Architettura degli Elaboratori

Informatica per la Comunicazione Digitale

Università degli Studi di Milano

Lezione 11:

# La memoria centrale (RAM)
## e gerarchie di memoria

Marco Tarini

1

## Memoria Centrale, o RAM

- Il **Banco dei Registri** contiene le poche decine di registri su cui il programma in esecuzione sta lavorando *in una certa fase*
  - Come abbiamo visto, il banco di registri fornisce gli operatori alla **ALU**, e ne memorizza i risultati
- Il grosso dei dati (e, vedremo, anche il programma da eseguire!) è invece memorizzato nella **memoria centrale** dell'elaboratore (molto più capiente del register file)
  - Anche se un banco di memoria è funzionalmente simile ad un Register file, la sua capienza molto maggiore (il numero molto superiore di parole da memorizzare) comporta una serie di modifiche progettuali anche profonde



2

## Memorie molto più capienti? La main memory (o RAM o memoria centrale)

- «**Capienza**» della memoria (o dimensione) = quantità di dati che può contenere (espressa in bits, bytes, etc)
- In quasi tutti i contesti, ci interessa di lavorare su migliaia, milioni, miliardi, o oltre, di parole (*word*) di dati.
- La struttura vista del Register File non si presta neanche lontanamente a dimensioni di questo tipo.
  - In altri termini: quest'architettura non è «***scalabile***», non «***scala***» facilmente a dimensioni molto maggiori di quelle viste
- Di seguito, vedremo come dotarci di una memoria molto più capiente: la **RAM**
  - Per ottenere questo scopo, faremo molti compromessi, e vedremo alcuni dispositivi a basso livello diversi
  - Il **Register File** rimane comunque la memoria più veloce e ottimizzata, adatta ad essere il «banco di lavoro» della ALU.
- Apparte la capienza (molto maggiore), la RAM è concettualmente simile ad un register file:
  - un insieme di parole memorizzate (word)
  - accedibili individualmente, in scrittura o in lettura, fornendo il loro indice (detto l'**indirizzo** di quella parola in memoria)



3

## Memoria Centrale (Main Memory) o RAM

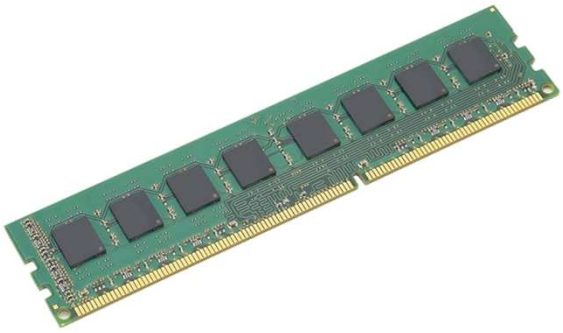
- Un blocco funzionale capace di memorizzare un numero $m$ di parole di un numero $n$ di bit ciascuna
  - quindi il banco di registri è un caso particolare di un banco di memoria… con $m$ molto piccolo
- Capacità totale (in bits): $m \times n$
  - Più spesso, espressa in byte (basta dividere per 8)
- **Spazio di Indirizzamento**:
  serviranno $k$ bit per esprimere un indirizzo, con
  - minimo $k$ tale che $2^k \geq m$
  - cioè $k = \lceil \log_2 n \rceil$
- Due operazioni sono eseguibili sul banco di memoria:
  - **lettura**, o **load**, ovvero si legge un word di $m$ bit (memorizzato)
  - **scrittura**, o **store**, ovvero si memorizzano $m$ bit in una parola



4

## Memoria RAM: tipicamente realizzata con tanti memory-bank, ciascuno su un chip

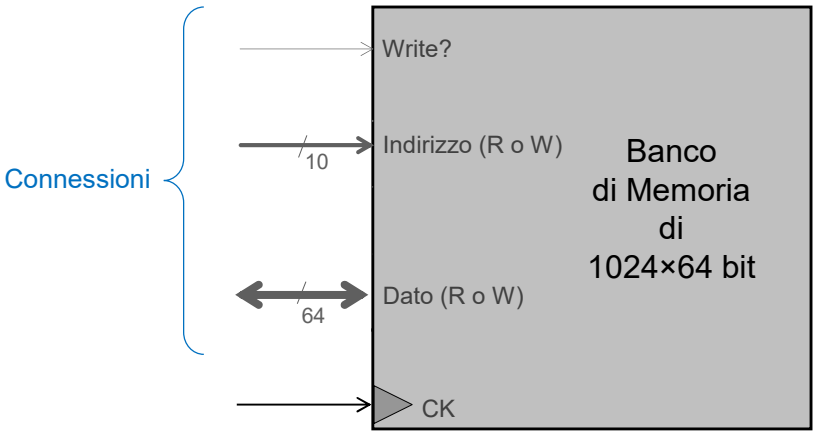
- La RAM è spesso ralizzata da diversi “memory bank” (banchi di memoria), ciascuno realizzato su un chip seaparato
- La capienza di ogni memory back è limitata da quanti componenti riesco ad alloggiare in un chip
- Per semplicità, considereremo la memoria RAM (di massa) come composta di un solo «memory bank» molto capiente.





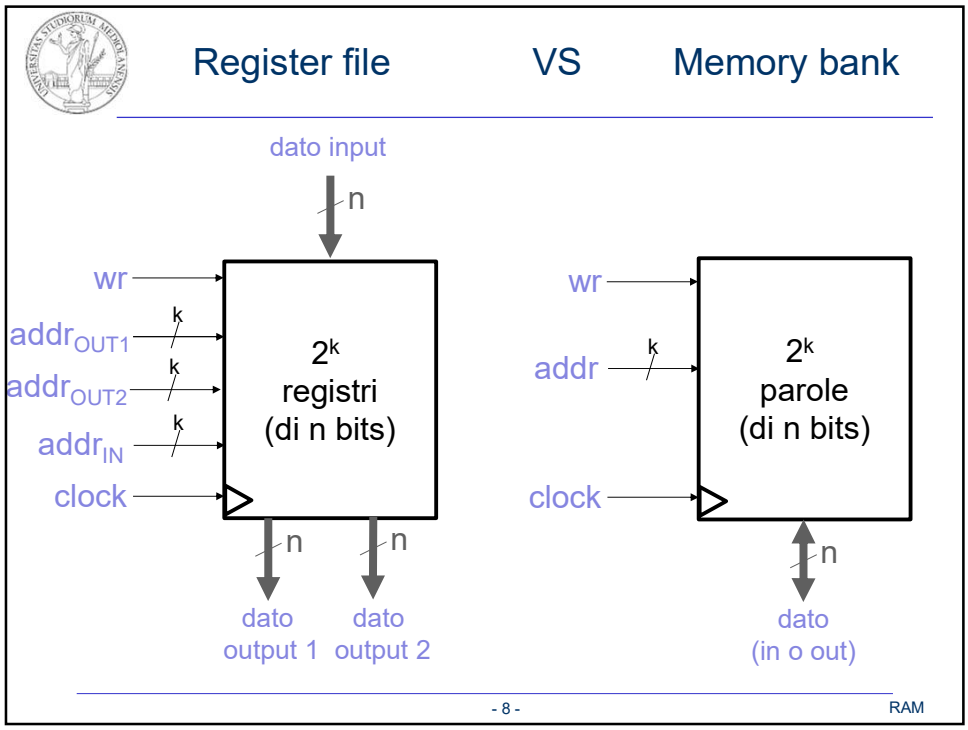
5

## Es: un componente di memoria da 1024×64 bit con 1 solo bus

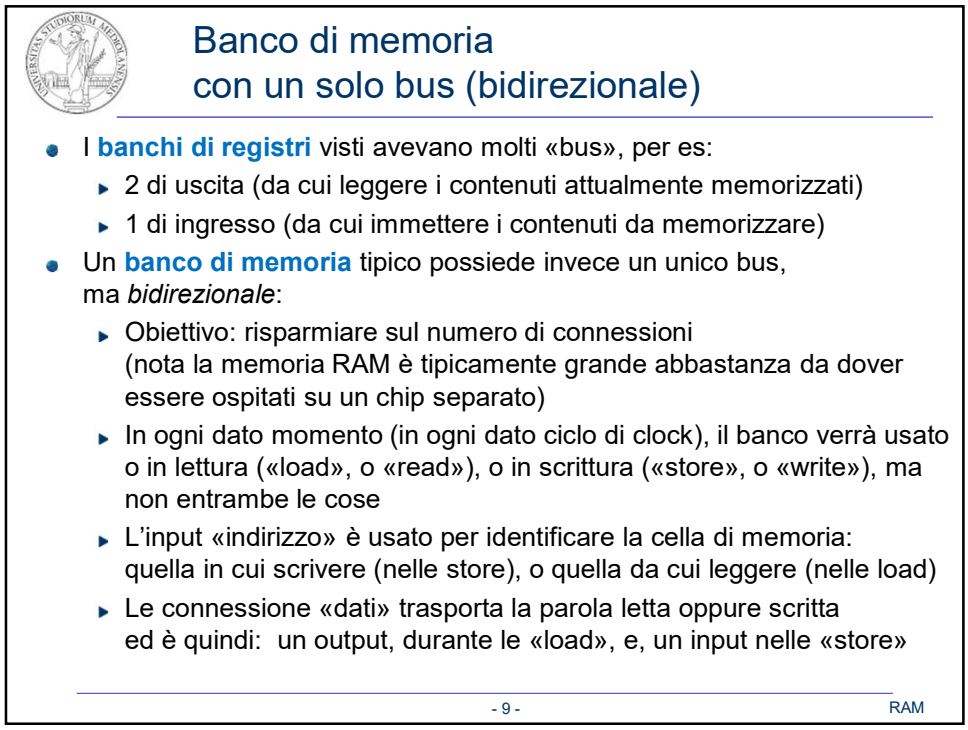





7

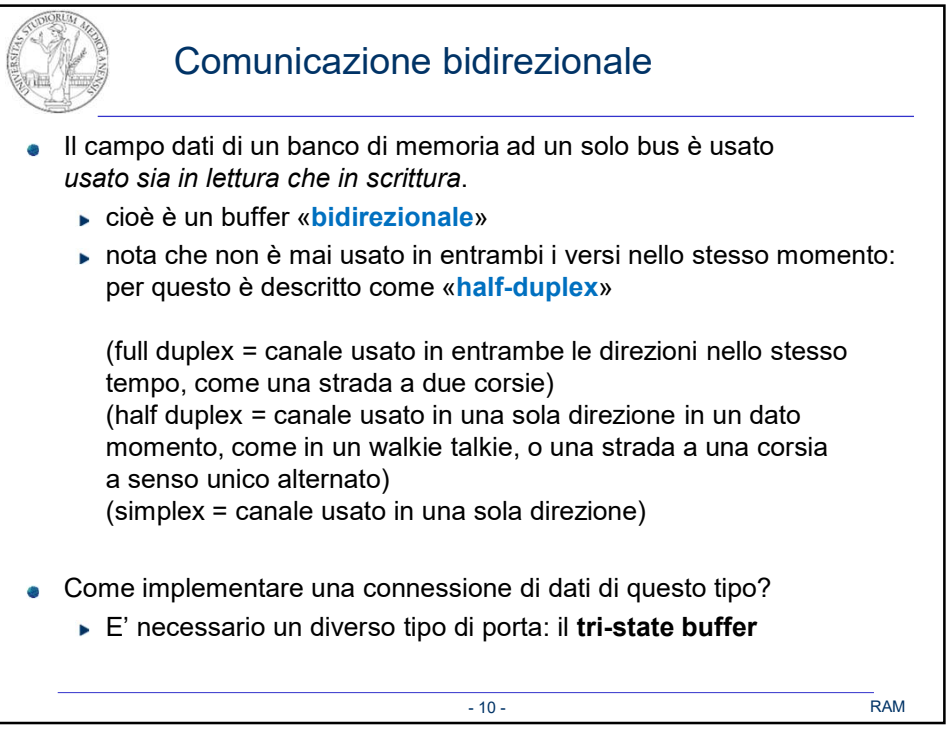



8




- I **banchi di registri** visti avevano molti «bus», per es:
  - 2 di uscita (da cui leggere i contenuti attualmente memorizzati)
  - 1 di ingresso (da cui immettere i contenuti da memorizzare)
- Un **banco di memoria** tipico possiede invece un unico bus, ma *bidirezionale*:
  - Obiettivo: risparmiare sul numero di connessioni (nota la memoria RAM è tipicamente grande abbastanza da dover essere ospitati su un chip separato)
  - In ogni dato momento (in ogni dato ciclo di clock), il banco verrà usato o in lettura («load», o «read»), o in scrittura («store», o «write»), ma non entrambe le cose
  - L'input «indirizzo» è usato per identificare la cella di memoria: quella in cui scrivere (nelle store), o quella da cui leggere (nelle load)
  - Le connessione «dati» trasporta la parola letta oppure scritta ed è quindi: un output, durante le «load», e, un input nelle «store»

9



## Comunicazione bidirezionale

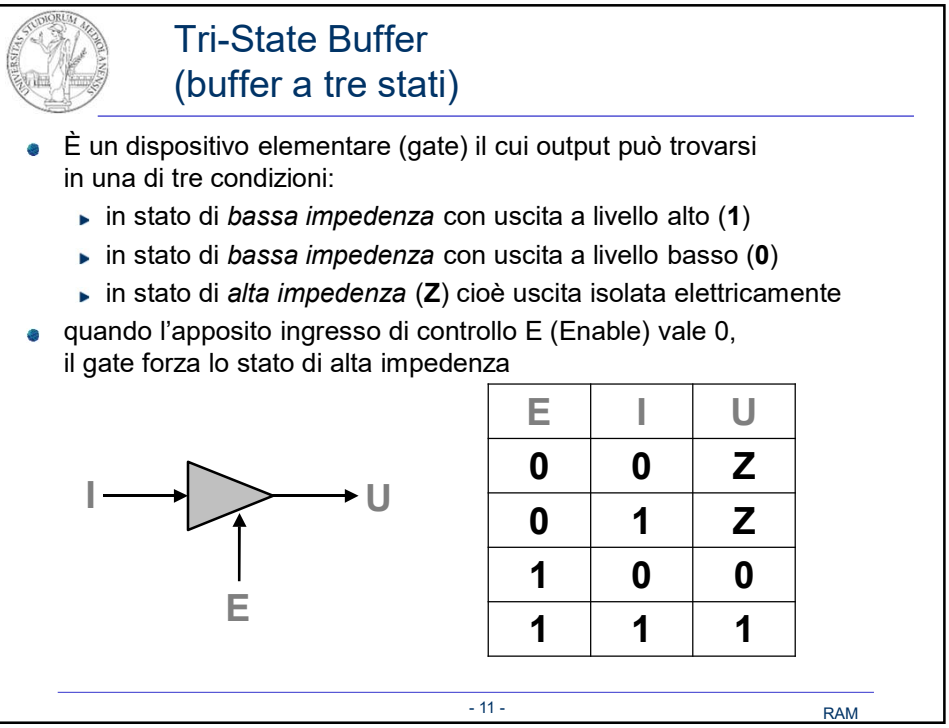
- Il campo dati di un banco di memoria ad un solo bus è usato *usato sia in lettura che in scrittura*.
  - cioè è un buffer «**bidirezionale**»
  - nota che non è mai usato in entrambi i versi nello stesso momento: per questo è descritto come «**half-duplex**»

    (full duplex = canale usato in entrambe le direzioni nello stesso tempo, come una strada a due corsie)
    (half duplex = canale usato in una sola direzione in un dato momento, come in un walkie talkie, o una strada a una corsia a senso unico alternato)
    (simplex = canale usato in una sola direzione)

- Come implementare una connessione di dati di questo tipo?
  - E' necessario un diverso tipo di porta: il **tri-state buffer**



10



## Tri-State Buffer (buffer a tre stati)

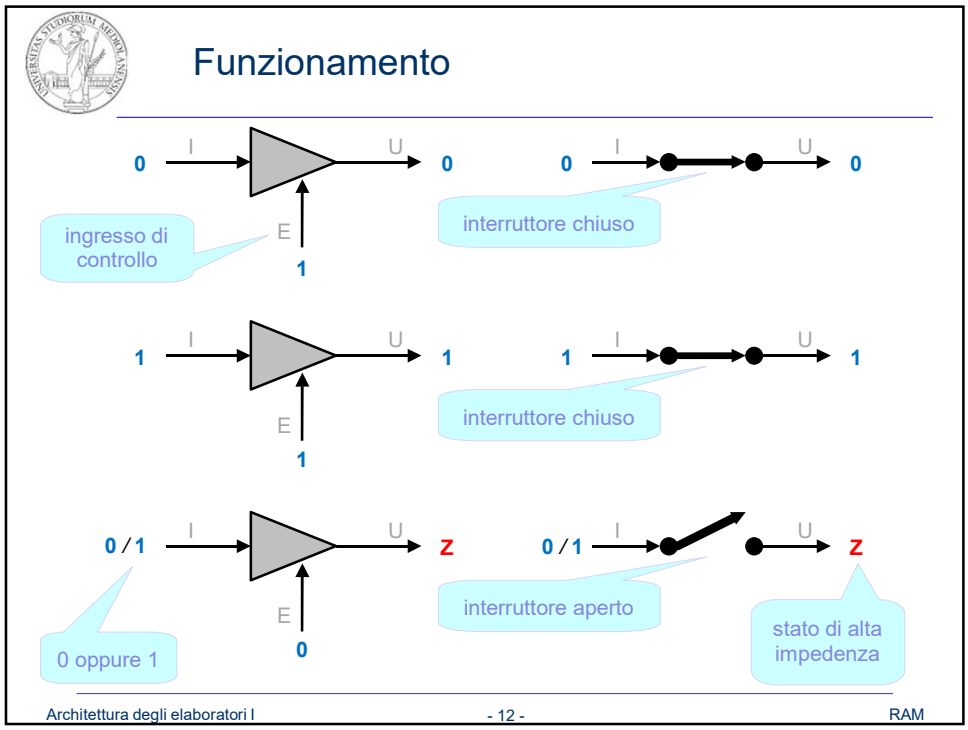
- È un dispositivo elementare (gate) il cui output può trovarsi in una di tre condizioni:
  - in stato di *bassa impedenza* con uscita a livello alto (**1**)
  - in stato di *bassa impedenza* con uscita a livello basso (**0**)
  - in stato di *alta impedenza* (**Z**) cioè uscita isolata elettricamente
- quando l'apposito ingresso di controllo E (Enable) vale 0, il gate forza lo stato di alta impedenza



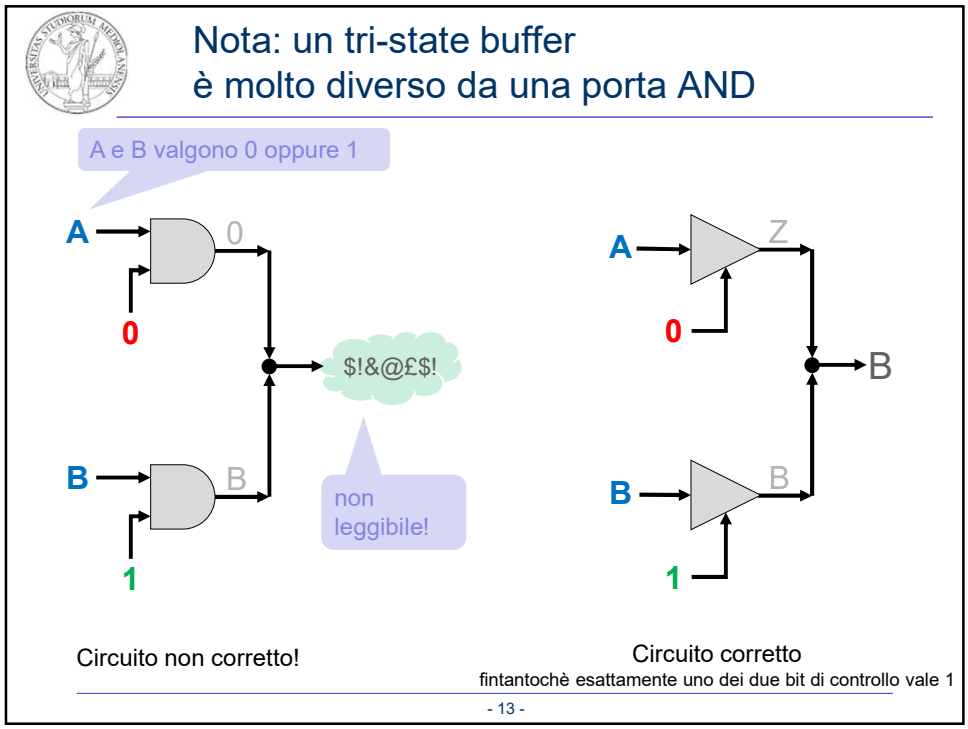

| E | I | U |
|---|---|---|
| 0 | 0 | Z |
| 0 | 1 | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



11


Funzionamento
0
I
U
0
E
1
ingresso di controllo
0
I
U
0
interruttore chiuso
1
I
U
1
E
1
1
I
U
1
interruttore chiuso
0 / 1
I
U
Z
E
0
0 oppure 1
0 / 1
I
U
Z
interruttore aperto
stato di alta impedenza
Architettura degli elaboratori I
- 12 -
RAM

12


Nota: un tri-state buffer
è molto diverso da una porta AND
A e B valgono 0 oppure 1
A
0
0
B
B
1
$!&@£$!
non leggibile!
Circuito non corretto!
A
Z
0
B
B
B
1
Circuito corretto
fintantochè esattamente uno dei due bit di controllo vale 1
- 13 -

13

## Nota: il valore «Z» (alta impedenza) è molto diverso da valore «0»

- 1: passa corrente a (relativamente) bassa tensione
- 0: passa corrente a (relativamente) alta tensione
- Z: non passa (quasi) corrente (è isolato)

Per misurare la tensione, devo avere corrente

Conseguenze:

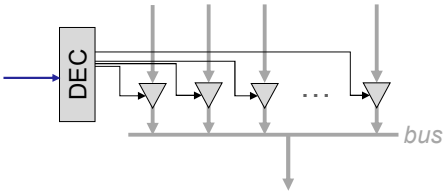
- se faccio confluire due cavi che codificano 0 o 1 in un cavo, ottengo un cavo illeggibile / rumoroso / casuale
- se faccio confluire due cavi con 0 e Z in un cavo, ottengo un cavo con 0
- se faccio confluire due cavi con 1 e Z in un cavo, ottengo un cavo con 1
- se faccio confluire due cavi con Z e Z in un cavo, ottengo un cavo con Z



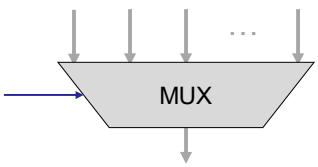
14

## Usi di un Tri-state buffer: per un MUTEX

- Un'implementazione di **MUTEX** molto **scalabile**
  - ma non necessariamente veloce
- Questo:

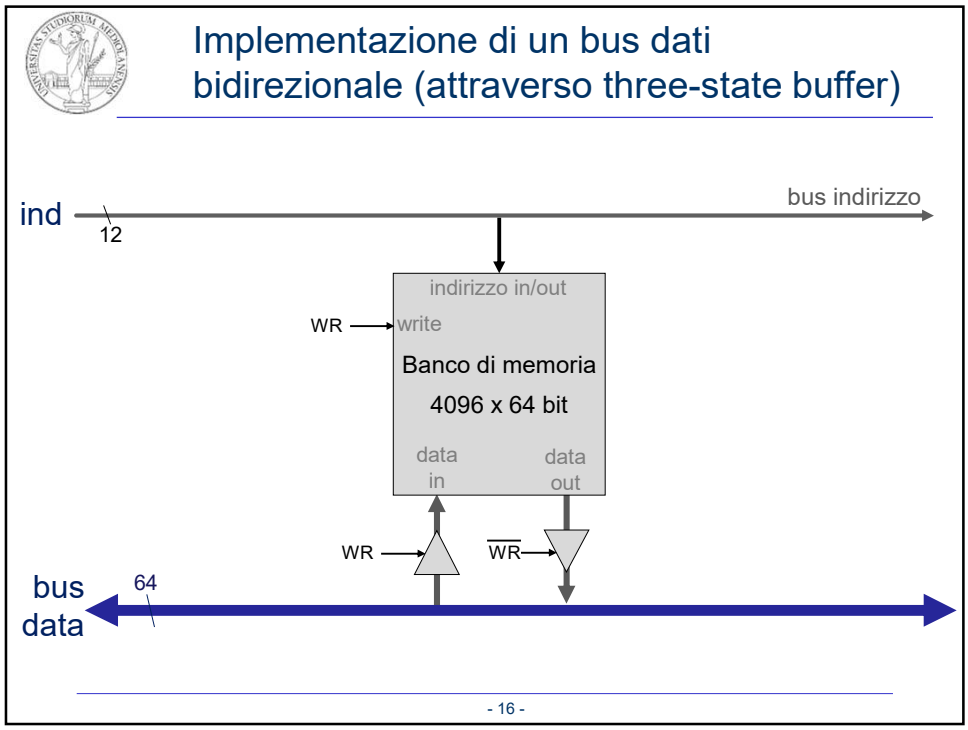



- E' un modo scalabile e economico di implementare questo:

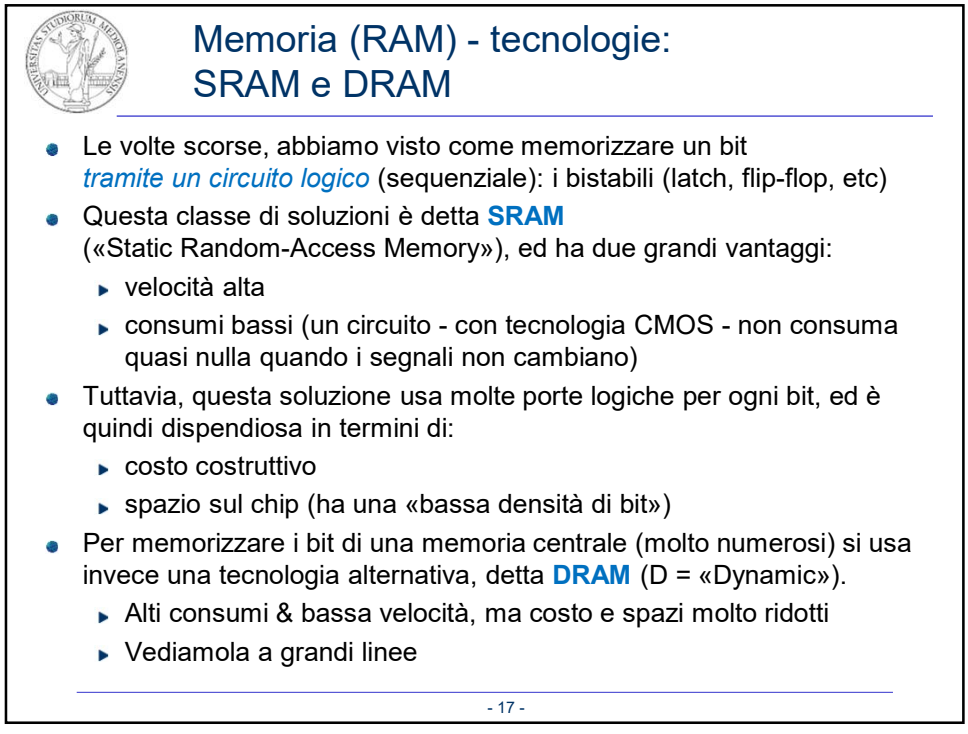




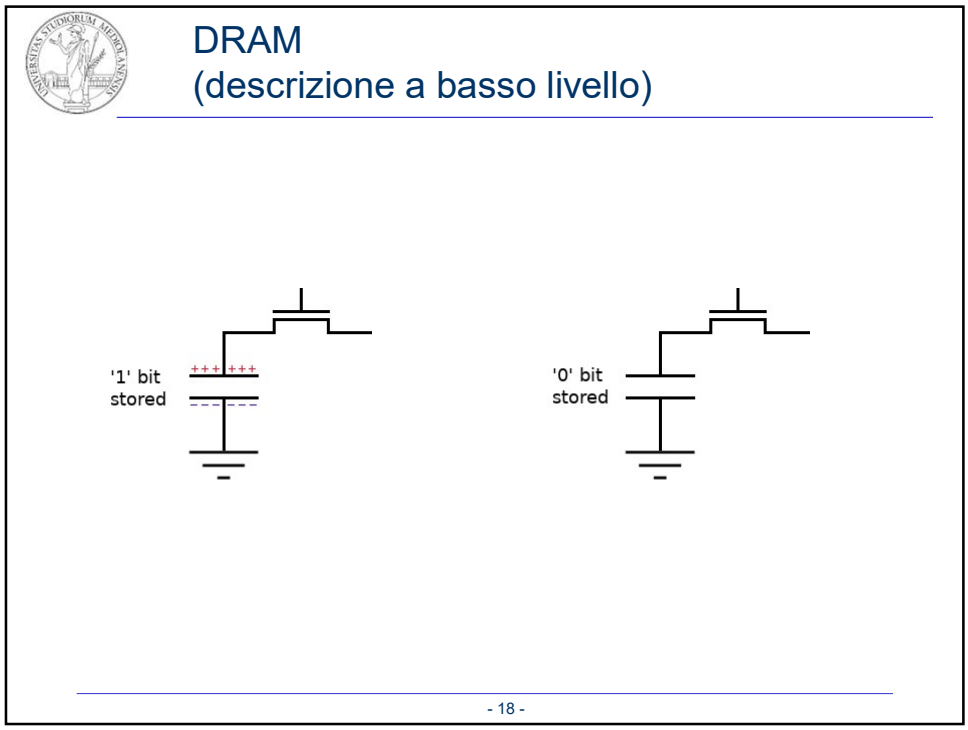

15


Implementazione di un bus dati bidirezionale (attraverso three-state buffer)
ind
12
bus indirizzo
indirizzo in/out
WR
write
Banco di memoria
4096 x 64 bit
data in
data out
WR
WR
bus data
64
- 16 -

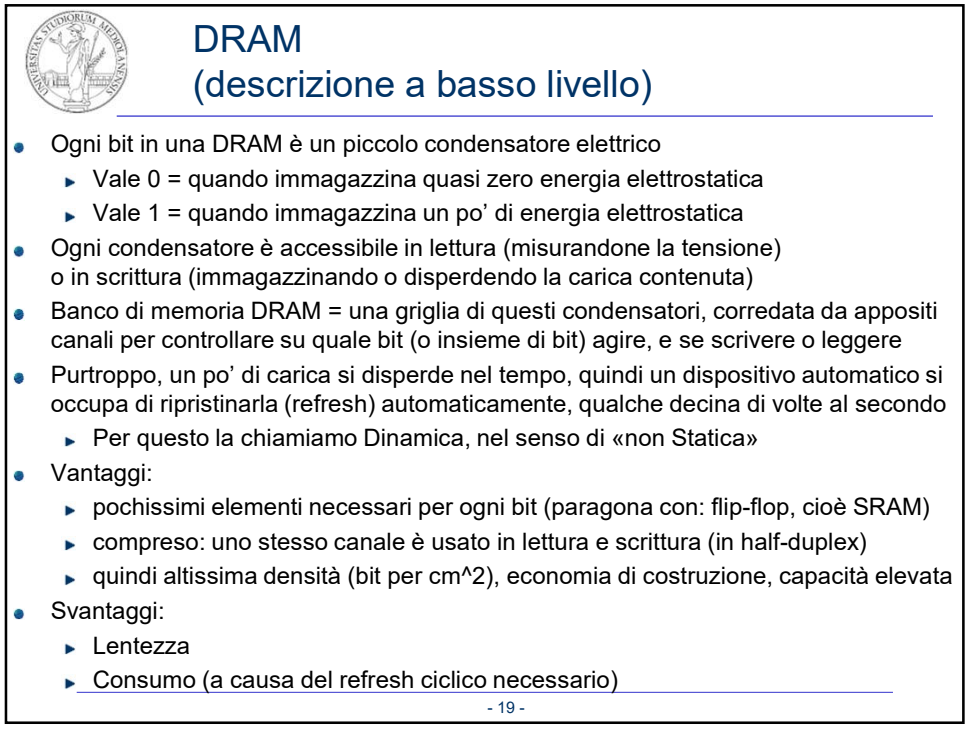
16


Memoria (RAM) - tecnologie: SRAM e DRAM
Le volte scorse, abbiamo visto come memorizzare un bit tramite un circuito logico (sequenziale): i bistabili (latch, flip-flop, etc)
Questa classe di soluzioni è detta SRAM («Static Random-Access Memory»), ed ha due grandi vantaggi:
velocità alta
consumi bassi (un circuito - con tecnologia CMOS - non consuma quasi nulla quando i segnali non cambiano)
Tuttavia, questa soluzione usa molte porte logiche per ogni bit, ed è quindi dispendiosa in termini di:
costo costruttivo
spazio sul chip (ha una «bassa densità di bit»)
Per memorizzare i bit di una memoria centrale (molto numerosi) si usa invece una tecnologia alternativa, detta DRAM (D = «Dynamic»).
Alti consumi & bassa velocità, ma costo e spazi molto ridotti
Vediamola a grandi linee
- 17 -

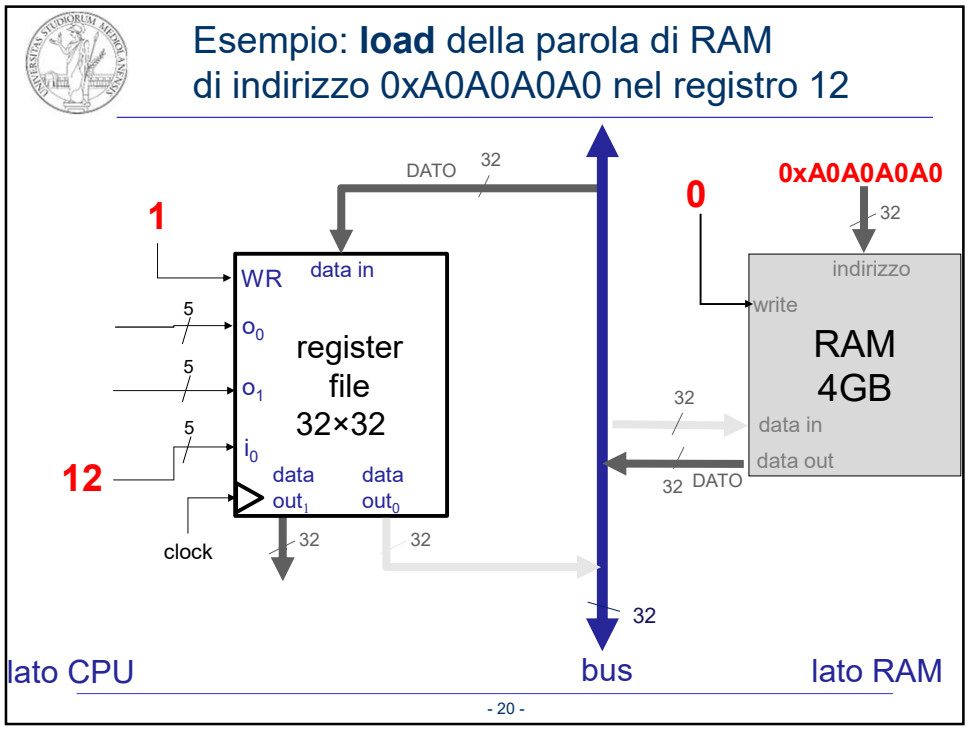
17


DRAM
(descrizione a basso livello)
'1' bit
stored
'0' bit
stored
- 18 -

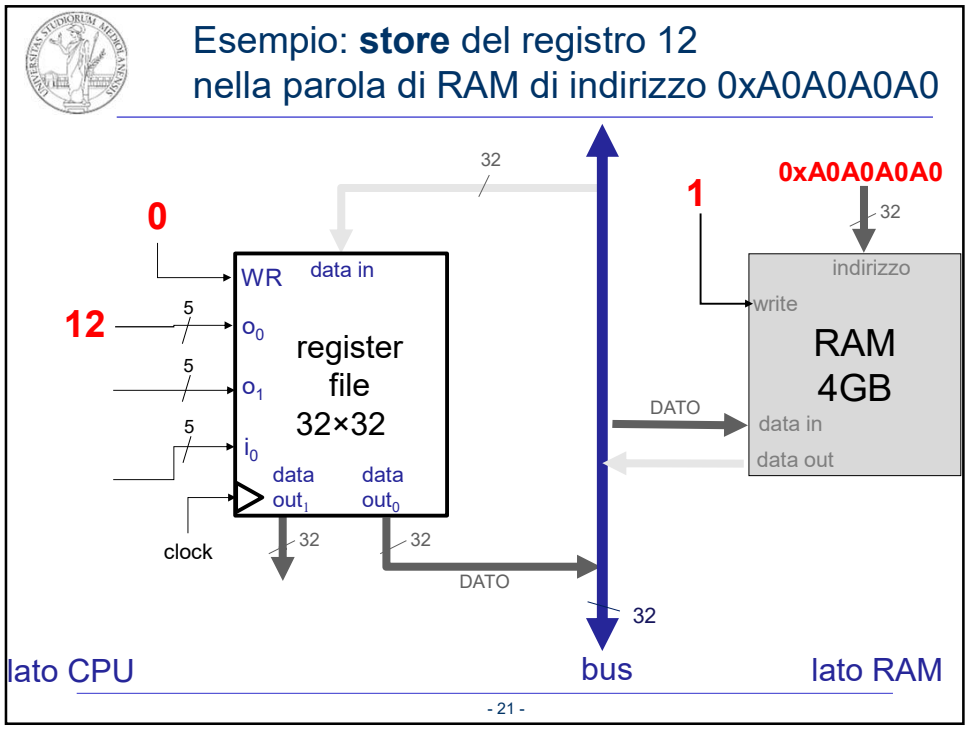
18


DRAM
(descrizione a basso livello)
Ogni bit in una DRAM è un piccolo condensatore elettrico
Vale 0 = quando immagazzina quasi zero energia elettrostatica
Vale 1 = quando immagazzina un po' di energia elettrostatica
Ogni condensatore è accessibile in lettura (misurandone la tensione)
o in scrittura (immagazzinando o disperdendo la carica contenuta)
Banco di memoria DRAM = una griglia di questi condensatori, corredata da appositi canali per controllare su quale bit (o insieme di bit) agire, e se scrivere o leggere
Purtroppo, un po' di carica si disperde nel tempo, quindi un dispositivo automatico si occupa di ripristinarla (refresh) automaticamente, qualche decina di volte al secondo
Per questo la chiamiamo Dinamica, nel senso di «non Statica»
Vantaggi:
pochissimi elementi necessari per ogni bit (paragona con: flip-flop, cioè SRAM)
compreso: uno stesso canale è usato in lettura e scrittura (in half-duplex)
quindi altissima densità (bit per cm^2), economia di costruzione, capacità elevata
Svantaggi:
Lentezza
Consumo (a causa del refresh ciclico necessario)
- 19 -

19


Esempio: load della parola di RAM
di indirizzo 0xA0A0A0A0 nel registro 12
DATO
32
0xA0A0A0A0
1
0
WR
data in
indirizzo
write
5
$o_0$
5
$o_1$
5
$i_0$
register
file
32×32
RAM
4GB
12
data
$out_1$
data
$out_0$
32
data in
data out
32
DATO
clock
32
32
32
lato CPU
bus
lato RAM
- 20 -

20


Esempio: store del registro 12
nella parola di RAM di indirizzo 0xA0A0A0A0
32
0xA0A0A0A0
0
1
WR
data in
indirizzo
write
12
5
$o_0$
5
$o_1$
5
$i_0$
register
file
32×32
RAM
4GB
DATO
data in
data out
data
$out_1$
data
$out_0$
clock
32
32
DATO
32
lato CPU
bus
lato RAM
- 21 -

21

## *Sommario:* Register file VS Main memory sono simili perché entrambi…

- Contengono un certo numero di parole di memoria di n bit ciascuno,
  - che possono essere letti oppure scritti individualmente
- Possiedono un certo numero (1 o più) di canali di ingresso
  - Per ciascuno:
    la parola da leggere o da scrivere (data).
    e l' indirizzo (address)
  - Se i bit di un indirizzo sono k, posso indirizzare $2^k$ parole diverse
- Prendono un comando WRITE che controllare se devono effettuare la scrittura oppure solo la lettura
- Sono sincronizzati su (il fronte di discesa di) un clock
  - Allo scattare del clock (e non prima),
    la parola in input viene memorizzata all'indirizzo in input
  - sempre che il comando WRITE sia, in quel momento alto, cioè 1
- Sono memorie "volatili" (cioè non "persistenti"):
  - In assenza di alimentazione, tutti i bit si perdono



22

## *Sommario:* Register file VS Main memory sono diversi nella pratica perché

- **Banco di registri**:
  - Numero di registri esiguo (da poche decine ad un centinaio)

  e di conseguenza…
  - Componente piccolo e veloce
  - E' integrato nella CPU, a bordo di uno stesso circuito e uno stesso chip (insieme alla ALU)
  - E' fisicamente vicino, e direttamente collegato, alla ALU
  - E' composto di un numero di registri paralleli di K bits,
    ciascuno implementato come Flip-Flip (tecnologia "DRAM")
  - Possiede N bus di ingresso e M di uscita, separate
    e usabili nello stesso momento
  - Cioè, ad ogni ciclo di clock, posso:
    leggere M parole da altrettanti registri, e contemporaneamente memorizzare N parole su altrettanti registri



23

## *Sommario:* Register file VS Main memory sono diversi nella pratica perché

- **Main memory**
  - Numero di celle di memoria molto grande: migliaia, milioni, miliardi di parole (words)

  e di conseguenza…
  - Componente grande e lento
  - Realizzato in un chip separato, esterno a quello che ospita la CPU (anzi, in realtà, su molti chip separati, connessi su uno stesso bus)
  - Possiede 1 solo bus di ingresso e 1 solo bus di uscita, oppure 1 solo bus usato sia in ingresso che uscita, (bus dati bidirezionale, implementato con three-state-buffer)
  - Quindi, in un ciclo di clock, o leggo o scrivo (non entrambe le cose)
  - Tecnologia utilizzata: “DRAM” (un mini condesatore per bit)



24

## Tecnologie per realizzare una memoria (note)

Esistono svariate altre tecnologie per implementare una memoria

Differiscono per:

- capacità realizzabile
- costo
- Velocità, cioè tempo di accesso a una parola (in lettura, o in scrittura)
- politica di accesso, come
  - lettura e scrittura (read/write),
  - sola lettura (read only) – i valori contenuti sono settati in fabbrica e non possono essere modificati una volta costruiti
  - scrittura limitata (read mostly) ad esempio, (scrittura supportata ma molto lenta, o possibile un numero limitato di volte, compreso «una sola»)
- stabilità: volatile o persistente
  - voltatile = appena cessa l’alimentazione, la memoria si cancella
  - persistente = i valori persistono anche senza alimentazione elettrica.



25

## Alcuni tipi di tecnologie comunemente utilizzate per la memoria, e loro caratteristiche

| Tipo di Memoria | Categoria | Modalità di cancellazione | Limiti di scrittura | volatile o persistente | Usi tipici |
|---|---|---|---|---|---|
| SRAM | Read/Write | elettrica | nessuno | voltatile | Registri cache |
| DRAM | Read/Write | elettrica | nessuno | voltatile | memoria centrale |
| ROM | Read Only | N/A | N/A | persistente | grandi vol. |
| PROM | Read Mostly | nessuna | lentissima | persistente | piccoli vol. |
| EPROM | Read Mostly | luce UV | lentissima | persistente | prototipi |
| EEPROM | Read Mostly | elettrica | lenta | persistente | prototipi |
| FLASH | Read/Write | elettrica | lenta, a blocchi | persistente | dischi esterni SSD |

«Read Mostly»: la scrittura è possibile, ma in modo molto lento o limitato



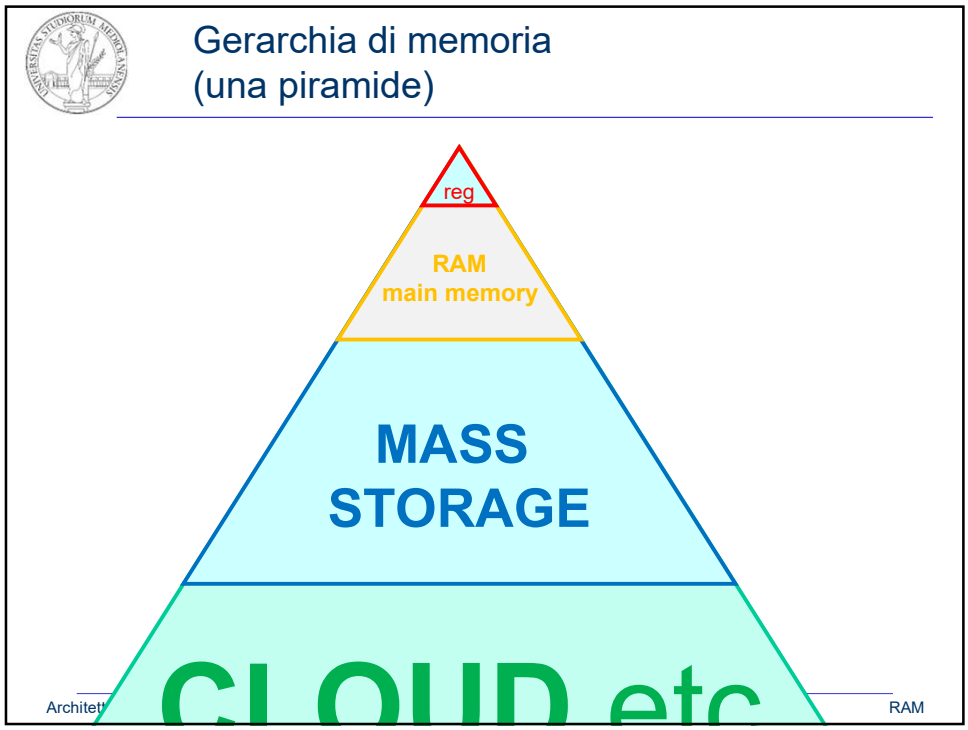
26

## I livelli della piramide di memoria (concetto generale)

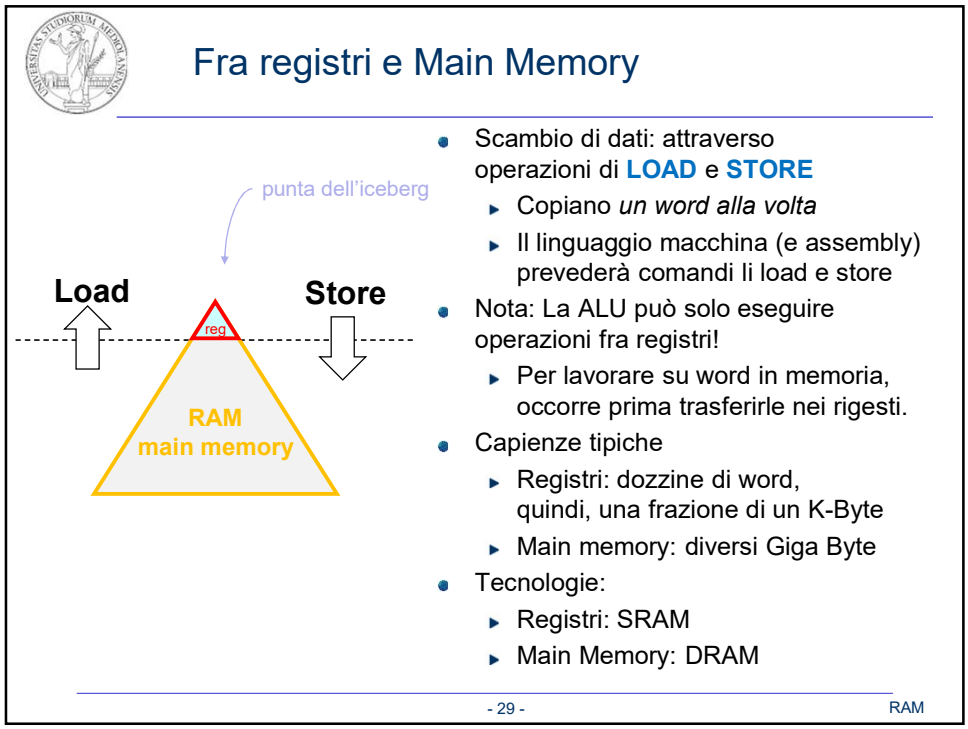
- Disponiamo di diverse tecnologie di memoria, che si distinguono per:
  - Dimensione, velocità di accesso e costo per bit
- In generale:
  - Le memorie piccole sono molto veloci, ma costose
  - Le memorie grandi sono più lente, ma economiche
- Idea chiave: combinare più tipi di memoria per ottenere un buon compromesso tra prestazioni e costo
  - Usiamo le memorie piccole e veloci per i dati usate di frequente e le memorie grandi e lente per l'archiviazione di grandi quantità di dati
- Questa osservazione porta al concetto di piramide della memoria, organizzata in livelli (*tiers*) con caratteristiche diverse
  - ⚠ Nota: qui "livello" (o *tiers*) non si riferisce al "livello di astrazione", ma alla gerarchia di memoria fisiche
- I dati più usati tendono a risiedere nei livelli più alti (fino ai registri), mentre quelli meno usati si trovano nei livelli inferiori (RAM, disco, ecc.)



27

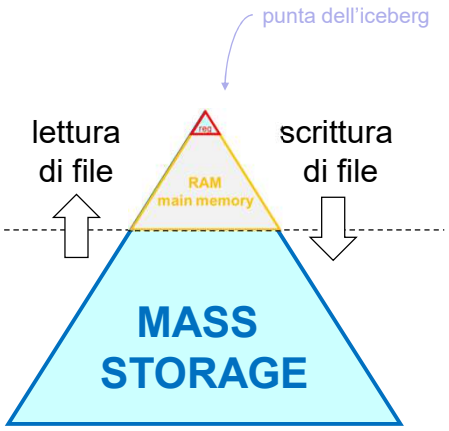

Gerarchia di memoria
(una piramide)
reg
RAM
main memory
MASS
STORAGE
CLOUD etc
Architett
RAM

28


Fra registri e Main Memory
punta dell'iceberg
Load
Store
reg
RAM
main memory
Scambio di dati: attraverso operazioni di LOAD e STORE
Copiano un word alla volta
Il linguaggio macchina (e assembly) prevederà comandi li load e store
Nota: La ALU può solo eseguire operazioni fra registri!
Per lavorare su word in memoria, occorre prima trasferirle nei rigesti.
Capienze tipiche
Registri: dozzine di word, quindi, una frazione di un K-Byte
Main memory: diversi Giga Byte
Tecnologie:
Registri: SRAM
Main Memory: DRAM
- 29 -
RAM

29

## Mass storage (cenno, esula in da questo corso)

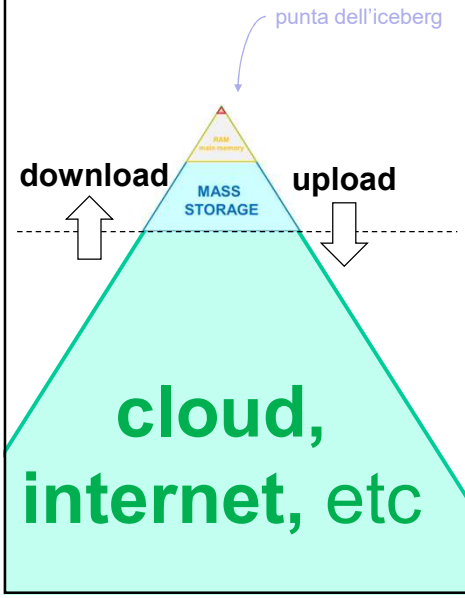



- Capienze tipiche
  - diversi Tera-Bytes
- Tipologie e tecnologie:
  - Hard Disk Drives (HDD)
  - Solid State Drives (SSD)
- A differenza dei livelli sopra, questo livello è **persistente**
- Trasferimento dati:
  - A blocchi (regioni di Kbyte alla volta)
  - Gestito dal sistema operativo
  - Astrazione esposta all'utente: memoria strutturata in diversi «file» del «file system»
  - Vedere corso di **Sist. Operativi**



30

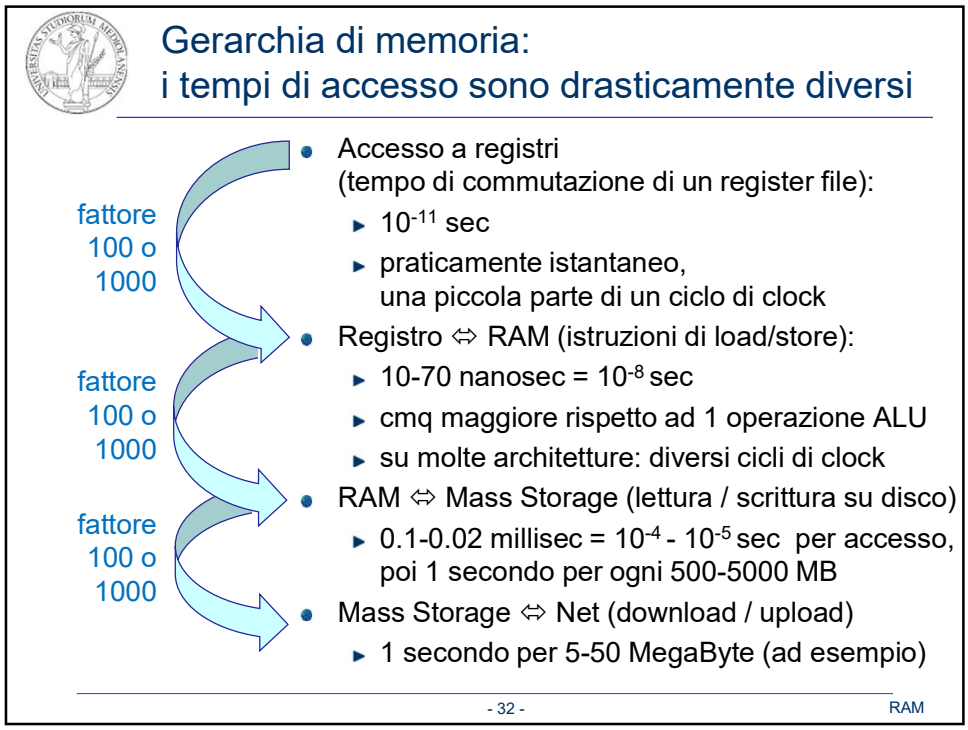
## Informazione al di fuori del computer



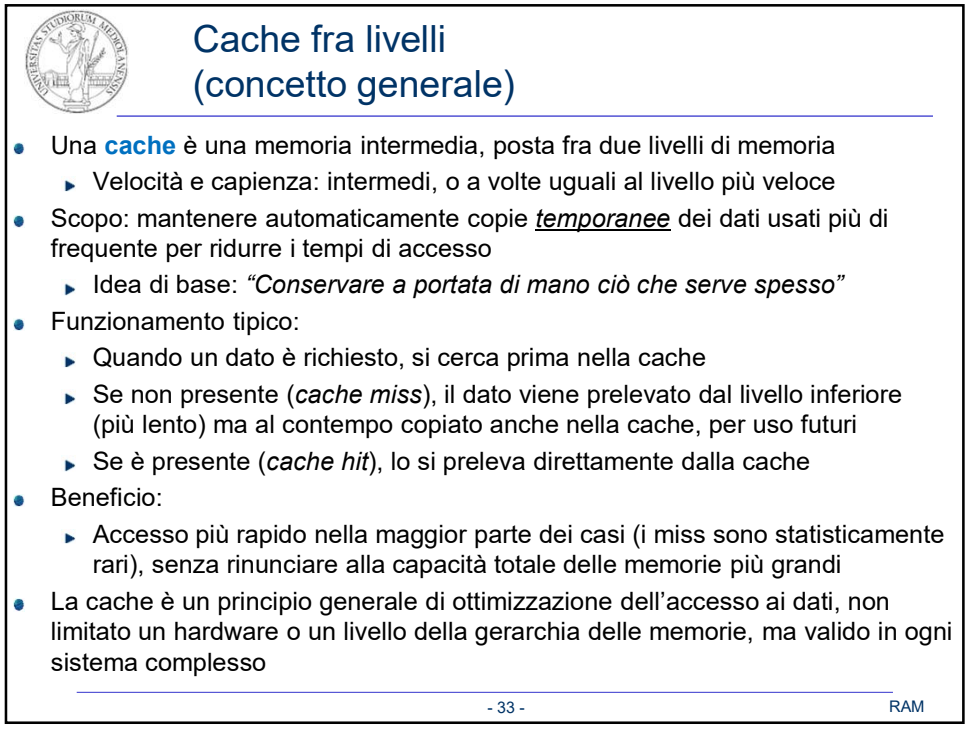

- A differenza dei livelli sopra, che erano **locali** (memorizzate sul computer) in questo livello i dati sono in **remoto** (memorizzati su dispositivi fisicamente al di fuori dal computer, connessi in una network)
- Capienze tipiche
  - Dipende: fino a Haxa-Bytes?
- Questi argomenti sono svolti nei corso di **Reti di Calcolatori**



31


Gerarchia di memoria:
i tempi di accesso sono drasticamente diversi
fattore 100 o 1000
fattore 100 o 1000
fattore 100 o 1000
Accesso a registri (tempo di commutazione di un register file):
10-11 sec
praticamente istantaneo, una piccola parte di un ciclo di clock
Registro ⇔ RAM (istruzioni di load/store):
10-70 nanosec = 10-8 sec
cmq maggiore rispetto ad 1 operazione ALU
su molte architetture: diversi cicli di clock
RAM ⇔ Mass Storage (lettura / scrittura su disco)
0.1-0.02 millisec = 10-4 - 10-5 sec per accesso, poi 1 secondo per ogni 500-5000 MB
Mass Storage ⇔ Net (download / upload)
1 secondo per 5-50 MegaByte (ad esempio)
- 32 -
RAM

32


Cache fra livelli
(concetto generale)
Una cache è una memoria intermedia, posta fra due livelli di memoria
Velocità e capienza: intermedi, o a volte uguali al livello più veloce
Scopo: mantenere automaticamente copie temporanee dei dati usati più di frequente per ridurre i tempi di accesso
Idea di base: "Conservare a portata di mano ciò che serve spesso"
Funzionamento tipico:
Quando un dato è richiesto, si cerca prima nella cache
Se non presente (cache miss), il dato viene prelevato dal livello inferiore (più lento) ma al contempo copiato anche nella cache, per uso futuri
Se è presente (cache hit), lo si preleva direttamente dalla cache
Beneficio:
Accesso più rapido nella maggior parte dei casi (i miss sono statisticamente rari), senza rinunciare alla capacità totale delle memorie più grandi
La cache è un principio generale di ottimizzazione dell'accesso ai dati, non limitato un hardware o un livello della gerarchia delle memorie, ma valido in ogni sistema complesso
- 33 -
RAM

33

## Istanze di cache fra diversi livelli

- Fra registri e RAM: **cache del processore**
  - Il processore incorpora alcuni banchi di memoria dedicati a memorizzare i dati letti e scritti di recente in RAM
  - spesso realizzati con tecnologia SRAM – insomma con flip-flop
  - nelle operazioni di load, se è richiesto un indirizzo che è presente nella cache, il dato viene prelevato dalla cache e non dalla main memory (RAM)
  - i processori moderni ospitano alcuni livelli diversi di cache, da usare in cascata (via via più grandi e meno veloci) chiamati L1 L2 L3
- Fra RAM e mass storage: **cache dei dischi**
  - Gli hard-disk incorporano della banchi di memoria (DRAM – non persistente) che memorizza i blocchi di dati letti o scritti nel disco, da usare senza accedere nuovamente al disco qualora venisse richiesto lo stesso blocco di memoria
  - Vantaggio ulteriore: riduce l'usura delle componenti meccaniche del drive fisico
- Fra PC (mass storage) e web: **cache del browser**
  - Il web-browser (come Firefox o Chrome) mantiene dei file su disco che registrano i file scaricati di recente. Se gli stessi file vengono richiesti una seconda volta, li si legge dal disco (locale) invece che effettuare un nuovo download (da remoto)



## Domande

- Se il mio computer ha uno spazio di indirizzamento di 32 bit, quanti byte di memoria possiede al massimo?
- Se ne ha 64 bit?
- (esprimiti in MegaByte, GigaByte, etc)