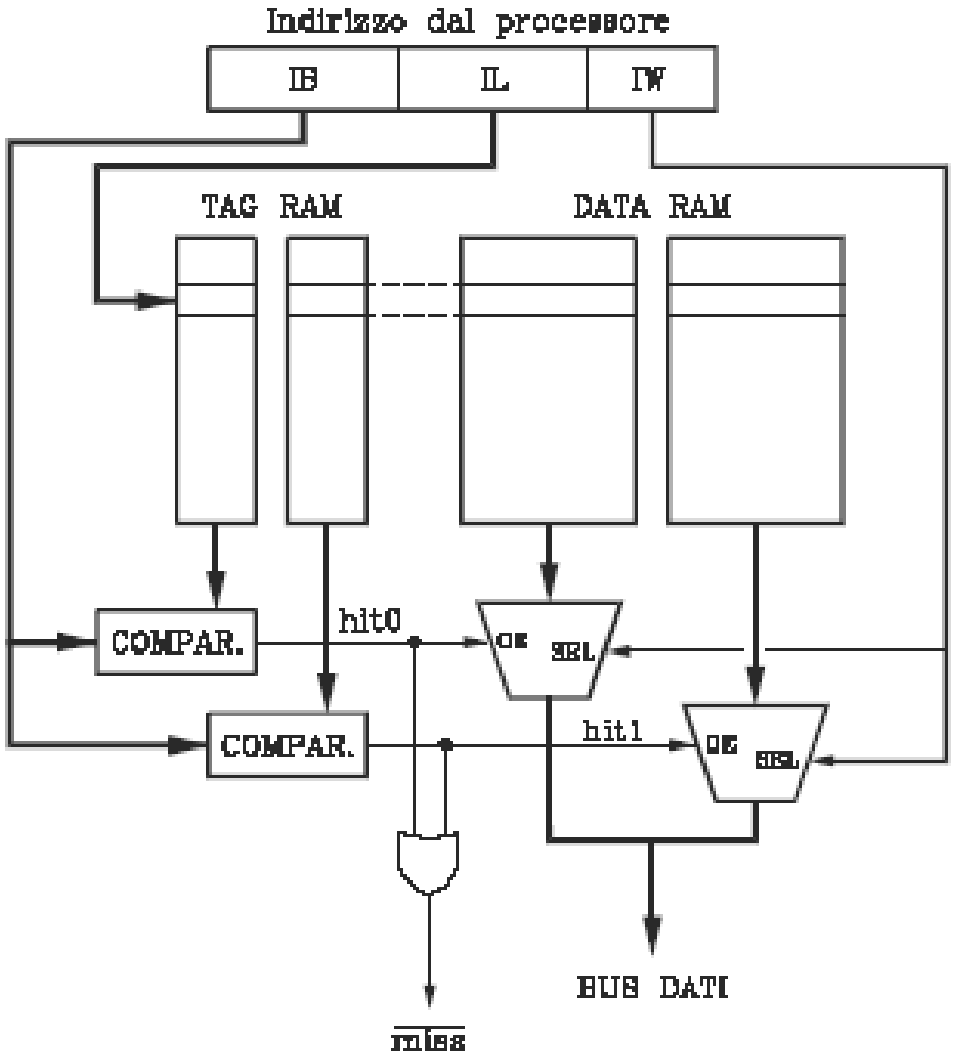
2) **FULLY ASSOCIATIVE**

Ogni blocco può essere messo ovunque nella cache; memoria **CAM** (Content Addressable Memory); manca la parte index (0 bit di index), più costosa (tanti comparatori quante le linee) e più efficace, no miss di conflitto



Piazzamento: Associatività (3/3)

- 3) **SET ASSOCIATIVE (n-way)** Ogni blocco può essere messo solo in un insieme fissato di n blocchi nella cache; la cache è divisa in $N_{BLOCCHI}/N_{SET}$ indicati dai bit di index (index minore che nel direct mapped).
- Ogni set è acceduto in modo direct mapped.
- Ogni blocco nel set è acceduto in modo fully associative, cioè richiede n comparatori;



Cache suddivise

- I processori hanno tutti ormai cache divise in **cache dati** e **cache codice**
- Permette l'accesso contemporaneo al codice (fetch) e ai dati (ld/st), evitando conflitti strutturali nella pipeline
- Sono cache di primo livello. Molti processori prevedono poi cache di secondo livello
- esempio di cache L2 (**Motorola MCM64AF2**)
- Cache di 256 KB, direct mapped:
  - 8 chip da 32 KB ciascuno per i dati
  - 1 chip da 8 KB per tag (in realtà è da 32KB anch'esso, ma i pin A13-A14 sono fissi a 0: usato per 1/4 della sua capacità)
- Linee da 32 byte $\Rightarrow$ 8192 set (256KB/32)

  5 bit di offset (A0-A4), 13 bit di index (A5-A17)

Associatività

- Associatività più larga:
  - Minore numero di miss
  - Minore varianza
  - Intuitivamente soddisfacente
- Minore associatività
  - Minore costo
  - Accesso più rapido (hit)
- Tipici valori di associatività
  - 1 per mapping diretto
  - 2,4,8,16 per associatività per set
  - Tutti i blocchi: associatività totale

Politiche di rimpiazzamento

- Politiche di rimpiazzamento
- Ad accesso diretto: non ci sono alternative. Nel caso full o n-way associative:
  - STRATEGIA **RANDOM**: rimpiazzamento casuale o pseudocasuale
  - STRATEGIA **LRU** (Least Recently Used): viene sostituito quello che è rimasto più a lungo inutilizzato

address		3	2	1	0	0	2	3	1	3	0
LRU	0	0	0	0	3	3	3	1	0	0	2

Tecniche di scrittura

- Scrittura sulle cache
- Gestione della scrittura più complessa
  - Le letture si fanno contemporaneamente al confronto dei tags. Le scritture no (e quindi le scritture sono spesso più lente)
- Se si è in presenza di un "hit"
  - **write-through** (store through) si scrive sempre non solo nella cache ma anche nella memoria centrale
  - **write-back** (store in, copy-back) si scrive sulla memoria centrale solo quando il blocco deve essere rimpiazzato
- In caso di miss si rimpiazzano i blocchi ?
  - Si - **write-allocate** (normalmente usato con il write-back); si alloca sulla cache e poi si modifica
  - No - **no-write-allocate** (normalmente usato con il write-through) si scrive solo nella memoria gerarchicamente più bassa
- nel Pentium esiste un pin esterno per specificare di tipo WT o WB; per usare la politica WB è necessario predisporre una politica e i corrispondenti meccanismi hardware per gestire la coerenza dei dati in memoria e sulle cache in sistemi multiprocessore

Pentium

- Cache PENTIUM
- Dato = 32 bytes (include 5 bit di offset)
- Index = 128 valori (include 7 bit di index)
- Tag = 32 - (5+7) = 20 bit

- Spazio di 4 Gbyte diviso in 2^{20} cache pages di 4 Kbyte che diviso in 128 set è diviso in 32 byte cache lines

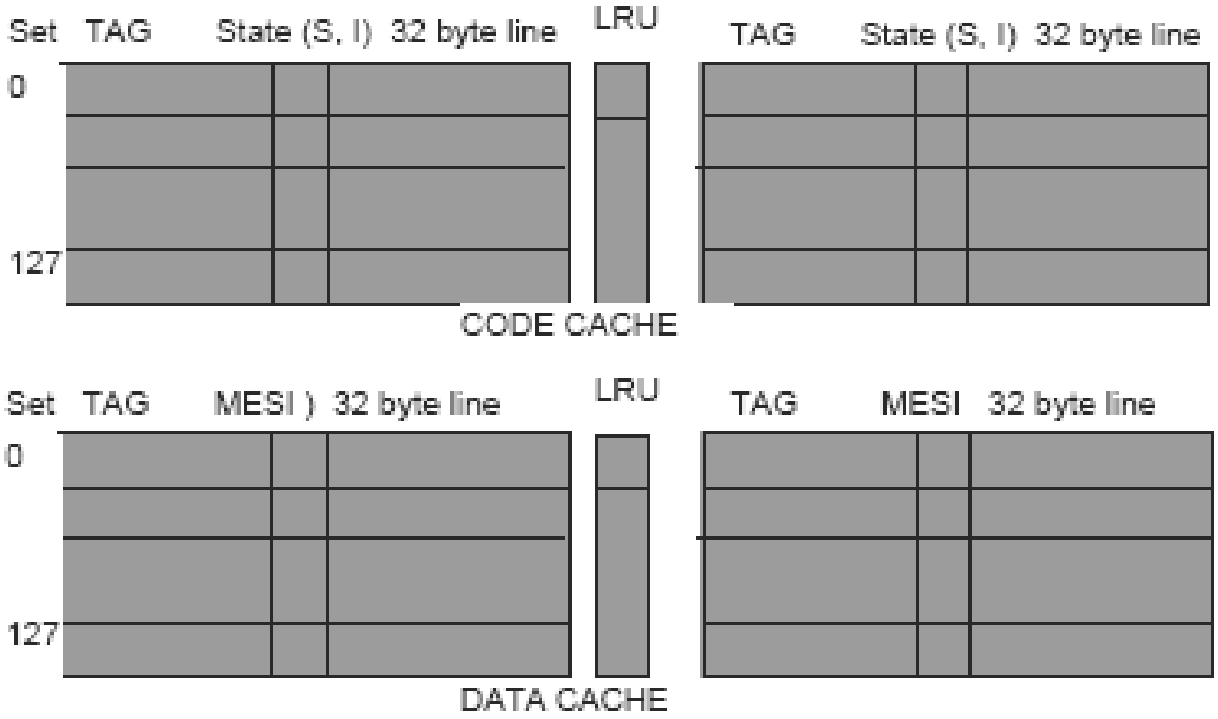
- La cache del PENTIUM (dati e istruzioni) è associativa a 2 vie. Dimensione di ciascuna cache: 128 * 2 * 32 = 8 Kbytes

- cache di secondo livello a 256 Kbyte 4-way associative con 2048 sets
- Algoritmo di Rimpiazzamento dei dati : LRU
- Per ogni linea di data cache vi sono due bit di stato (MESI)
- Per ogni linea di code cache vi e' un bit di stato (SI)

Cache Pentium L1

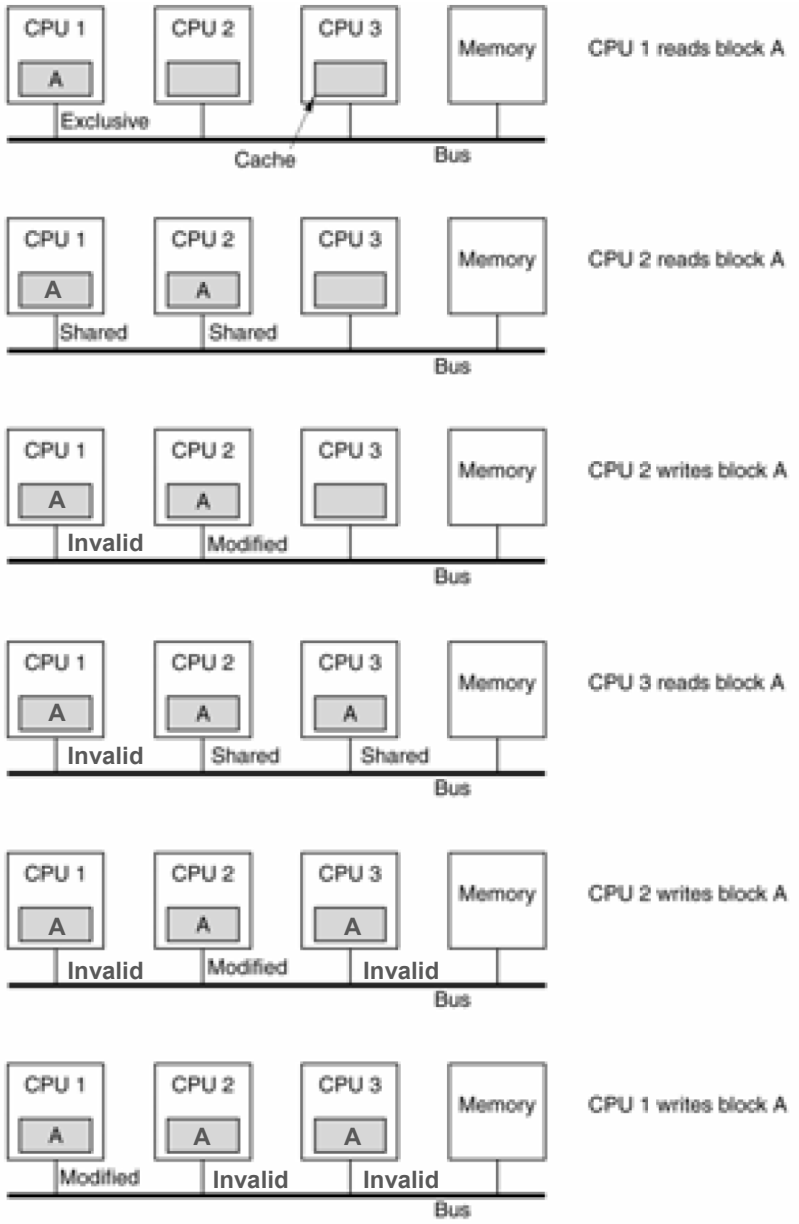
- Es: **cache a 2 vie nel Pentium (8K)**

- Con l'index si seleziona uno dei 128 set a due blocchi, che hanno lo stesso bit di rimpiazzamento LRU. Ogni blocco ha i propri bit di stato (diversi nel caso CODE e nel caso DATA)



Protocollo M.E.S.I.

- Protocollo per sistemi multiprocessore.
- Per ogni linea presente nella cache e' necessario sapere in quale stato si trova per motivi di coerenza delle caches presenti nei vari processori (2 bit)
- **M - modified**
  - La linea è disponibile in una sola cache ed è stata modificata senza essere stata riscritta sulla memoria. Alla linea si può accedere senza ciclo di bus
- **E - exclusive**
  - La linea è disponibile in una sola cache ma non è stata modificata. Una scrittura la porta in M
- **S - shared**
  - La linea potenzialmente è presente in più caches. Una scrittura sulla cache locale causa un WRITE-THROUGH e invalida la stessa linea nelle altre caches
- **I - invalid**
  - La linea non è disponibile nella cache e l'accesso in lettura causa una allocazione (LINE FILL). Una scrittura provoca solo un WRITE THROUGH.
- La cache del codice può essere solo S o I



Cache memory

Table 4.3 Cache Sizes of Some Processors

Processor	Type	Year of Introduction	L1 cache[a]	L2 cache	L3 cache
IBM 360/85	Mainframe	1968	16 to 32 KB	—	—
PDP-11/70	Minicomputer	1975	1 KB	—	—
VAX 11/780	Minicomputer	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 to 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 to 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 KB	2 MB	—
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 KB/32 KB	4 MB	—

[a] Two values seperated by a slash refer to instruction and data caches

[b] Both caches are instruction only; no data caches

Confronto Processori Intel

Processor	Series Nomenclature	Code Name	Production Date	Supported Features (Instruction Set)	Clock Rate	Socket	Fabrication	TDP	Number of Cores	Bus Speed	L1 Cache	L2 Cache	L3 Cache
Intel Pentium	N/A	P5, P54C, P54CTB, P54CS	1993 - 1999		60 MHz - 200 MHz	Socket 2, Socket 3, Socket 4, Socket 5, Socket 7	800 nm - 350 nm	Unknown	Single	50 MHz - 66 MHz	16 KiB	N/A	N/A
Intel Pentium MMX	N/A	P55C, Tillamook	1996 - 1999		120 MHz - 300 MHz	Socket 7	350 nm - 250 nm	Unknown	Single	60 MHz - 66 MHz	32 KiB	N/A	N/A
Intel Atom	Z5xx, Z6xx, N2xx, 2xx, 3xx, N4xx, D4xx, D5xx, N5xx, D2xxx, N2xxx	Diamondville, Pineview, Silverthorne, Lincroft, Cedarview, Medfield, Clover Trail	2008 - 2009 (as Centrino Atom) 2008 - present (as Atom)		800 MHz - 2.13 GHz	Socket PBGA437, Socket PBGA441, Socket micro-FCBGA8 559	32 nm, 45 nm	0.65 W - 13 W	Single, Double	400 MHz, 533 MHz, 667 MHz, 2.5 GT/s	56 KiB per core	512 KiB - 1 MiB	N/A
Intel Celeron	3xx, 4xx, 5xx	Banias, Cedar Mill, Conroe, Coppermine, Covington, Dothan, Mendocino, Northwood, Prescott, Tualatin, Willamette, Yonah	1998 - present		266 MHz - 3.6 GHz	Slot 1, Socket 370, Socket 478, Socket 479, Socket 495, LGA 775, Socket M, Socket T	45 nm, 65 nm, 90 nm, 130 nm, 180 nm, 250 nm	5.5 W - 86 W	Single, Double	66 MHz, 100 MHz, 133 MHz, 400 MHz, 533 MHz, 800 MHz	8 KiB ~ 64 KiB per core	0 KiB - 1 MiB	0 KiB - 2 MiB
Intel Pentium Pro	52x	P6	1995 - 1998		150 MHz - 200 MHz	Socket 8	350 nm, 500 nm	29.2 W - 47 W	Single	60 MHz, 66 MHz	16 KiB	256 KiB, 512 KiB, 1024 KiB	N/A
Intel Pentium II	52x	Klamath, Deschutes, Tonga, Dixon	1997 - 1999		233 MHz - 450 MHz	Slot 1, MMC-1, MMC-2, Mini-Cartridge	250 nm, 350 nm	16.8 W - 38.2 W	Single	66 MHz, 100 MHz	32KiB	256 KiB - 512 KiB	N/A
Intel Pentium III	52x, 53x	Katmai, Coppermine, Tualatin	1999 - 2003		450 MHz - 1.4 GHz	Slot 1, Socket 370	130 nm, 180 nm, 250 nm	17 W - 34.5 W	Single	100 MHz, 133 MHz	32 KiB	256 KiB - 512 KiB	N/A
Intel Xeon	n3xxx, n5xxx, n7xxx	Allendale, Cascades, Clovertown, Conroe, Cranford, Dempsey, Drake, Dunnington, Foster, Gainestown, Gallatin, Harpertown, Irwindale, Kentsfield, Nocona, Paxville, Potomac, Prestonia, Sossaman, Tanner, Tigerton, Tulsa, Wolfdale, Woodcrest	1998 - present		400 MHz - 4.4 GHz	Slot 2, Socket 603, Socket 604, Socket J, Socket T, Socket B, LGA 1150, LGA 1155, LGA 1156, LGA 1366, LGA 2011	45 nm, 65 nm, 90 nm, 130 nm, 180 nm, 250 nm	16 W - 165 W	Single, Double, Quad, Hexa, Octa	100 MHz, 133 MHz, 400 MHz, 533 MHz, 667 MHz, 800 MHz, 1066 MHz, 1333 MHz, 1600 MHz, 4.8 GT/s, 5.86 GT/s, 6.4 GT/s	8 KiB ~ 64 KiB per core	256 KiB - 12 MiB	4 MiB - 16 MiB
Pentium 4	5xx, 6xx	Cedar Mill, Northwood, Prescott, Willamette	2000 - 2008		1.3 GHz - 3.8 GHz	Socket 423, Socket 478, LGA 775, Socket T	65 nm, 90 nm, 130 nm, 180 nm	21 W - 115 W	Single	400 MHz, 533 MHz, 800 MHz, 1066 MHz	8 KiB - 16 KiB	256 KiB - 2 MiB	2 MiB
Processor	5xx, 6xx Series Nomenclature	Gallatin, Code Name	2000 - 2008 Production Date	Supported Features (Instruction Set)	Clock Rate	Socket 478, Socket	90 nm, 130 nm Fabrication	TDP	Single Number of Cores	800 MHz, 1066 MHz Bus Speed	8 KiB L1 Cache	L2 Cache	L3 Cache

Confronto Processori Intel

Processor	Series Nomenclature	Code Name	Production Date	Supported Features (Instruction Set)	Clock Rate	Socket	Fabrication	TDP	Number of Cores	Bus Speed	L1 Cache	L2 Cache	L3 Cache
Pentium 4 Extreme Edition		Prescott 2M			3.2 GHz - 3.73 GHz	Socket T		92 W - 115 W				512 KiB - 1 MiB	0 KiB - 2 MiB
Pentium M	7xx	Banias, Dothan	2003 - 2008		800 MHz - 2.266 GHz	Socket 479	90 nm, 130 nm	5.5 W - 27 W	Single	400 MHz, 533 MHz	32 KiB	1 MiB - 2 MiB	N/A
Pentium D/EE	8xx, 9xx	Smithfield, Presler	2005 - 2008		2.66 GHz - 3.73 GHz	Socket T	65 nm, 90 nm	95 W - 130 W	Double	533 MHz, 800 MHz, 1066 MHz	16 KiB per core	2×1 MiB - 2×2 MiB	N/A
Intel Pentium Dual-Core	E2xxx, E3xxx, E5xxx, T2xxx, T3xxx	Allendale, Penryn, Wolfdale, Yonah	2006 - 2009		1.6 GHz - 2.93 GHz	Socket 775, Socket M, Socket P, Socket T	45 nm, 65 nm	10 W - 65 W	Double	533 MHz, 667 MHz, 800 MHz, 1066 MHz	64 KiB per core	1 MiB - 2 MiB	N/A
Intel Pentium New	E5xxx, E6xxx, T4xxx, SU2xxx, SU4xxx, G69xx, P6xxx, U5xxx, G6xx, G8xx, B9xx	Penryn, Wolfdale, Clarkdale, Sandy Bridge,			1.2 GHz - 3.33 GHz	Socket 775, Socket P, Socket T, LGA 1156, LGA 1155,	32 nm, 45 nm, 65 nm	5.5 W - 73 W	Single, Double	800 MHz, 1066 MHz, 2.5GT/s, 5 GT/s	64 KiB per core	2x256 KiB - 2 MiB	0 KiB - 3 MiB
Intel Core	Txxxx, Lxxxx, Uxxxx	Yonah	2006 - 2008		1.06 GHz - 2.33 GHz	Socket M	65 nm	5.5 W - 49 W	Single, Double	533 MHz, 667 MHz	64 KiB per core	2 MiB	N/A
Intel Core 2	Uxxxx, Lxxxx, Exxxx, Txxxx, P7xxx, Xxxxx, Qxxxx, QXxxxx	Allendale, Conroe, Merom, Penryn, Kentsfield, Wolfdale, Yorkfield	2006 - 2011		1.06 GHz - 3.33 GHz	Socket 775, Socket M, Socket P, Socket J, Socket T	45 nm, 65 nm	5.5 W - 150 W	Single, Double, Quad	533 MHz, 667 MHz, 800 MHz, 1066 MHz, 1333 MHz, 1600 MHz	64 KiB per core	1 MiB - 12 MiB	N/A
Intel Core i3	i3-xxx, i3-2xxx, i3-3xxx, i3-4xxx	Arrandale, Clarkdale, Sandy Bridge, Ivy Bridge, Haswell	2010 - present		1.2 GHz - 3.7 GHz	LGA 1156, LGA 1155, LGA 1150	22 nm, 32 nm	35 W - 73 W	Double	1066 MHz, 1600 MHz, 2.5 - 5 GT/s	64 KiB per core	256 KiB	3 MiB - 4 MiB
Intel Core i5	i5-7xx, i5-6xx, i5-2xxx, i5-3xxx, i5-4xxx	Arrandale, Clarkdale, Clarksfield, Lynnfield, Sandy Bridge, Ivy Bridge, Haswell	2009 - present		1.06 GHz - 3.6 GHz	LGA 1156, LGA 1155, LGA 1150	22 nm, 32 nm, 45 nm	17 W - 95 W	Double, Quad	2.5 - 5 GT/s	64 KiB per core	256 KiB	4 MiB - 8 MiB
Intel Core i7	i7-6xx, i7-7xx, i7-8xx, i7-9xx, i7-2xxx, i7-37xx, i7-38xx, i7-47xx, i7-48xx	Bloomfield,	2008 - present		1.6 GHz - 4.4 GHz	LGA 1156, LGA 1155, LGA 1366, LGA 2011, LGA 1150	22 nm, 32 nm, 45 nm	45 W - 130 W	Quad	4.8 GT/s, 6.4 GT/s	64 KiB per core	4×256 KiB	6 MiB - 10 MiB
Processor	Series Nomenclature	Code Name	Production Date	Supported Features (Instruction Set)	Clock Rate	Socket	Fabrication	TDP	Number of Cores	Bus Speed	L1 Cache	L2 Cache	L3 Cache

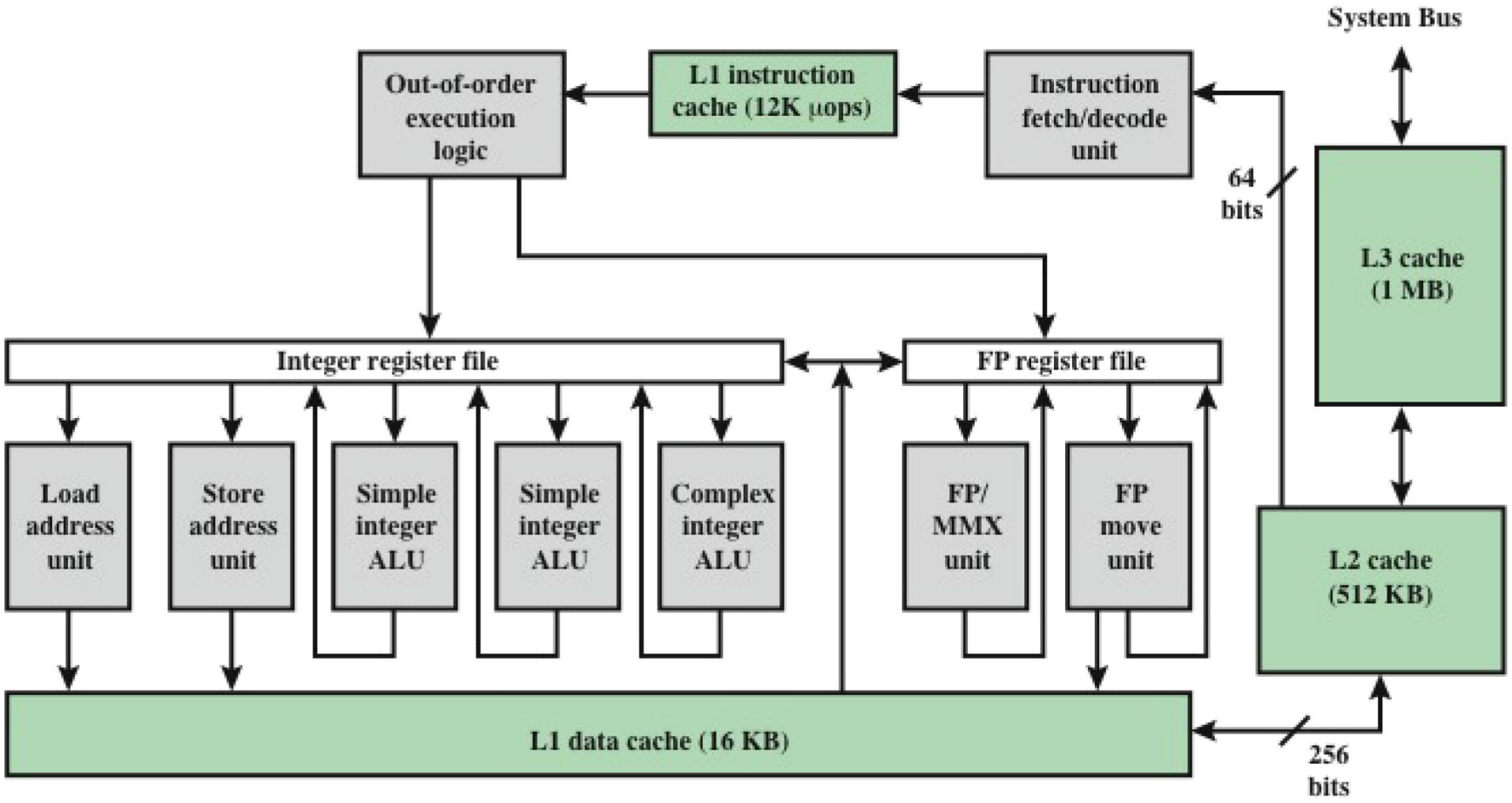
Confronto Processori Intel

Processor	Series Nomenclature	Code Name	Production Date	Supported Features (Instruction Set)	Clock Rate	Socket	Fabrication	TDP	Number of Cores	Bus Speed	L1 Cache	L2 Cache	L3 Cache
		Nehalem, Clarksfield, Clarksfield XM, Lynnfield, Sandy Bridge, Sandy Bridge-E, Ivy Bridge, Ivy Bridge-E, Haswell, Haswell Refresh, Devil's Canyon											
Intel Core i7	i7-970, i7-980, i7-980x, i7-990x, i7-39xx, i7-49xx, i7 58xx, i7 59xx	Gulftown, Sandy Bridge-E, Ivy Bridge-E, Haswell-E	2011 - present		3.0 GHz - 4.0 GHz	LGA 1366, LGA 2011, LGA 2011-v3	32 nm, 22 nm	130 W - 150 W	Quad, Hexa, Octa	2.5GT/s - 6.4 GT/s	64 KiB per core	6x256 KiB	12 MiB - 20 MiB
Processor	Series Nomenclature	Code Name	Production Date	Supported Features (Instruction Set)	Clock Rate	Socket	Fabrication	TDP	Number of Cores	Bus Speed	L1 Cache	L2 Cache	L3 Cache

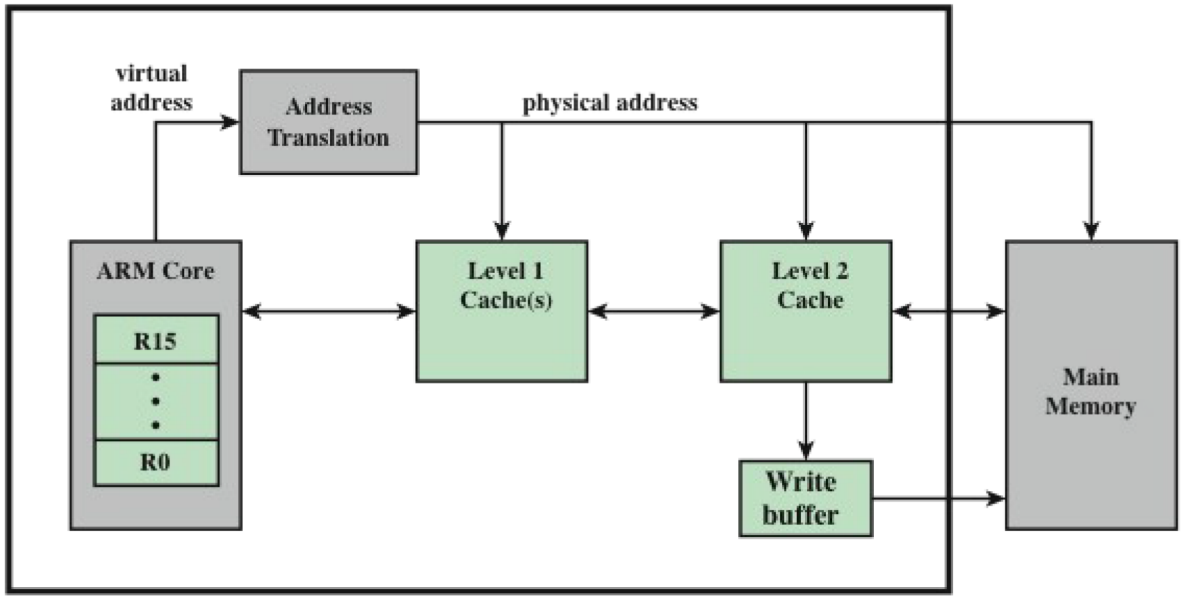
Cache e processori x86

External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Esempio: Pentium 4



Esempio: ARM



- Presenza di un buffer di scrittura
  - Piccolo: 4 parole
- In pratica gestisce la scrittura in parallelo rispetto alla cache
- Piú velocemente rispetto la cache
- Fino a che non è vuoto quell'area di memoria non può essere letta

MEMORIA VIRTUALE

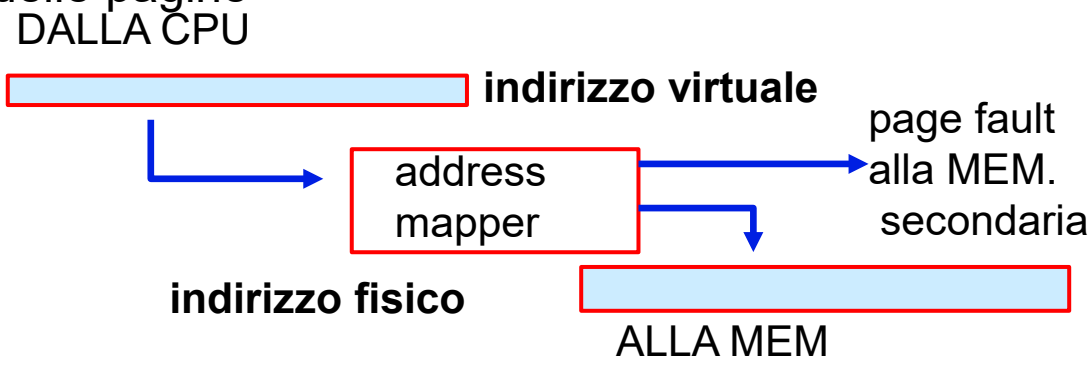
Memoria virtuale (1/4)

- Tecnica gestita dal sistema operativo per fare in modo che il processore veda virtualmente tutto lo spazio di indirizzamento malgrado la memoria centrale sia limitata. E' il livello della gerarchia di memoria tra memoria centrale e hard disk.
- Se, ad esempio, ho un sistema con solo 4 KB di memoria centrale potrei, in teoria, gestire solo gli indirizzi da 0 a 4095. Cosa succede se voglio accedere ad un indirizzo tra 5023 e 9118 (4 KB)? Se non avessi la memoria virtuale non posso che generare un errore. Con la memoria virtuale si eseguono le seguenti operazioni:
  - Il contenuto della memoria centrale è salvato sul disco
  - Vengono localizzati i dati con indirizzo tra 5023 e 9118 sul disco
  - Questi dati vengono copiati in memoria centrale
  - Si modifica la mappatura degli indirizzo per far sì che a 0 corrisponda 5023 e linearmente fino a 4095 che corrisponde a 9118

- Questa tecnica si basa sulla **paginazione**: lo spazio di indirizzamento della memoria centrale viene diviso in blocchi di dimensione fissata (**pagine**) contigui.
- L'indirizzo può sempre essere diviso in un indice di pagina e in un offset rispetto alla pagina

Memoria virtuale (2/4)

- Nel caso della memoria virtuale si ha un metodo diverso di accesso ai dati.
- Il processore usa lo spazio di indirizzamento fisico (tutto quello possibile) per fare riferimento ad uno spazio reale molto minore; quindi l'indirizzo che viene impiegato è in realtà virtuale e non corrisponde 1:1 con le locazioni di memoria reali.
- E' compito del sistema operativo gestire la paginazione (ossia i blocchi) e spostarli dalla memoria di massa alla memoria principale
- E' necessario un metodo di **traduzione degli indirizzi**, attraverso la tabella delle pagine

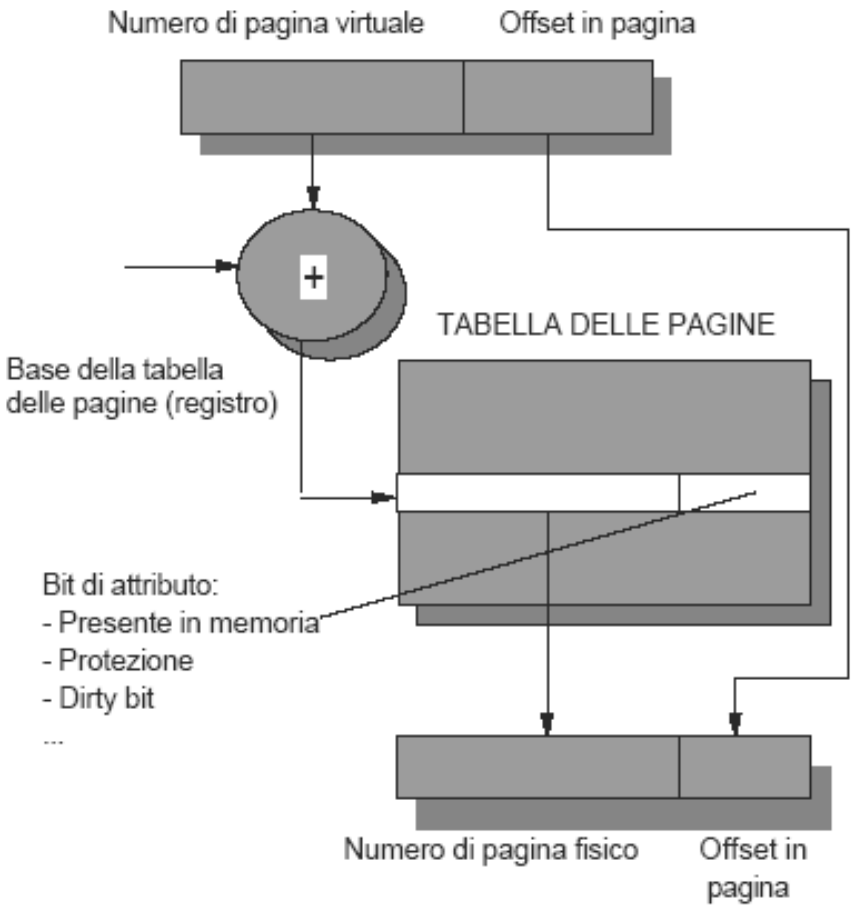


- blocchi (pagine) da 512 a 16K(4M) bytes

- Posizionamento delle pagine:
  - Totalmente associativo

Memoria virtuale (3/4)

- **Identificazione delle pagine**
  - Traduzione dell'indirizzo (virtuale -> fisico)
  - Indiretto con tabelle delle pagine
  - Traduzione "cached" in un buffer di traslazione (TLB)
- **Rimpiazzamento** delle pagine
  - Quasi sempre del tipo Least Recently Used, dividendo le pagine in due gruppi, di uso recente e remoto
- **Strategie di scrittura**
  - Write-back (usando un bit di pagina sporca – *dirty bit*)



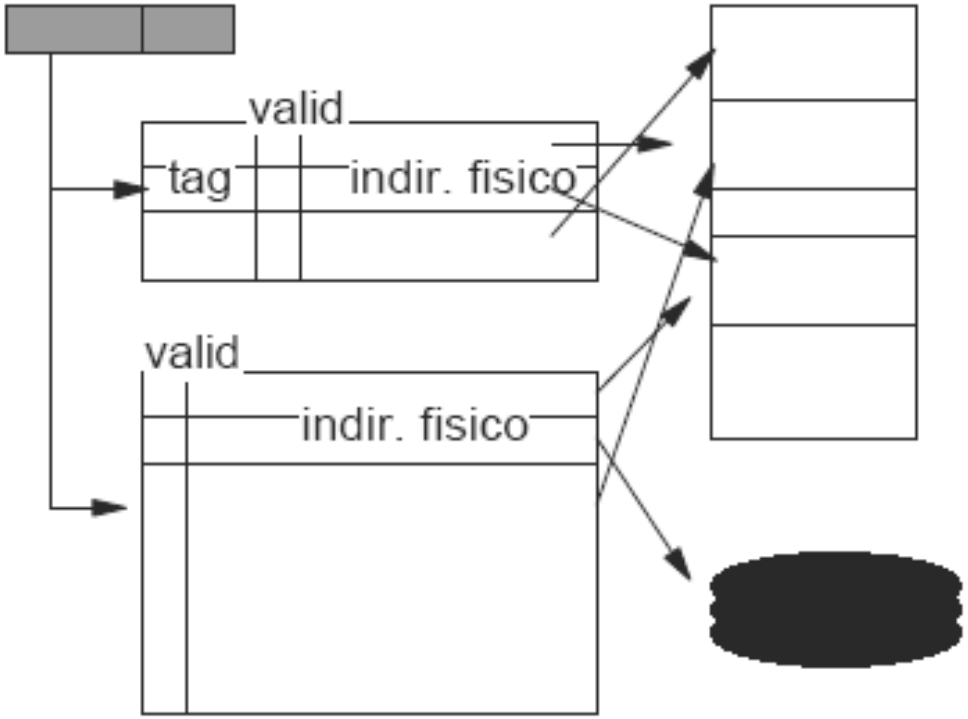
Memoria Virtuale (4/4)

- La CPU esegue il programma in cui è codificato **l'indirizzo virtuale** (che copre tutto lo spazio di indirizzamento possibile) deciso dal compilatore.
- La CPU prima di usare l'indirizzo per un accesso alla memoria (cache o memoria centrale che sia) traduce l'indirizzo virtuale in fisico. Viene gestito dalla MMU che accede alla memoria centrale ad un indirizzo dove è allocata la tabella delle pagine, usa (parte del)l'indirizzo virtuale come puntatore
- Se alla locazione indicata c'è l'indirizzo fisico (bit di stato valid) lo usa per accedere alla memoria
- Se non c'è l'indirizzo fisico corrispondente inizia un page fault che chiama il SO il quale rimpiazza la pagina da memoria secondaria a memoria centrale e scrive l'indirizzo fisico sulla tabella delle pagine
- Due accessi. Accesso in memoria per la tabella e accesso per il dato

- La tabella delle pagine può essere troppo lunga (ad esempio indirizzo a 32 bit con pagine di 4Kbyte, tabella delle pagine contiene 2^{20} indirizzi), occupa troppa memoria (1Mx4B=4MB). Quindi un indirizzamento ad una tabella che punta alle tabelle delle pagine (due livelli di tabella delle pagine ognuno da 2^{10} indirizzi)→ tre accessi in memoria!! (TROPPO LENTO)

TLB

- Per rendere più veloce l'accesso alle pagine si usa la **TLB** (Translation Lookaside Buffer): una cache della tabella delle pagine
- Se il tag è nel TLB, viene preso l'indirizzo della pagina direttamente, altrimenti si va alla tabella delle pagine in memoria che può puntare alla memoria centrale o di massa (se page fault)



Segmentazione con paginazione