Architettura degli Elaboratori

La Gerarchia di Memoria

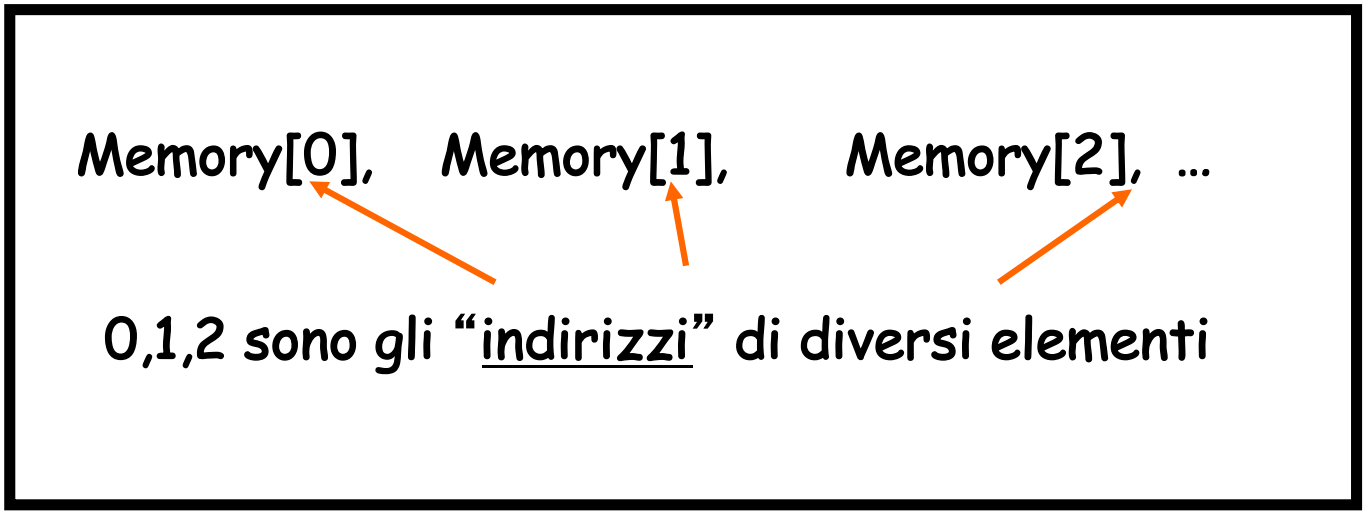
Barbara Masucci
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Punto della situazione

- Abbiamo studiato
  - Il processore MIPS e due diverse implementazioni per un sottoinsieme del suo IS
- Obiettivo di oggi
  - Analizzare alcune tecnologie per la realizzazione di diversi tipi di memoria
  - Comprendere il funzionamento della gerarchia di memoria utilizzata negli elaboratori moderni
  - Apprendere i principi base delle memorie cache

La Memoria

- La memoria è il luogo dove vengono tenuti i programmi in esecuzione e i dati di cui essi necessitano
  - Possiamo immaginare la memoria come un grande vettore unidimensionale di "celle" in sequenza
  - Ogni elemento di memoria ha un indirizzo attraverso il quale si può accedere ad esso



Parametri

La memoria è caratterizzata a diversi parametri, in particolare

- Dimensione o capacità
  Misurata in multipli di byte, indica la quantità di dati memorizzabili
- Velocità o tempo di accesso
  Indica l'intervallo di tempo (misurato in nanosecondi) tra la richiesta del dato e il momento in cui viene reso disponibile
- Costo
  Dipende dal volume di acquisto (maggiore è il volume di acquisto, minore è il costo per bit)

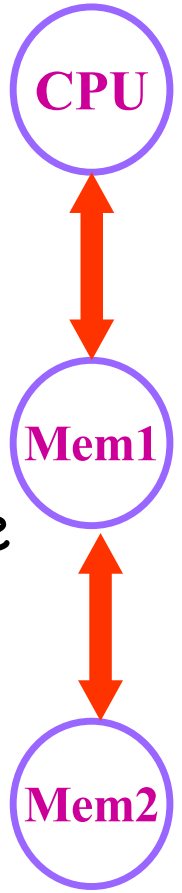
Idealmente la memoria dovrebbe avere
grande dimensione, elevata velocità, minimo costo

Tipi di accesso

- Nelle memorie ad accesso casuale, il tempo di accesso ai dati è indipendente dalla posizione in cui essi sono memorizzati
  - Tipico delle memorie a semiconduttori
  - Esempio: la memoria principale (RAM)
- Nelle memorie ad accesso sequenziale, il tempo di accesso ai dati dipende dalla posizione in cui essi sono memorizzati
  - Tipico delle memorie magnetiche
  - Esempio: la memoria secondaria (dischi e nastri)

Gerarchie di memoria

- Non è possibile avere un'unica memoria con tutte le caratteristiche ideali
- Idea: organizzare una gerarchia in cui
  - Le memorie piccole, più veloci (e costose) sono poste ai livelli alti, vicino al processore
  - Le memorie ampie, più lente (e meno costose) sono poste ai livelli più bassi
- L'obiettivo è quello di creare l'illusione di avere a disposizione una memoria che sia grande veloce ed economica



Diversi Livelli, Diverse Tecnologie

- SRAM: Static Random Access Memory
  - Statica (l'informazione memorizzata è permanente fino a quando l'alimentazione è attiva)
  - Costosa (ciascun bit è memorizzato mediante 6 oppure 8 transistor)
  - Tempo di accesso molto rapido, infatti è usata per i livelli più vicini alla CPU
- DRAM: Dynamic Random Access Memory
  - Dinamica (richiede un refresh periodico, cioè il dato va letto e riscritto continuamente)
  - Economica (ciascun bit è memorizzato mediante un solo transistor)
  - Tempo di accesso più lento, infatti è usata per la memoria principale

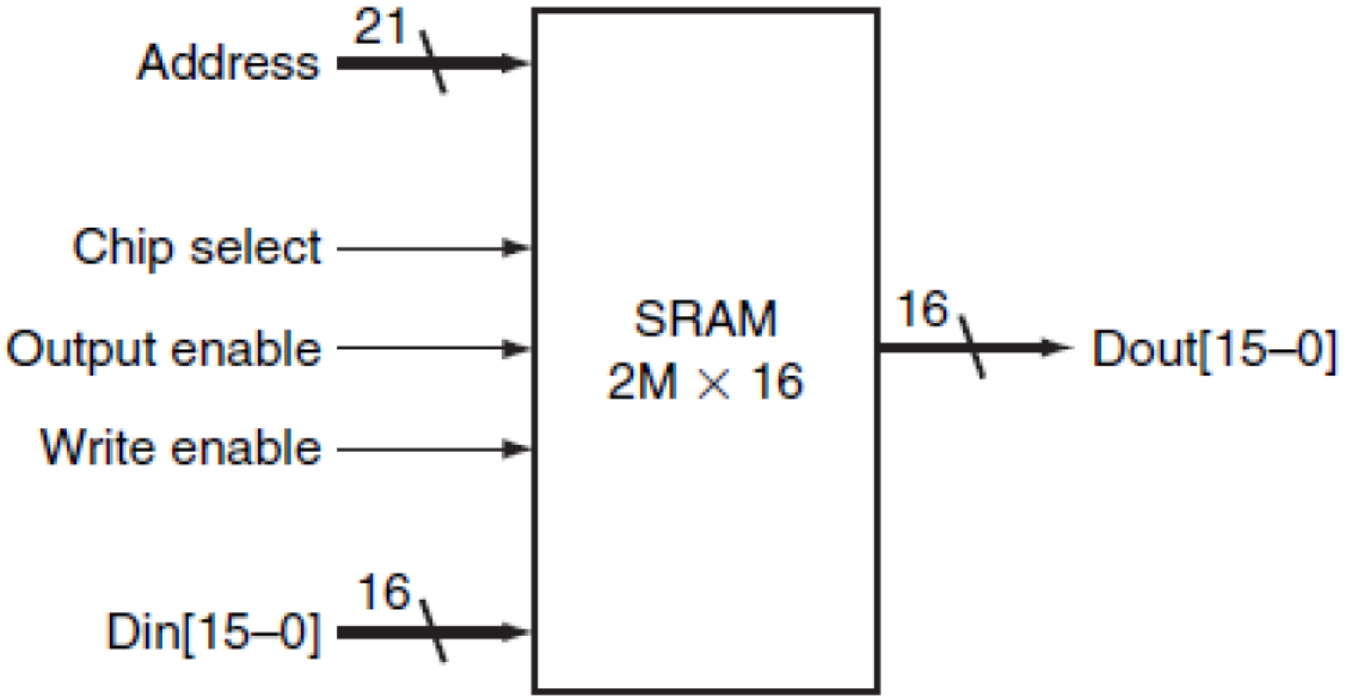
Memory technology	Typical access time	$ per GIB in 2012
SRAM semiconductor memory	0.5–2.5 ns	$500–$1000
DRAM semiconductor memory	50–70 ns	$10–$20

SRAM

- Una 2^n X m SRAM è un array di 2^n celle, ciascuna di m bit
  - Si può accedere alle celle in lettura specificandone l'indirizzo
  - Si può accedere alle celle in scrittura specificandone l'indirizzo e il dato da scrivere
- Gli input a una 2^n X m SRAM sono
  - Indirizzo cella a cui accedere (n bit)
  - Tre segnali di controllo di 1 bit
    - Chip select (deve essere asserito per poter leggere o scrivere)
    - Output enable (deve essere asserito per poter leggere)
    - Write enable (deve essere asserito per poter scrivere)
  - Dato da scrivere (m bit), se l'accesso è in scrittura
- L'output di una 2^n X m SRAM è il dato letto (m bit)

SRAM

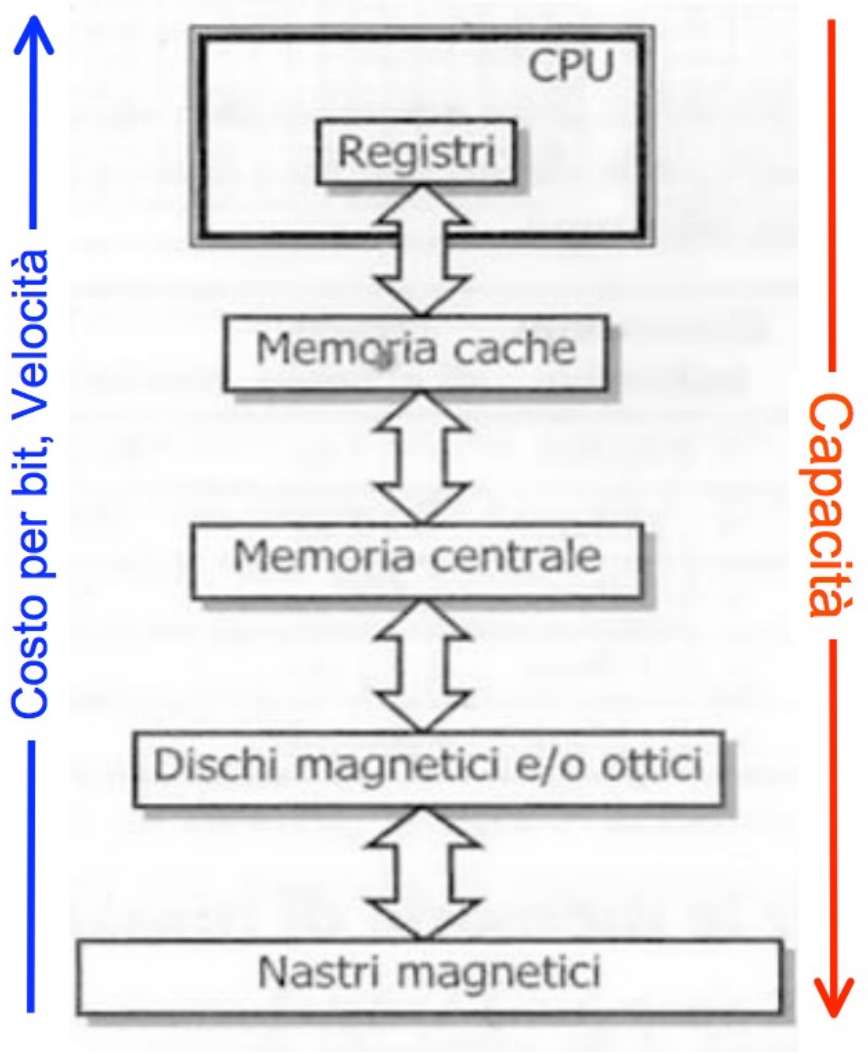
- Ad esempio, una 2M X 16 SRAM ha $2M = 2 \times 2^{20} = 2^{21}$ celle, ciascuna di 16 bit
  - Ogni cella può essere indirizzata specificando 21 bit
  - Il dato in output è di 16 bit



SRAM

- Una 2^n X m SRAM è costituita da un decoder n a 2^n connesso a 2^n X m flip-flop di tipo D
- Tramite il decoder, a partire dall'indirizzo di n bit si seleziona una delle 2^n celle, ciascuna costituita da m flip-flop di tipo D
- Ciascun flip-flop consente la memorizzazione di 1 bit

Gerarchia di memoria



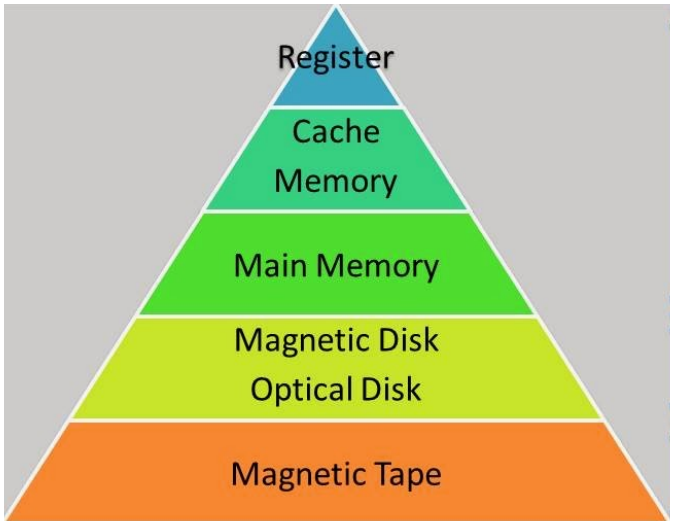
Memoria cache

- La cache è una memoria veloce di piccole dimensioni posta fra la CPU e la memoria principale
- La tecnologia utilizzata per la costruzione della memoria cache è la SRAM
- La cache e la memoria principale formano una gerarchia di memoria

Gerarchia di memoria

- La gestione dei livelli avviene in base al principio di località:
  - I dati utilizzati più spesso sono posti in memorie più veloci e più vicine al processore
  - I dati utilizzati più raramente sono posti in memorie più lente e più lontane dal processore

- Allocazione dinamica
  - Spostamento automatico dei dati tra i livelli
  - Canali di comunicazione veloci fra i livelli



Principio di località

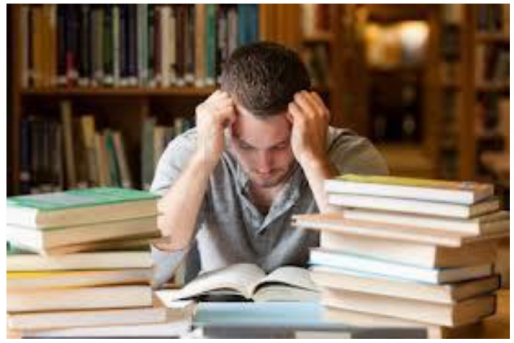
- Esistono due diversi tipi di località
  - Località temporale
    - Se un elemento di memoria (dato o istruzione) è stato letto, tenderà ad essere letto nuovamente entro breve tempo
    - Caso tipico: le istruzioni e i dati entro un ciclo saranno letti ripetutamente
  - Località spaziale
    - Se un elemento di memoria (dato o istruzione) è stato letto, gli elementi i cui indirizzi sono vicini tenderanno ad essere letti entro breve tempo
    - Caso tipico: gli accessi agli elementi di un array

La biblioteca: Analogia

- Uno studente deve scrivere una relazione
  - Svolge il ruolo del processore

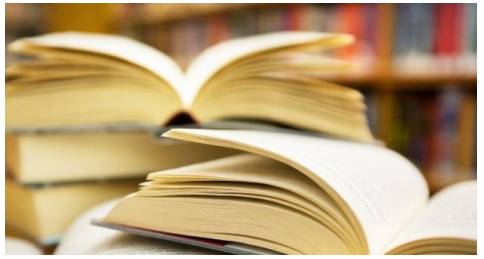
- La biblioteca è il disco
  - Spazio di memoria senza limiti
  - Per trovare il libro che serve ci vuole tempo

- Il tavolo di studio è la memoria
  - Ha meno spazio…: se il tavolo è pieno lo studente deve restituire un libro
  - Trovare il libro da usare è molto più veloce, se si trova già sul tavolo



La biblioteca: Analogia

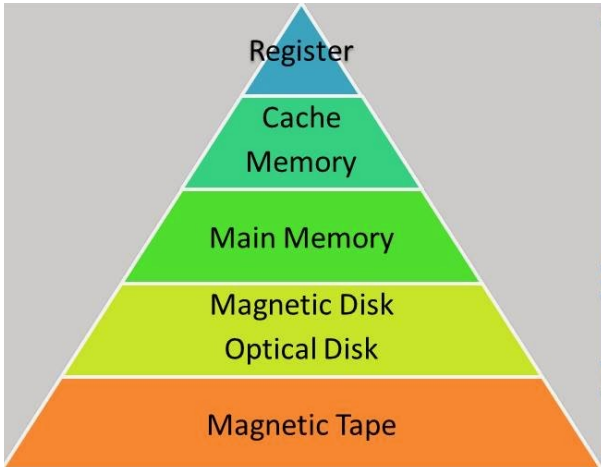
- I libri aperti sul tavolo sono la cache
  - Capacità ancora più piccola: si possono avere pochi libri aperti sul tavolo…
    - Quando sono troppi bisogna chiuderne uno…
  - Molto più veloce nel recuperare i dati



- Per creare l'illusione dell'intera biblioteca "aperta" sul tavolo si possono
  - Tenere aperti sul tavolo i libri usati più recentemente
    - E' probabile doverli riutilizzare
  - Tenere i libri sul tavolo il più a lungo possibile…
    - Restituirli fa perdere tempo

Gerarchia di memoria

- Più il livello della gerarchia è vicino al processore
  - Minore è la sua capacità di memoria
  - Maggiore è la velocità di accesso ai suoi dati
- Ciascun livello contiene
  - Un sottoinsieme dei dati del livello sottostante
    - Di solito quelli usati più recentemente
  - Tutti i dati contenuti nei livelli soprastanti
    - Di conseguenza, il livello più basso (il disco) contiene tutti i dati
- Le informazioni vengono di volta in volta copiate tra livelli adiacenti della gerarchia



Livelli nella gerarchia

- Registri
  - Costituiscono la memoria più veloce, interna al processore
  - Memorizzano temporaneamente dati o istruzioni
  - Sono gestiti dal compilatore

- Cache di primo livello
  - Si trova sullo stesso chip del processore (per cui esso può accedervi direttamente)
  - Utilizza la tecnologia SRAM
  - E' trasparente al programmatore e al compilatore
  - I trasferimenti dalle memorie di livello inferiore sono completamente gestiti dall'hardware
  - Possono esserci cache separate per istruzioni (I-cache) e dati (D-cache), o una sola cache per entrambi

Livelli nella gerarchia

- Cache di secondo (e terzo) livello
  - Quando esiste, può essere sia sullo stesso chip del processore, sia su un chip separato
  - Utilizza la tecnologia SRAM
  - E' trasparente al programmatore e al compilatore
  - I trasferimenti dalle memorie di livello inferiore sono completamente gestiti dall'hardware

- RAM
  - Utilizza la tecnologia DRAM
  - I trasferimenti dalle memorie di livello inferiore sono gestiti dal sistema operativo (memoria virtuale) e dal programmatore

Migrazione delle informazioni

- Consideriamo due livelli adiacenti della gerarchia
  - Livello superiore
  - Livello inferiore
- Blocco o linea
  - La più piccolo quantità di informazione che può essere trasferita tra due livelli adiacenti nella gerarchia
  - Analogia con la biblioteca: blocco = libro
- Hit (successo)
  - Se il dato richiesto dal processore è contenuto in uno dei blocchi presenti nel livello superiore
  - Analogia con la biblioteca: l'informazione cercata dallo studente è presente in uno dei libri a disposizione sulla scrivania

Migrazione delle informazioni

- Miss (fallimento)
  - Se il dato richiesto dal processore non è contenuto in uno dei blocchi presenti nel livello superiore
  - In tal caso bisogna accedere al livello inferiore della gerarchia per recuperare il blocco contenente il dato richiesto e copiarlo nel livello superiore
- Analogia con la biblioteca
  - L'informazione cercata dallo studente non è presente in nessuno dei libri sulla scrivania
  - In tal caso lo studente deve cercare tra gli scaffali della biblioteca per recuperare il libro contenente l'informazione di cui ha bisogno
  - Se lo studente trova il libro, lo porta sulla scrivania

Migrazione delle informazioni

- Tempo di Hit
  - Tempo necessario per trovare il dato richiesto nel livello superiore
- Hit rate
  - Frequenza di successo: frazione degli accessi in cui troviamo il dato nel livello superiore
- Miss rate
  - Frequenza di insuccesso: frazione degli accessi in cui non troviamo il dato nel livello superiore
- Penalità di miss
  - Tempo necessario per trovare il dato se questo non è presente nel livello superiore

Migrazione delle informazioni

- L'hit rate viene usato come indice delle prestazioni della gerarchia di memoria
- Motivo:
  - Il livello superiore della gerarchia è più piccolo e costruito con componenti più veloci
  - Quindi il tempo di hit sarà molto inferiore alla penalità di miss
  - Analogia con la biblioteca
    - Il tempo richiesto per esaminare i libri sulla scrivania (tempo di hit) è molto inferiore di quello impiegato per alzarsi e andare a prelevare un altro libro dagli scaffali della biblioteca (penalità di miss)

Memoria Cache: Un esempio semplice

- Consideriamo una cache molto semplice in cui
  - Il processore carica una sola parola per volta
  - I blocchi sono costituiti da una sola parola
- Supponiamo che ad un certo punto il processore richieda la parola X_n, che non è presente in cache
  - Prima della richiesta la cache contiene le parole $X_1,\ldots,X_{n-1}$ utilizzate più recentemente
  - Si produce una miss e la parola X_n viene prelevata dal livello inferiore della gerarchia e portato nella cache

Memoria Cache

- Domande:
  - Come facciamo a capire se un dato richiesto è presente nella memoria cache?
  - Se è presente, come facciamo a trovarlo?

- Le risposte a queste due domande sono collegate
  - Se ogni parola può essere scritta in una sola posizione della cache, allora sappiamo dove trovarla (se è presente)
  - Ci basta definire una corrispondenza tra l'indirizzo in memoria della parola e la sua locazione nella cache

Cache ad indirizzamento diretto

- Ad ogni indirizzo della memoria corrisponde in modo univoco una locazione della cache
- La corrispondenza è molto semplice:

  indirizzo nella cache = indirizzo della memoria modulo N_B

  dove N_B indica il numero di blocchi nella cache

Cache ad indirizzamento diretto

- Se N_B è una potenza di 2, basta considerare i $\log_2 N_B$ bit meno significativi dell'indirizzo del blocco in memoria

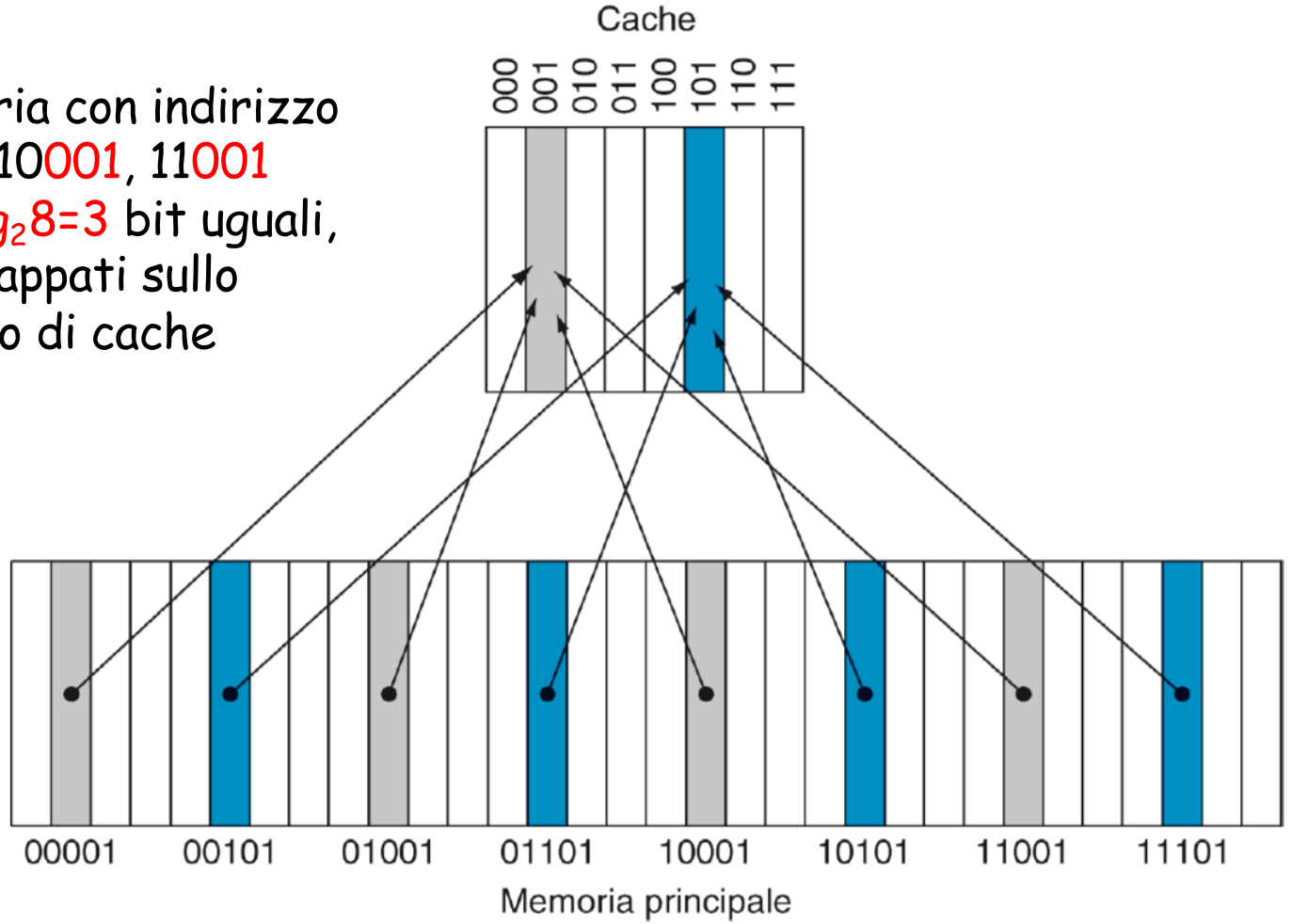
- Esempio:
  - N_B=2, $\log_2 N_B$=1, 10010 mod 2=0, 11011 mod 2=1
  - N_B=4, $\log_2 N_B$=2, 10010 mod 4=10, 11011 mod 4=11
  - N_B=8, $\log_2 N_B$=3, 10010 mod 8=010, 11011 mod 8=011

- Quindi, tutti i blocchi della memoria che hanno i $\log_2 N_B$ meno significativi dell'indirizzo uguali vengono mappati sullo stesso blocco di cache

Cache ad indirizzamento diretto

Esempio di cache con 8 blocchi ad indirizzamento diretto per una memoria con 32 locazioni

I blocchi di memoria con indirizzo
00001, 01001, 10001, 11001
hanno gli ultimi $\log_2 8=3$ bit uguali, quindi sono mappati sullo stesso blocco di cache



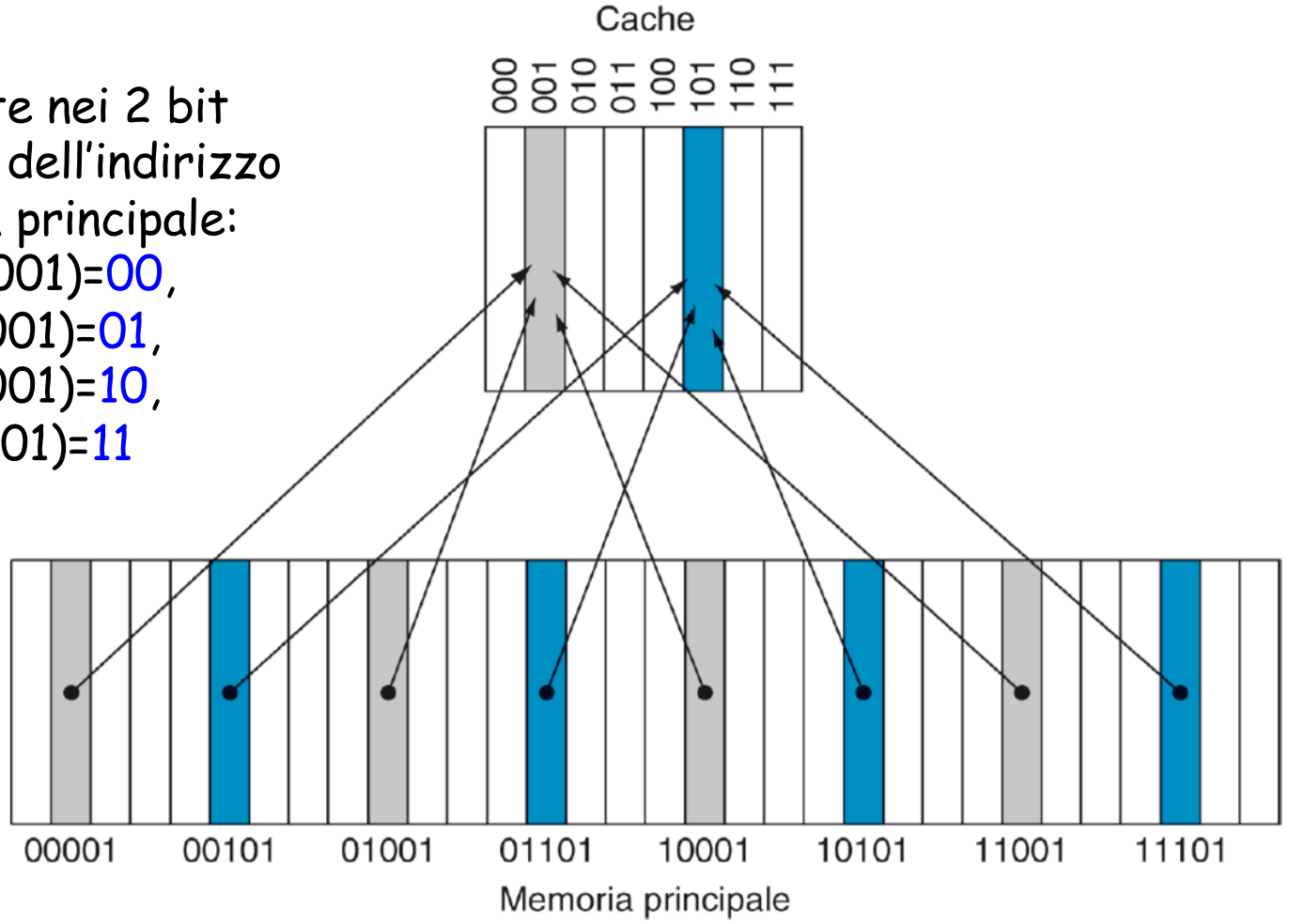
Cache ad indirizzamento diretto

- Problema:
  - Poichè ogni blocco della cache può contenere parole provenienti da diverse locazioni di memoria, come facciamo a capire se il dato presente nella cache corrisponde a quello che cerchiamo?
- Soluzione:
  - Si ricorre a un campo, detto tag, che contiene un'informazione che consente di verificare se un dato presente nella cache corrisponde ad una certa locazione di memoria

Cache ad indirizzamento diretto

Esempio di cache con 8 blocchi ad indirizzamento diretto per una memoria con 32 locazioni

Il tag consiste nei 2 bit più significativi dell'indirizzo nella memoria principale:
Tag(00001)=00,
Tag(01001)=01,
Tag(10001)=10,
Tag(11001)=11



Validità

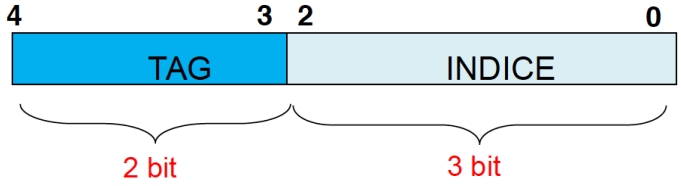
- Problema:
  - Come capire se il dato contenuto in un blocco della cache è valido oppure no?
- Soluzione:
  - Si ricorre a un campo, detto validità
    - Se il bit di validità è 1, il dato può essere letto
    - Se il bit di validità è 0, il dato viene ignorato
    - All'avvio, la cache è vuota per cui tutti i bit di validità sono impostati a 0

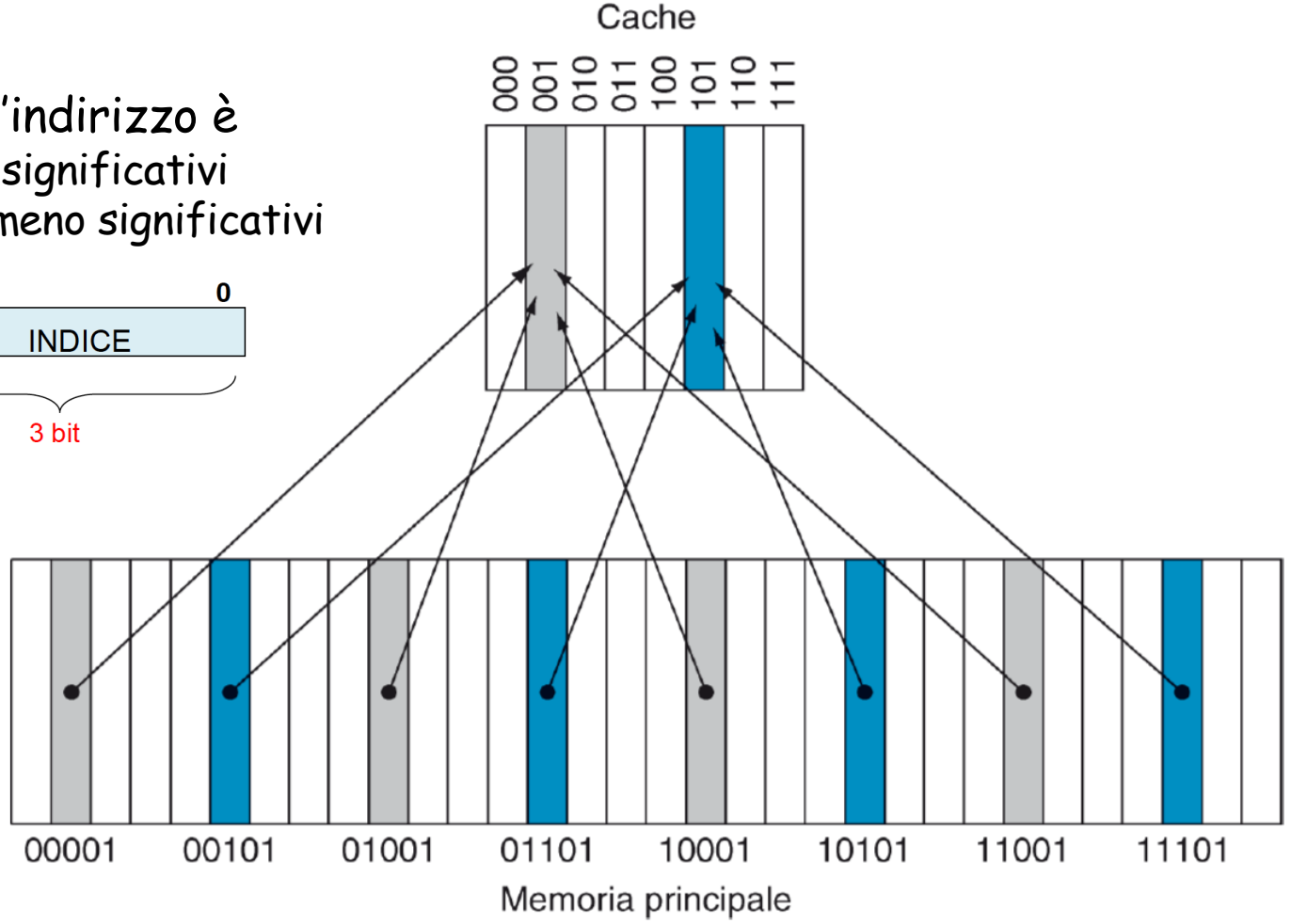
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})

Cache ad indirizzamento diretto

Esempio di cache con 8 blocchi ad indirizzamento diretto per una memoria con 32 locazioni

- Indirizzi a 5 bit
- La struttura dell'indirizzo è
  - Tag: 2 bit più significativi
  - Indice: 3 bit meno significativi





Cache

- Struttura:
  - Ciascun blocco, o linea, della cache è costituita da 4 campi
    - Indice
    - Bit di validità
    - Tag
    - Dato

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})

Accesso alla cache

- Vediamo come cambia il contenuto della cache
  - Nello stato iniziale, la cache è vuota

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

- Viene richiesta la parola all'indirizzo $(10110)_2$: MISS
  - Indice = 110
  - Tag = 10

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Esempio

- Viene richiesta la parola all'indirizzo $(11010)_2$: MISS
  - Indice = 010
  - Tag = 11

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

- Viene richiesta la parola all'indirizzo $(10000)_2$: MISS
  - Indice = 000
  - Tag = 10

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Esempio

- Viene richiesta la parola all'indirizzo $(00011)_2$: MISS
  - Indice = 011
  - Tag = 00

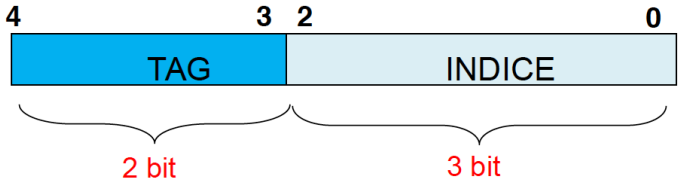
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

- Viene richiesta la parola all'indirizzo $(10010)_2$: MISS
  - Indice = 010
  - Tag = 10
  - La parola 10010 sostituisce la precedente nel blocco con indice 010

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Hit!

Se ricevo un indirizzo dalla memoria



Avrò una hit (dato presente nella cache) se, nella linea di cache INDICE

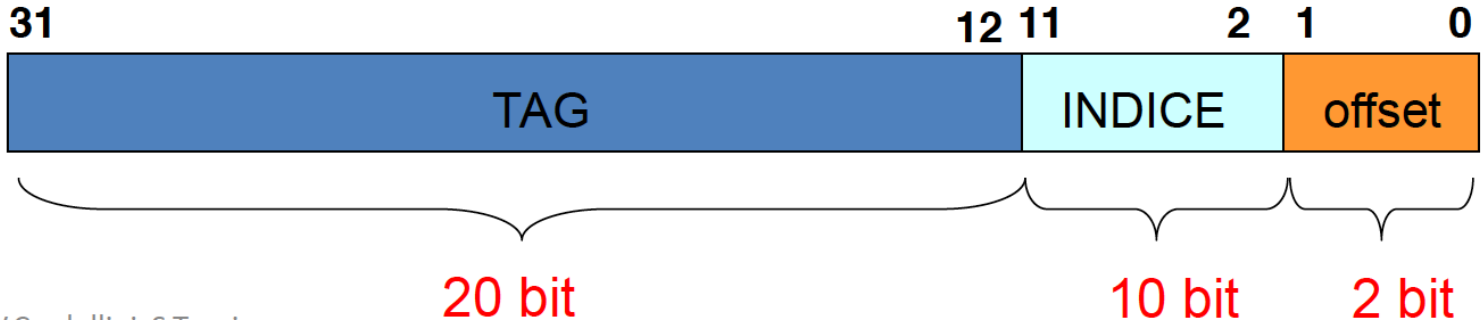
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})

Il tag coincide con TAG e V=1

Un esempio più realistico (MIPS)

- Indirizzi a 32 bit (2^{32} locazioni in memoria centrale)
  - Tag: 20 bit più significativi
  - Indice: successivi 10 bit
  - Offset: ultimi 2 bit (non li consideriamo)

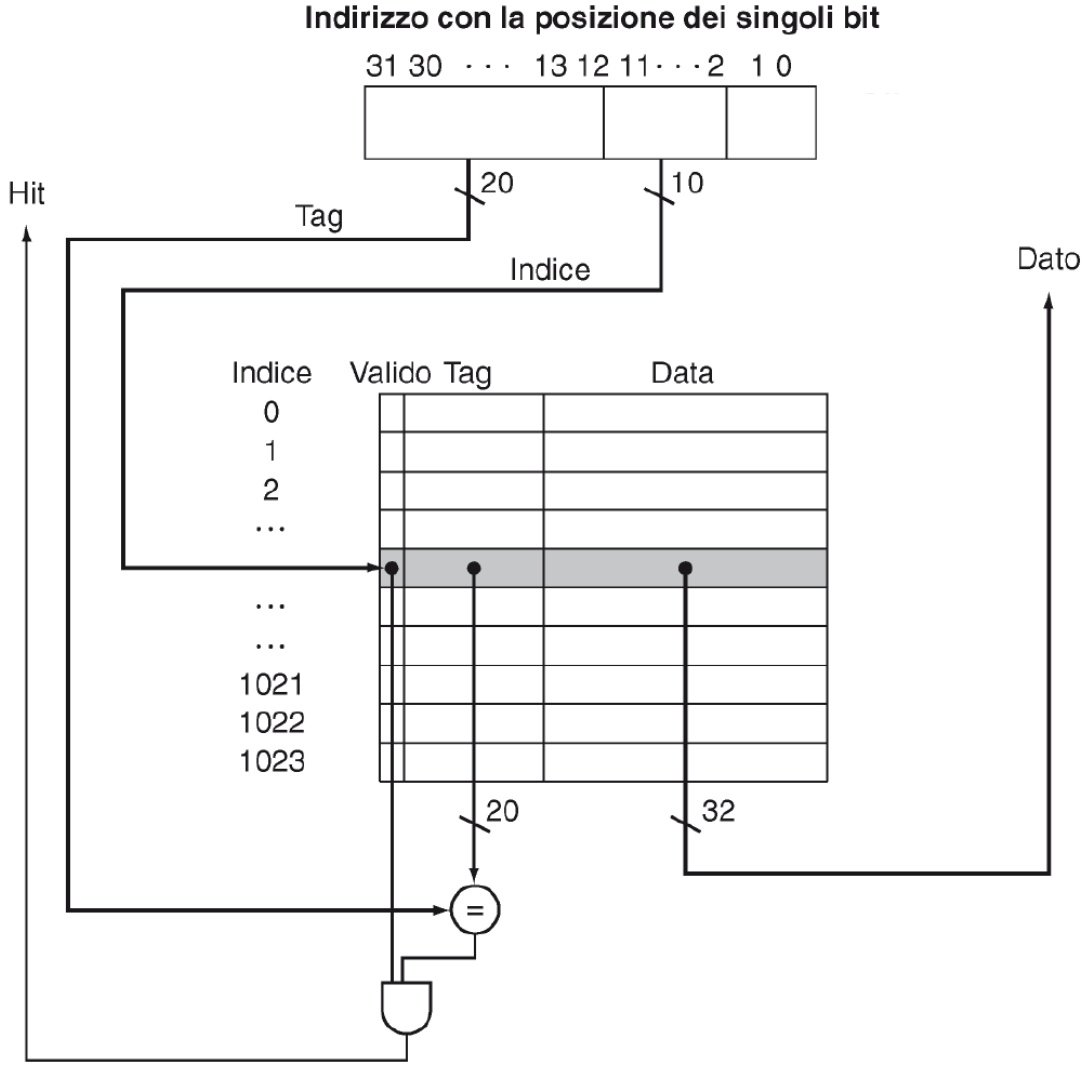
- Cache con 2^{10} blocchi
- Dimensione del blocco: 32 bit (4 byte)
- Dimensione cache = #blocchi X dimensione blocco
  = 2^{10} X 4 byte = 4 KB



Un esempio più realistico (MIPS)

Accesso in cache:

- Si confronta il tag dell'indirizzo con il tag della linea di cache individuata tramite l'indice
- Si controlla il bit di validità
- Viene segnalato l'hit al processore e trasferito il dato



Riepilogo e riferimenti

- Abbiamo visto come è fatta la memoria
  - Introduzione alla memoria: [PH] parr. 1.4, 5.1
  - Le tecnologie delle memorie: [PH] par. 5.2, [PH] Appendice B.9
  - Principi base delle memorie cache: [PH] par. 5.3
    (escluso il caso di blocchi composti da più parole)

Il corso termina qui...

Il corso termina qui...

- Vi aspetto alla prova scritta
  - Seconda prova intercorso
    - 21 Dicembre ore 9:00, aule P3, P4
  - Uno dei prossimi appelli (due date a Gennaio, una a Febbraio)
    - Ciascuna prova scritta sarà preceduta da una lezione di tutorato

Seconda prova

- Sono ammessi a partecipare tutti gli studenti che hanno partecipato alla prima prova
  - Non è necessaria la prenotazione, ma comunicatemi via e-mail entro il 18 Dicembre se non volete partecipare
- Le prove intercorso si intendono superate se il voto ottenuto dalla media delle due prove è almeno 18/30
- Dopo la pubblicazione dei risultati, gli studenti ammessi devono comunicarmi se
  - Intendono accettare il voto proposto
  - Intendono sostenere la prova orale
  - Intendono ripetere lo scritto in uno degli appelli fissati

Prenotazione ESSE3

- Per il completamento dell'esame è necessaria la prenotazione su ESSE3 in uno degli appelli fissati
  - Per la registrazione del voto proposto dopo il superamento delle prove, prenotare l'appello dell'11 Gennaio
  - Per sostenere l'orale a partire dal voto proposto, prenotare l'appello dell'11 Gennaio