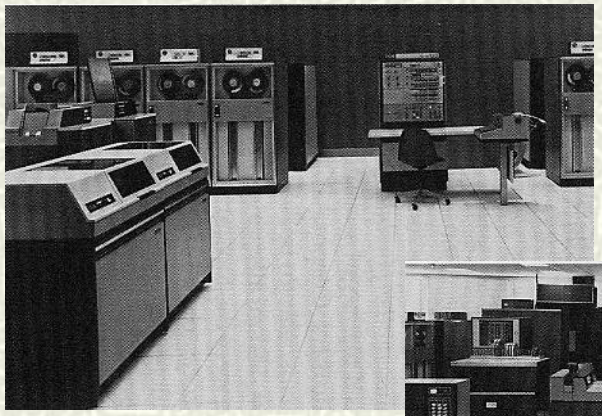
Informatica Generale

Introduzione

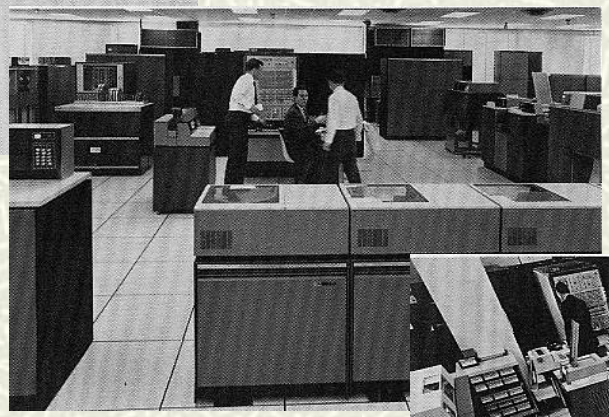
La struttura del calcolatore

- La struttura del calcolatore
  - L'architettura a BUS
  - La CPU
  - La memoria centrale
  - La memoria secondaria
  - I dispositivi di I/O
- Linguaggio macchina e assembler
- Il sistema operativo
  - Scopo del sistema operativo
  - Architettura e servizi

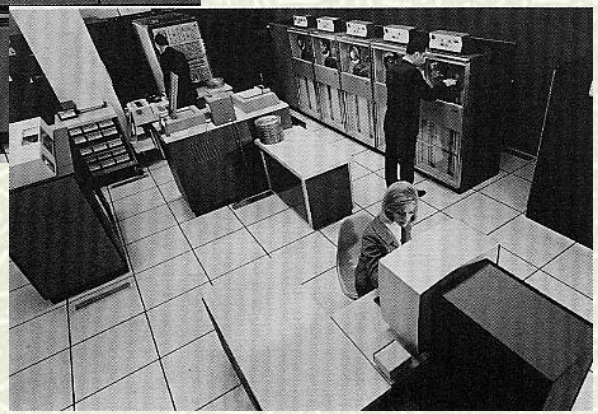
Il mercato informatico nel 1964



IBM S/360 Mod 40
1.6 MHz 32KB-256KB
225.000$



IBM S/360 Mod 50
2.0 MHz 128KB-256KB
550.000$



IBM S/360 Mod 65
5.0 MHz 256KB-1MB
1.200.000$

La struttura del calcolatore

Struttura del calcolatore

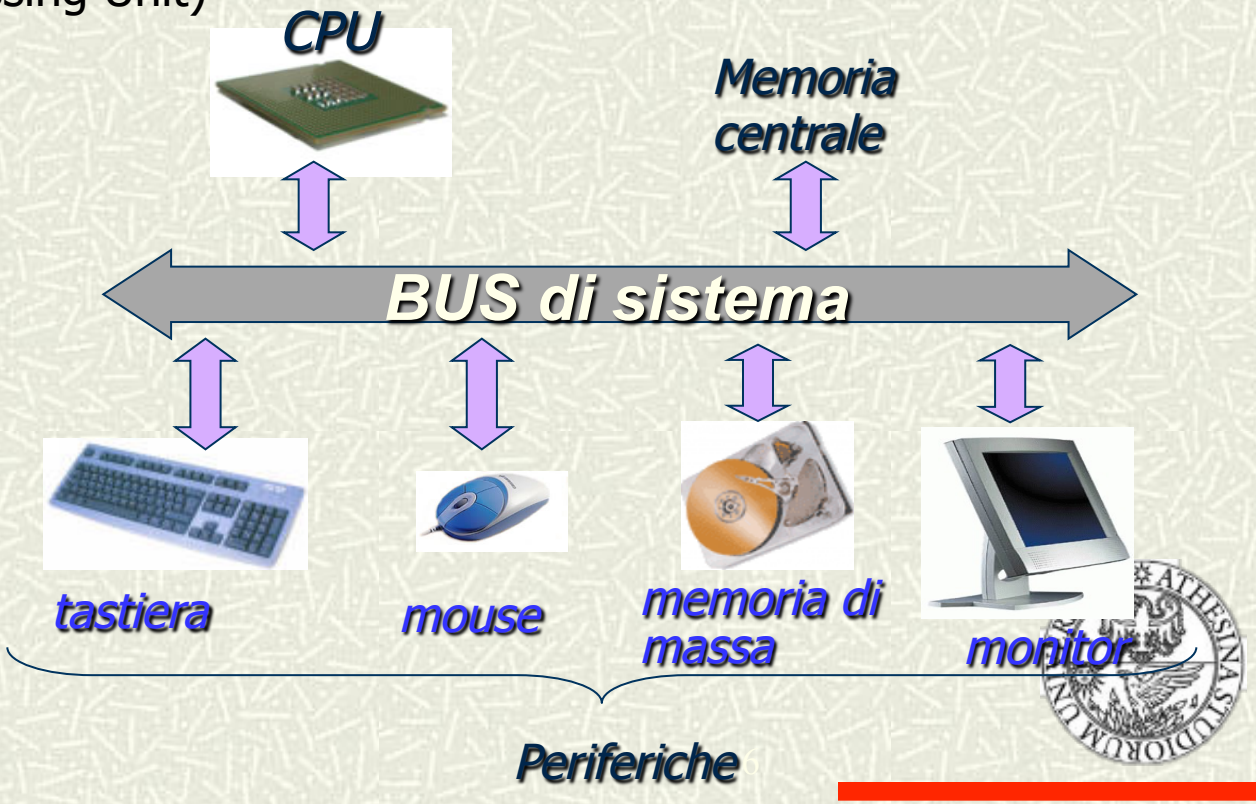
- Si possono considerare vari livelli di astrazione:
  - Circuiti elettronici (hardware)
  - Architettura e linguaggio macchina
  - Sistema operativo (software di sistema)
  - Linguaggi di programmazione
  - Programmi applicativi

Silicon Graphics

Il calcolatore è basato su circuiti elettronici digitali, ovvero modellabili con l'algebra di Boole; i circuiti elettronici implementano le funzioni logiche AND, OR, NOT, permettono di memorizzare il valore di variabili booleane, di effettuare calcoli, etc.

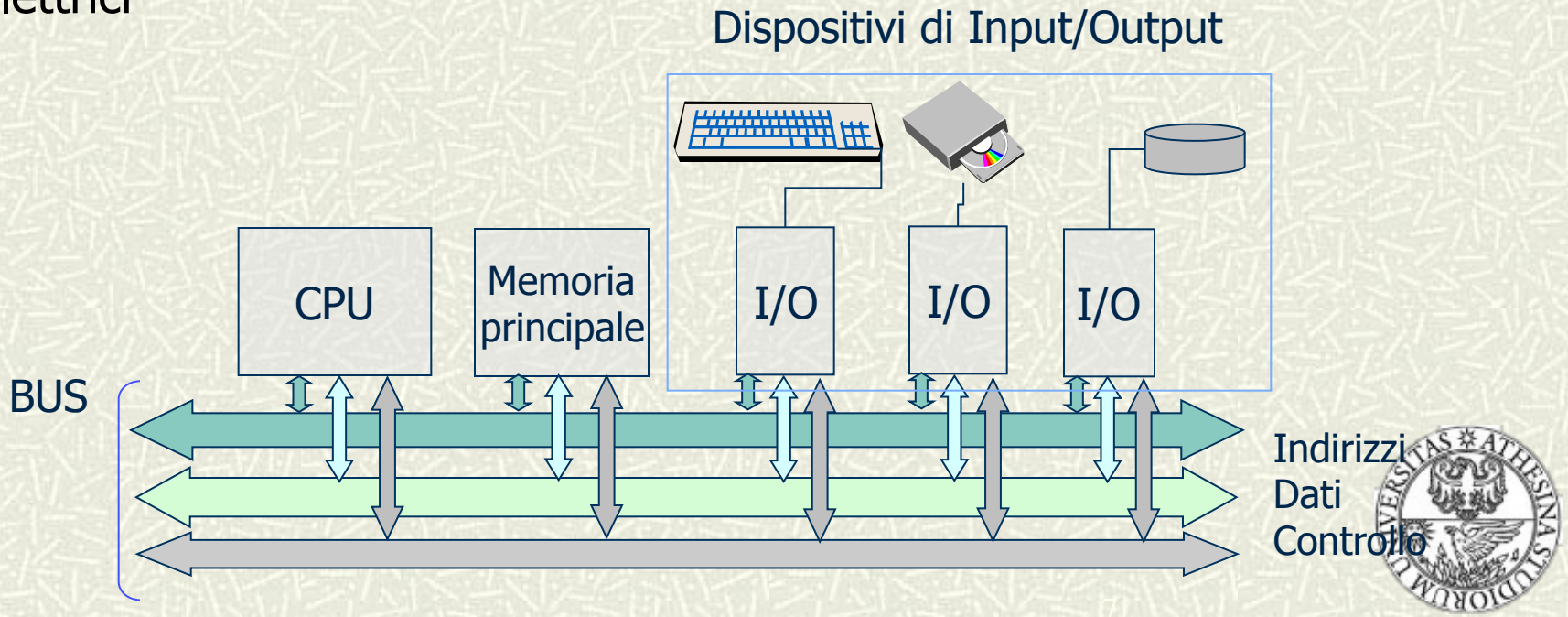
La "macchina" di Von Neumann

- Tutti i calcolatori attuali si rifanno all'architettura di Von Neumann, costituita dalle quattro componenti:
  - CPU (Central Processing Unit)
  - Memoria centrale
  - Bus di sistema
  - Periferiche



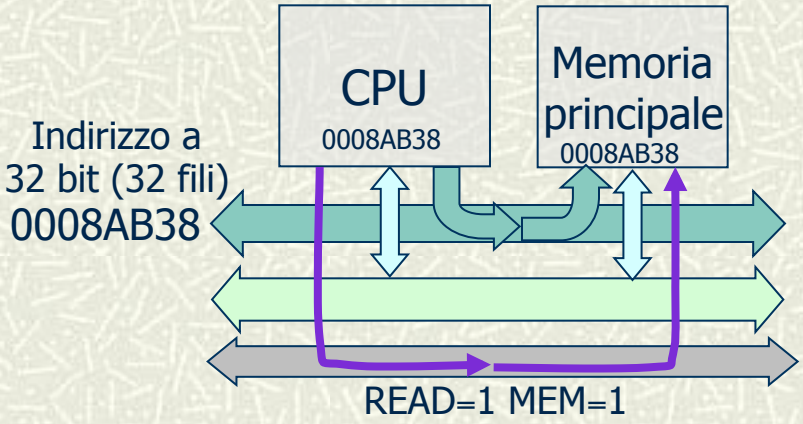
Architettura a BUS

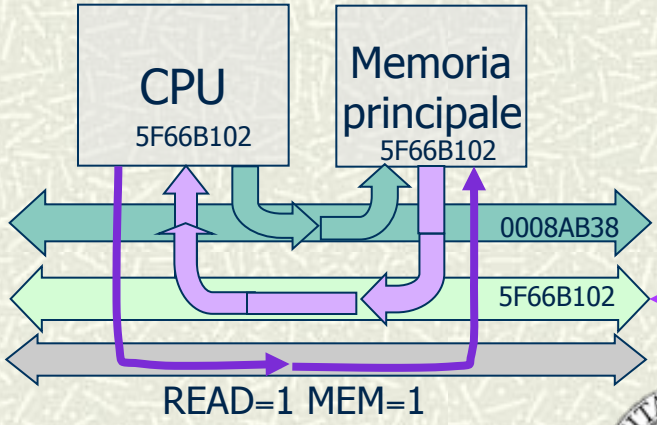
- L'architettura più consolidata per il calcolatore prevede quindi unità funzionali fra loro collegate attraverso un unico canale di comunicazione, il **bus**
- Il bus è fisicamente realizzato mediante un insieme di conduttori elettrici



Il BUS

- Il bus è utilizzato per trasferire dati fra le unità funzionali
  - L'unità che inizia il trasferimento (in genere la CPU) fornisce l'indirizzo, che individua univocamente il dato, sulle linee del **bus indirizzi**, e configura le linee del **bus controllo**, inviando un comando al dispositivo che contiene il dato (es. READ)
  - Il dato da trasferire è reso disponibile sul **bus dati** e viene ricopiato nel dispositivo destinatario

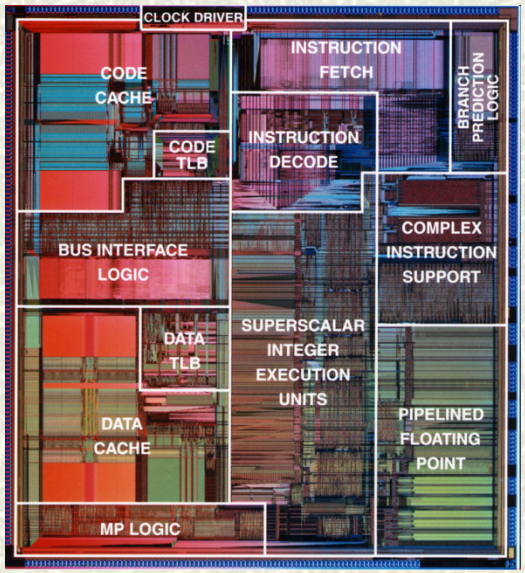




5F66B102
Dato a 32 bit (32 fili)
trasferito dalla memoria principale alla CPU

La CPU - 1

- La **Central Processing Unit** è l'unità centrale di elaborazione: esegue le istruzioni del programma e ne regola il flusso, esegue i calcoli
- La CPU è un dispositivo ***sincrono***, cioè può cambiare stato solo quando riceve un impulso di clock, l'**orologio del sistema** che fornisce al computer un battito regolare
- La CPU lavora a *N* GHz: segue un ritmo di *N* miliardi di impulsi al secondo (es., una CPU con un clock a 3 GHz è temporizzata da tre miliardi di impulsi al secondo)



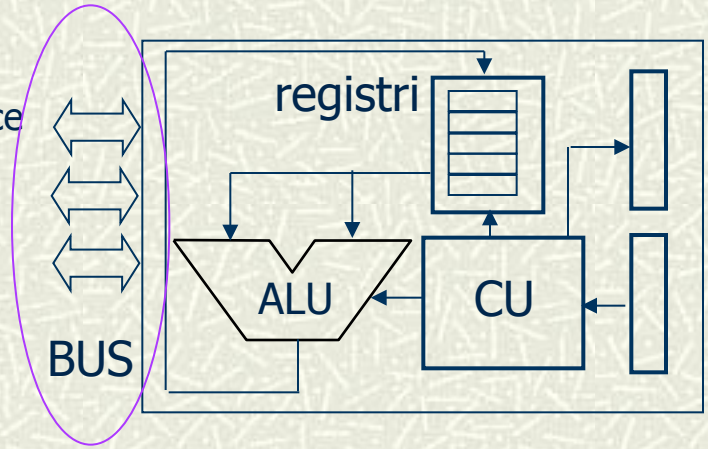
Intel Pentium

La CPU - 2

- È costituita da tre elementi fondamentali:
  - Unità Aritmetico-Logica (ALU)
  - Registri
  - Unità di Controllo (CU)

} EU
Execution Unit

BIU
Bus Interface Unit



PC Contatore di programma (Program Counter)

IR Registro Istruzione (Instruction Register)

La CPU - 3

- A livello "macroscopico", ad ogni impulso di clock la CPU:
  - "legge" il suo stato interno (determinato dal contenuto dei registri di stato) e la sequenza di ingresso (determinata dal contenuto dei registri istruzione e dati)
  - produce un nuovo stato "dipendente" dallo stato in cui si trovava originariamente
- In pratica, la CPU realizza una complessa funzione logica, con decine di ingressi e di uscite
  - ⇨la corrispondente tabella della verità avrebbe un numero enorme di righe (miliardi di miliardi)

La CPU - 4

- Lo stato della CPU è costituito da informazioni (memorizzate negli opportuni registri) su:
  - dati da elaborare (contenuti nei ***registri dati***)
  - istruzione da eseguire (nel ***registro istruzioni***)
  - indirizzo in memoria della prossima istruzione da eseguire (nel ***program counter***)
  - eventuali anomalie/eventi verificatisi durante l'elaborazione (nei ***registri flag***)

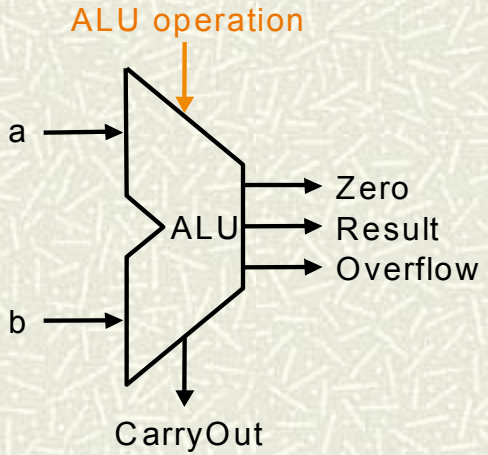
La CPU - 5

- Set di istruzioni di base:
  - somma (da cui sottrazione)
  - scorrimento (shift)
  - operazioni logiche

  } (da cui moltiplicazione e divisione)
  - operazioni di accesso alla memoria
    - trasferimento di un dato da una locazione di memoria ad un'altra
    - trasferimento da memoria a un registro della CPU
    - trasferimento da un registro della CPU a memoria
  - operazioni di confronto (sufficiente confronto con zero)
- Le operazioni (eccetto quelle di accesso alla memoria) sono eseguite all'interno della ALU e "coordinate" dall'unità di controllo

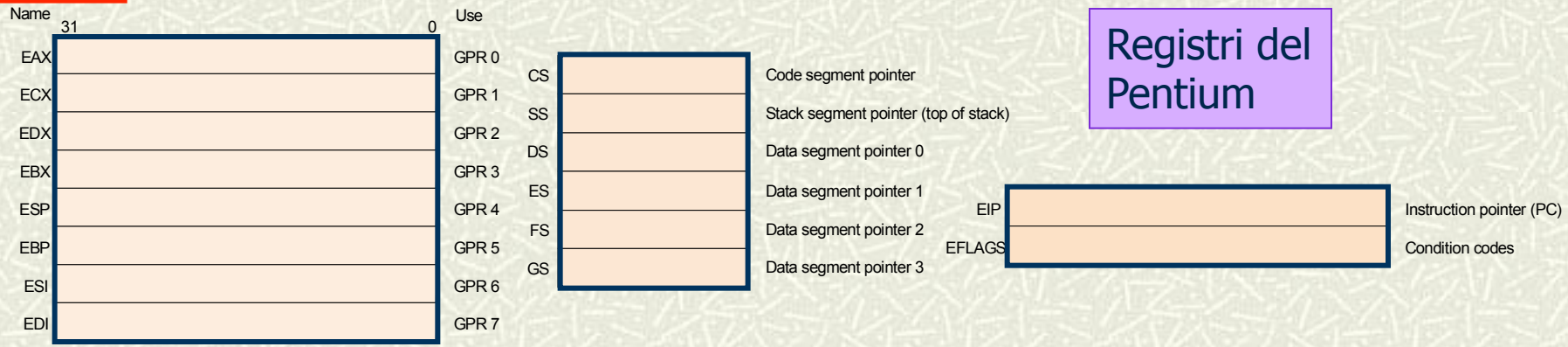
L'Unità Aritmetico-Logica

- L'ALU (Arithmetic-Logic Unit) è un circuito in grado di eseguire operazioni aritmetiche e logiche su 2 operandi, rappresentati su ***n*** bit (es. 32/64 bit); oltre al risultato dell'operazione può produrre informazioni ulteriori su linee specifiche (il risultato è zero, si è verificato un overflow, etc.)



- Il tipo di operazione selezionata, in un dato istante, dipende dallo stato di alcune linee di controllo provenienti dalla CU
- Le operazioni logiche (es. AND) vengono eseguite bit a bit fra i due operandi
- Esiste una unità specializzata per le operazioni in virgola mobile (FPU)

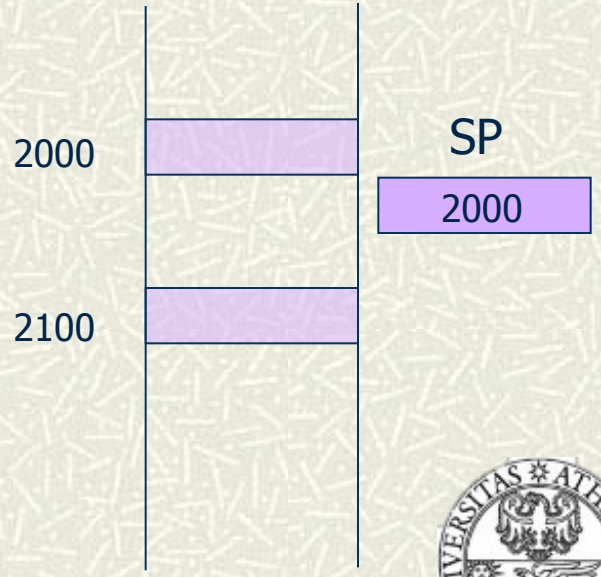
I registri



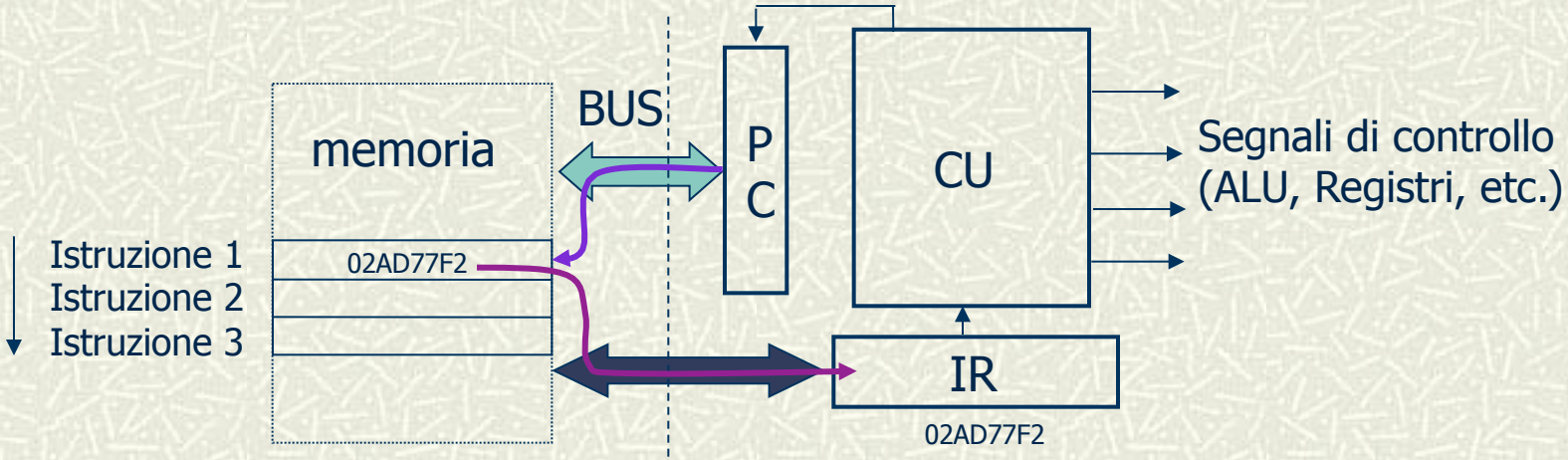
- I registri sono dispositivi di memorizzazione che consentono un accesso molto veloce ai dati che contengono; hanno dimensioni prefissate (es. 32/64 bit)
- Alcuni registri hanno funzioni specifiche (es. contatore di programma)
- Nella maggior parte delle architetture, le operazioni della ALU si possono effettuare solo fra dati presenti in due registri
- Il risultato di un'operazione effettuata dalla ALU viene normalmente memorizzato in un registro

Registri e loro funzioni

- **Registri accumulatori**: sono integrati alla ALU, che vi può operare direttamente; mantengono risultati temporanei
- **Registro di stato**: contiene informazioni che globalmente determinano, in ogni istante, lo stato dell'unità centrale
- **Registro puntatore allo stack** (Stack Pointer): lo stack è un insieme di celle di memoria che può essere acceduto in modalità LIFO (*Last-In-First-Out*); la posizione in cui si trova l'elemento inserito per ultimo è la testa dello stack, che viene puntata dal registro SP; lo stack è organizzato in modo da crescere verso indirizzi decrescenti: SP viene decrementato ad ogni nuova immissione (**push**) ed incrementato ad ogni prelievo (**pop**)



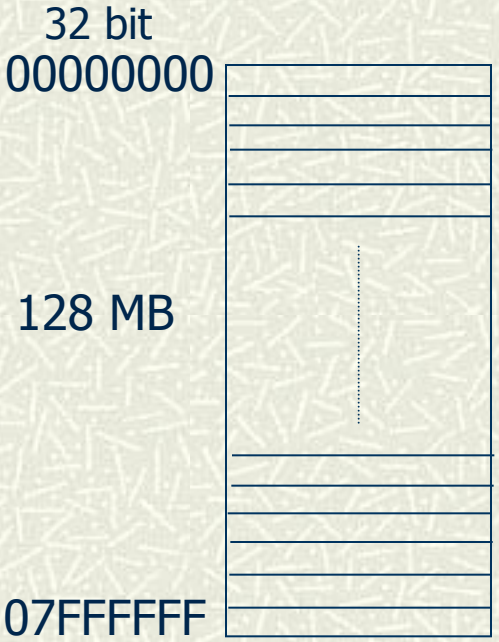
L'Unità di Controllo



- Esegue le istruzioni prelevandole dalla memoria nella fase di **fetch**
- La prossima istruzione da eseguire è individuata dall'indirizzo presente nel **registro contatore di programma** (PC)
- L'istruzione in esecuzione è memorizzata nel **registro istruzione** (IR)
- L'istruzione è un codice binario che deve essere **decodificato** dalla CU; specifica il tipo di operazione, gli eventuali operandi, etc.
- Normalmente le istruzioni sono eseguite in sequenza: dopo il fetch, il **PC viene incrementato**, per fare riferimento all'istruzione successiva

La memoria centrale - 1

- L'unità di memorizzazione è il byte
- Ciascun byte nella memoria è individuato da un **indirizzo** che lo distingue da tutti gli altri, costituito da un numero variabile da 0 a 2^N-1 dove N è la dimensione in bit dell'indirizzo (es. numero di bit/fili sul bus indirizzi)



2^{27}

128 MB = 2^7 2^{20} byte =
= 137438953472 byte

00000xxxxxxxxxxxxxxxxxxxxxxxxx

27 bit — indirizzo all'interno del blocco

La memoria centrale - 2

- La memoria centrale viene anche chiamata memoria ad accesso casuale o RAM (**Random Access Memory**): qualsiasi cella può essere letta/scritta in un tempo (mediamente) costante
- **La memoria centrale RAM è volatile**
- Una parte della memoria centrale, la ROM (**Read Only Memory**), viene scritta in modo permanente in fase costruttiva: le celle della ROM possono essere successivamente lette (ed in generale contengono informazioni fondamentali, specialmente per l'inizializzazione dell'elaboratore), ma mai riscritte

Il software della ROM - 1

- La ROM contiene il software e i dati necessari ad inizializzare il computer ed a far funzionare i dispositivi periferici
- Il nucleo del software della ROM è costituito dalle **routine di avviamento** che comprendono il **caricatore di boot-strap** ed il **ROM BIOS**
- **Le routine di avviamento** realizzano l'inizializzazione del calcolatore:
  - Ne effettuano un rapido controllo di affidabilità, per accertare che tutte le componenti hardware siano perfettamente funzionanti
  - Caricano il sistema operativo dal disco (caricatore di boot-strap)

Il software della ROM - 2

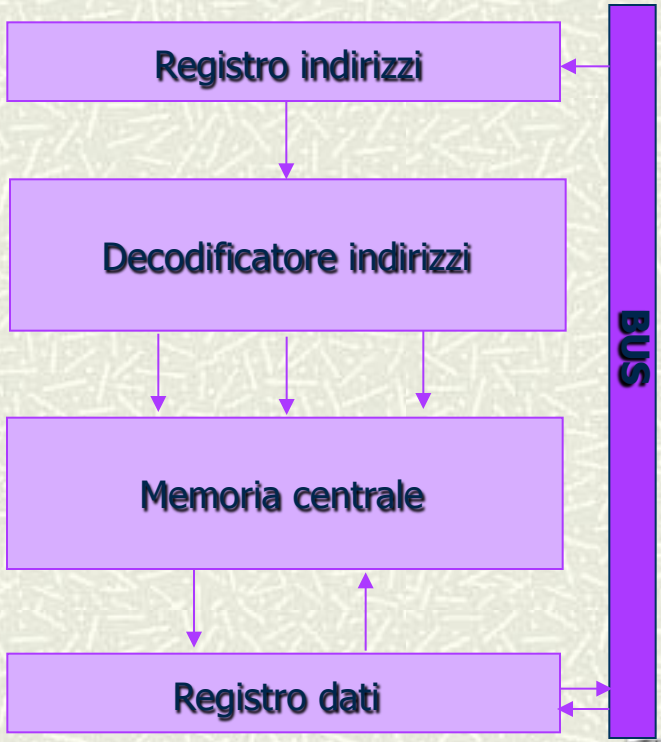
- Il caricatore di boot-strap ha la funzione di leggere un programma di lancio dal disco, detto ***bootstrap***, e di trasferire ad esso il controllo: il boostrap carica il nucleo del sistema operativo e lo manda in esecuzione
- Il ROM BIOS — ***Binary Input-Output System*** — è la parte della ROM attiva quando il computer è al lavoro: il suo ruolo è quello di fornire un insieme di servizi di base richiesti per il funzionamento delle periferiche

Operazioni sulla memoria centrale

- Le operazioni che si effettuano sulla memoria sono operazioni di lettura e scrittura
- Entrambe presuppongono l'utilizzo di un indirizzo che identifica univocamente la cella interessata all'operazione
- L'operazione di scrittura è **distruttiva**, cioè cancella l'informazione precedentemente contenuta nella cella
- L'operazione di lettura preserva il contenuto della cella indirizzata: all'esterno della memoria centrale viene trasferita copia dell'informazione

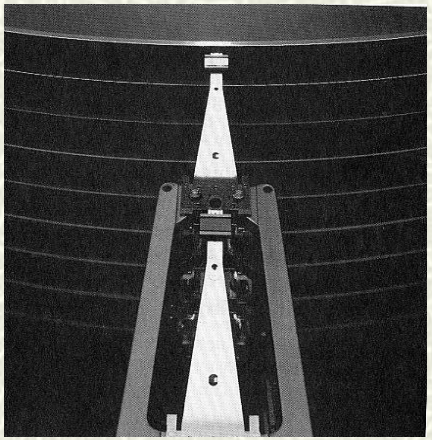
Architettura della memoria centrale

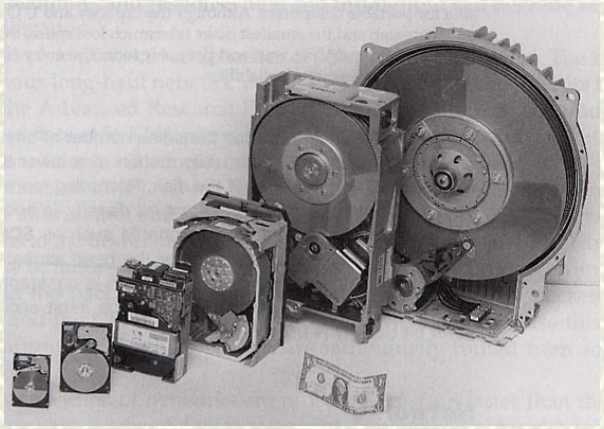
- Il **registro indirizzi** contiene l'indirizzo della cella che deve essere acceduta; la sua dimensione deve essere tale da permettere che tutte le celle di memoria possano essere indirizzate
- **Esempio**: un registro a 16 bit indirizza 65536 posizioni di memoria
- Il **decodificatore di indirizzi** è un dispositivo in grado di selezionare la cella il cui indirizzo corrisponde a quello contenuto nel registro indirizzi
- Il **registro dati** contiene l'informazione scritta/letta sulla/dalla cella indirizzata; la dimensione del registro è uguale a quella delle celle di memoria



La memoria secondaria

- Esistono diversi dispositivi di memoria secondaria: dischi magnetici (hard disk), dischi ottici (CD, DVD), dispositivi USB
- Memoria non volatile ad alta capacità



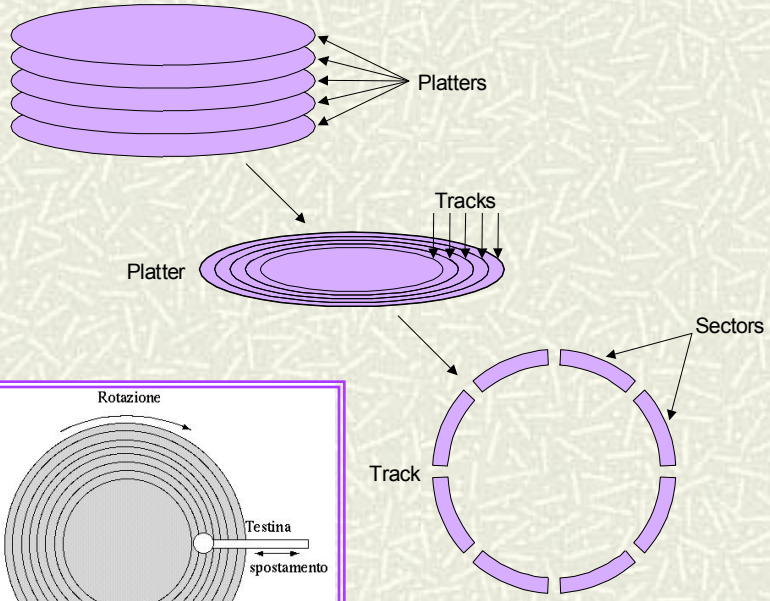


Dischi magnetici

- Il disco fisso è costituito da uno o più piatti metallici ricoperti di materiale magnetico su entrambe le facce
- Ciascuna superficie è associata ad una o più testine di lettura/scrittura che si muovono radialmente per leggere/scrivere l'informazione organizzata in tracce concentriche

I dischi magnetici

- L'informazione è disposta sul disco in **tracce**, ovvero in cerchi concentrici
- Le tracce sono divise in **settori**
- Il numero di bit contenuti in una traccia è dell'ordine delle centinaia di migliaia (512/4096 byte a settore)



- Per leggere (o scrivere) sul disco, la testina si deve posizionare sulla traccia che contiene il dato, ed attendere che il dato passi sotto di essa

Accesso al disco

- Il tempo medio di accesso all'informazione memorizzata su disco è dato da

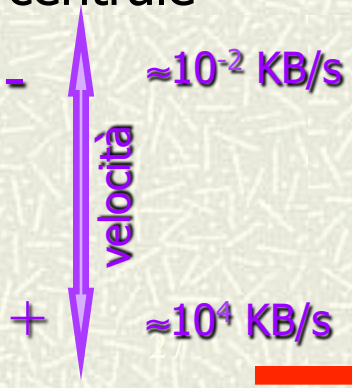
$$T = t_{seek} + t_{lat} + t_{tr}$$

  - t_{seek} è il *tempo di ricerca*, necessario per posizionare la testina sulla traccia che contiene l'informazione; dipende dall'ampiezza dello spostamento
  - t_{lat} è il *tempo di latenza*, necessario perché l'informazione ricercata passi sotto la testina; dipende dalla velocità di rotazione dei dischi
  - t_{tr} è il *tempo di trasferimento*; dipende dalla velocità di rotazione, dalla densità di registrazione e dalla quantità di informazione da trasferire

I dispositivi di Input-Output

- Insieme di dispositivi che consentono l'acquisizione di dati (**input**), la loro archiviazione (**storage**) e la loro presentazione verso il mondo esterno (**output**)
- Si possono classificare in base a tre diverse caratteristiche:
  - Comportamento: Input (read once), output (write only), memoria (rilettura/riscrittura)
  - Partner: uomo o macchina
  - Velocità del flusso dei dati: quantità di dati trasferiti nell'unità di tempo da o verso la CPU o la memoria centrale

Tastiera	input	uomo
Mouse	input	uomo
Rete	input/output	macchina
Hard Disk	storage	macchina



Linguaggio macchina e assembler

Il linguaggio macchina

- Quando il programma è in esecuzione, è memorizzato nella memoria principale; esso è rappresentato da una serie di numeri binari che codificano le istruzioni eseguibili dall'unità centrale

```
00000000101000010000000000011000   <-- PC
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
```

- Il programma non è quindi distinguibile dai dati osservando il contenuto della memoria; le istruzioni sono individuate dai valori assunti dal registro PC durante l'esecuzione del programma
- Ogni codice binario codifica il tipo di istruzione (OPCODE) ed eventuali parametri (es. registri, indirizzi in memoria)
- I primi calcolatori si programmavano direttamente in linguaggio macchina!

Il set di istruzioni macchina

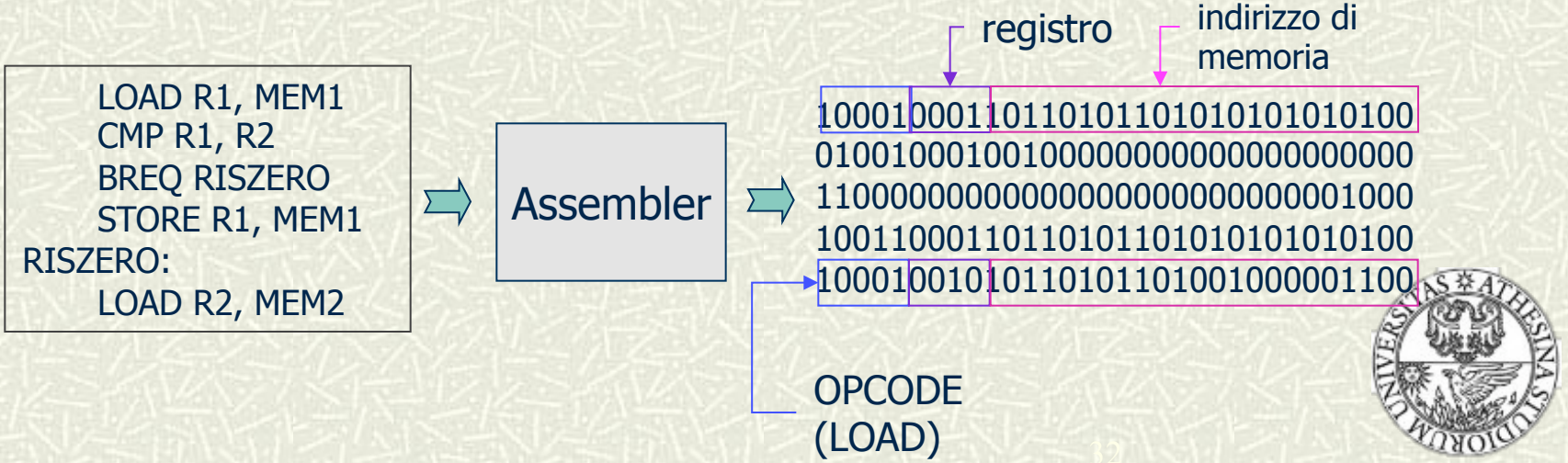
- L'insieme delle istruzioni eseguibili, e la relativa codifica, sono generalmente diverse per modelli diversi di processore
- Le istruzioni possono essere codificate con un numero variabile di bit (es. Pentium) o con un numero fisso (es. MIPS - 32 bit)
- Le categorie di istruzioni normalmente disponibili sono:
  - Trasferimento dati: spostano dati (byte, word) tra registri, memoria principale e dispositivi di ingresso/uscita (I/O)
  - Aritmetico-logiche: eseguono i calcoli nella ALU
  - Salti (condizionati e incondizionati): prendono decisioni e alterano la normale esecuzione sequenziale delle istruzioni

Esempio di programma in linguaggio macchina

0100000000010000	leggi un valore in ingresso e ponilo nella cella numero 16 (variabile x)
0100000000010001	leggi un valore e ponilo nella cella numero 17 (variabile y)
0100000000010010	leggi un valore e ponilo nella cella numero 18 (variabile z)
0100000000010011	leggi un valore e ponilo nella cella numero 19 (variabile r)
0000000000010000	carica il registro A con il contenuto della cella 16
0001000000010001	carica il registro B con il contenuto della cella 17
0110000000000000	somma i contenuti dei dei registri A e B
0010000000010100	copia il contenuto del registro A nella cella 20 (risultato, variabile s)
0000000000010010	carica il registro A con il contenuto della cella 18
0001000000010011	carica il registro B con il contenuto della cella 19
0110000000000000	somma i contenuti dei registi A e B
0001000000010100	carica il registro B con il contenuto della cella 20
1000000000000000	moltiplica i contenuti dei registri A e B
0010000000010100	copia il contenuto del registro A nella cella numero 20
0101000000010100	scrivi in output il contenuto della cella numero 20
1101000000000000	arresta l'esecuzione (HALT)
.....	spazio per la variabile x (cella 16)
.....	spazio per la variabile y (cella 17)
.....	spazio per la variabile z (cella 18)
.....	spazio per la variabile r (cella 19)
.....	spazio per la variabile s (cella 20)

Assembler

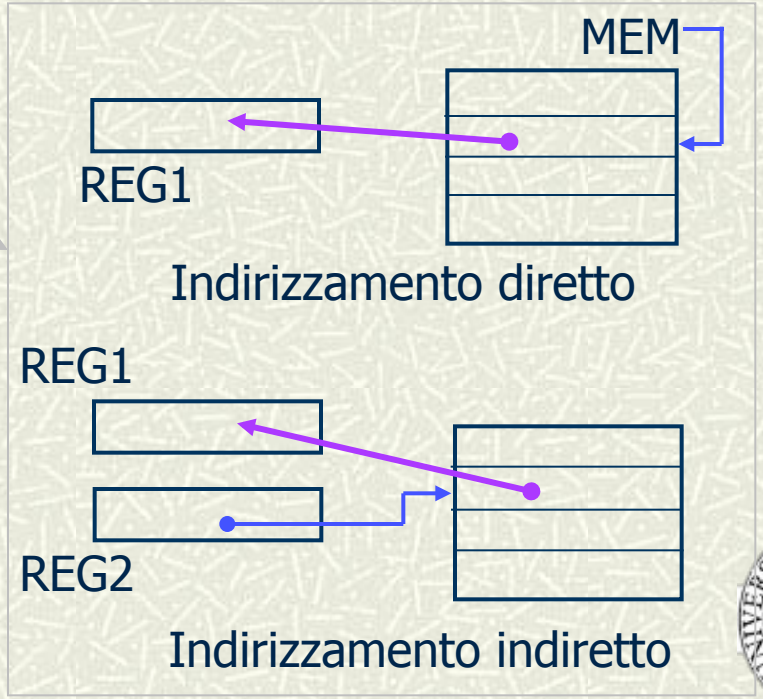
- Per facilitare la programmazione è stato definito il linguaggio **assembly**
- L'assembly impiega una notazione simbolica che è in stretta relazione con i codici in linguaggio macchina; il programma scritto in assembly è convertito automaticamente in linguaggio macchina per mezzo del programma traduttore, l'**assembler**



Trasferimento dei dati

- Le istruzioni di trasferimento dati permettono di copiare il valore di un dato fra registri o fra un registro e la memoria
- Si fa riferimento ad un assembly generico

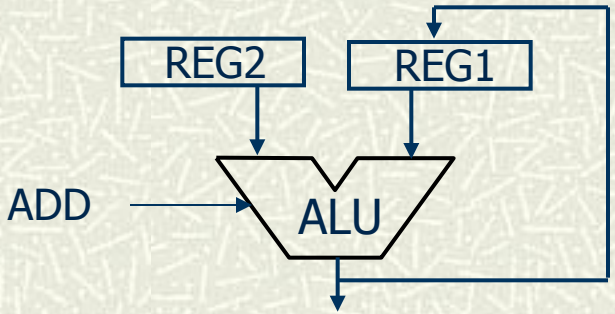
memoria registro	LOAD REG1, MEM LOAD REG1, [REG2]
registro memoria	STORE REG1, MEM STORE REG1, [REG2]
registro registro	MOVE REG1, REG2



Operazioni aritmetico-logiche

- Permettono di eseguire operazioni aritmetiche o logiche su due operandi

ADD REG1, REG2

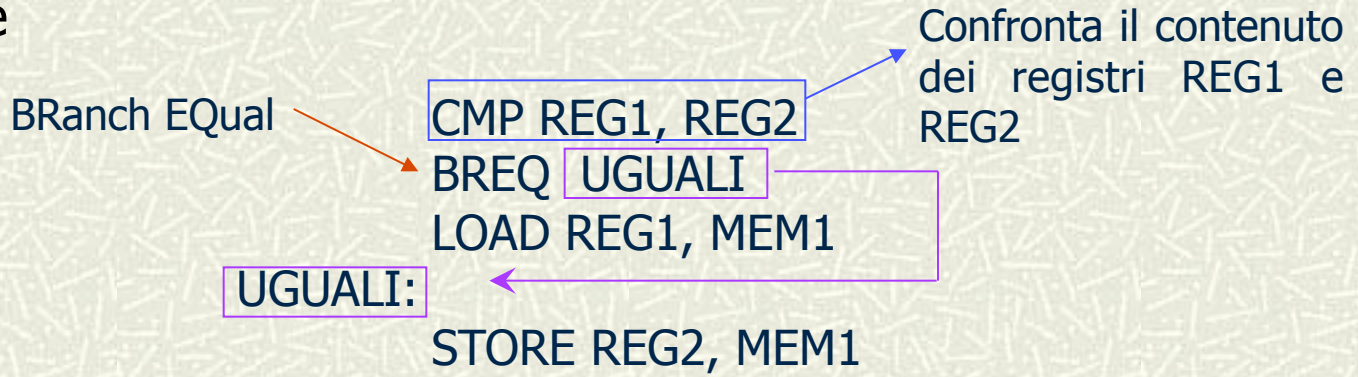


OR REG1, REG2

```
REG1  0010 1100 1100 0101 1100 1111 0101 0000
REG2  1010 1110 0000 0000 0010 0001 1000 0001
-----------------------------------------------
REG1  1010 1110 1100 0101 1110 1111 1101 0001
```

Istruzioni di salto

- Modificano il flusso di esecuzione del programma; il salto è **condizionato** se viene effettuato solo quando si verifica una certa condizione



- L'istruzione **CMP** confronta i due argomenti, assegnando valori particolari a bit di condizione che si trovano in un registro speciale della CPU (zero, segno)
- L'istruzione di salto condizionato può essere basata su diverse condizioni: **BRNE** (branch not equal), **BRLE** (branch less equal), etc.
- L'istruzione di salto incondizionato, **BRANCH**, effettua sempre il salto

Assembler e linguaggi di alto livello

- Il programma in assembler…

```
LOAD REG1, a
LOAD REG2, b
ADD REG1, REG2
LOAD REG3, c
LOAD REG4, d
ADD REG3, REG4
MULT REG1, REG3
STORE REG1, e
```

…corrisponde all'unica istruzione C:

```
e=(a+b)*(c+d);
```

Il sistema operativo

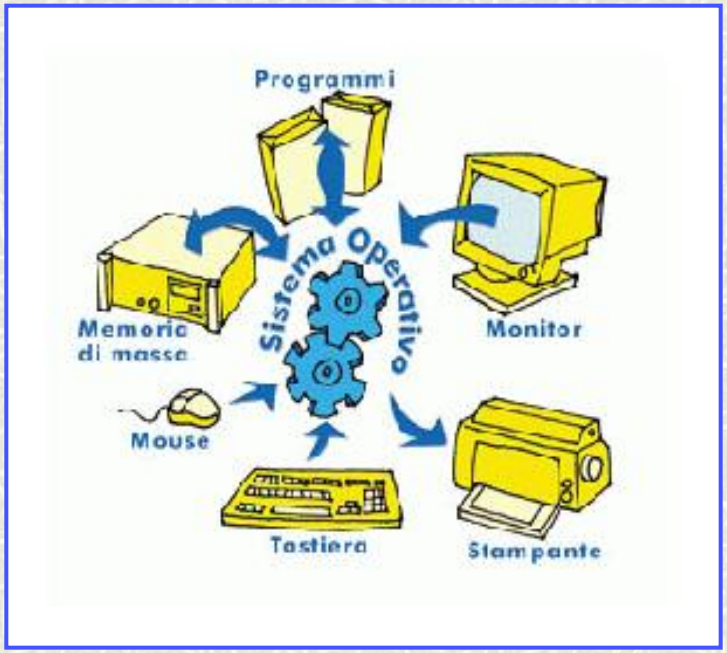
Cos'è un sistema operativo - 1

- Il software può essere diviso in due grandi classi:
  - i **programmi di sistema**, che gestiscono le funzionalità del sistema di calcolo
  - i **programmi applicativi**, che risolvono i problemi degli utenti
- L'insieme dei **programmi di sistema** viene comunemente identificato con il nome di **Sistema Operativo** (SO)
- **Definizione**

  Un sistema operativo è un programma che **controlla l'esecuzione di programmi applicativi** ed agisce come **interfaccia fra le applicazioni e l'hardware** del calcolatore

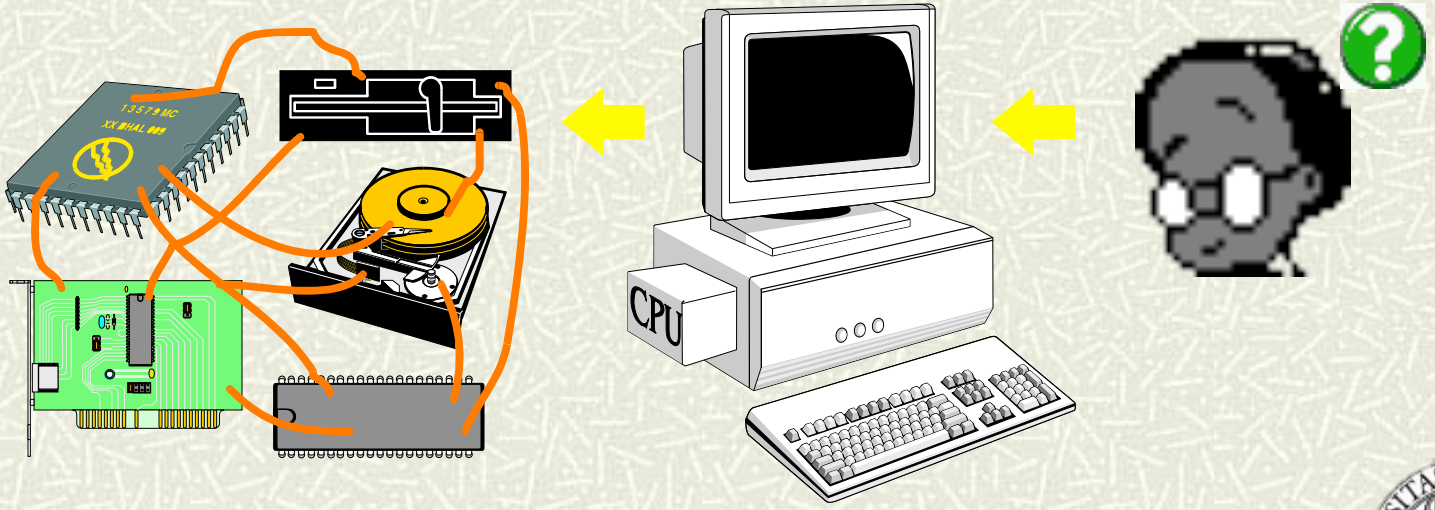
Cos'è un sistema operativo - 2

- Tutte le piattaforme hardware/software richiedono un sistema operativo
- Quando si accende un elaboratore, occorre attendere alcuni istanti per poter iniziare a lavorare: durante questa pausa il computer carica il SO



Scopo del sistema operativo

- Gestione EFFICIENTE delle risorse del sistema di elaborazione
- Rendere AGEVOLE l'interfaccia tra l'uomo e la macchina



Esempio: il SO come gestore risorse – 1

- Si consideri un ristorante con un capo-cuoco (che dirige la cucina) ed i suoi aiutanti, camerieri e clienti:
  - I clienti scelgono un piatto dal menù
  - Un cameriere prende l'ordine e lo consegna al capo-cuoco
  - Il capo-cuoco riceve l'ordine e assegna uno o più aiutanti alla preparazione del piatto
  - Ogni aiutante si dedicherà alla preparazione di un piatto, il che potrà richiedere più attività diverse
  - Il capo-cuoco supervisiona la preparazione dei piatti e gestisce le risorse (limitate) disponibili

Esempio: il SO come gestore risorse – 2

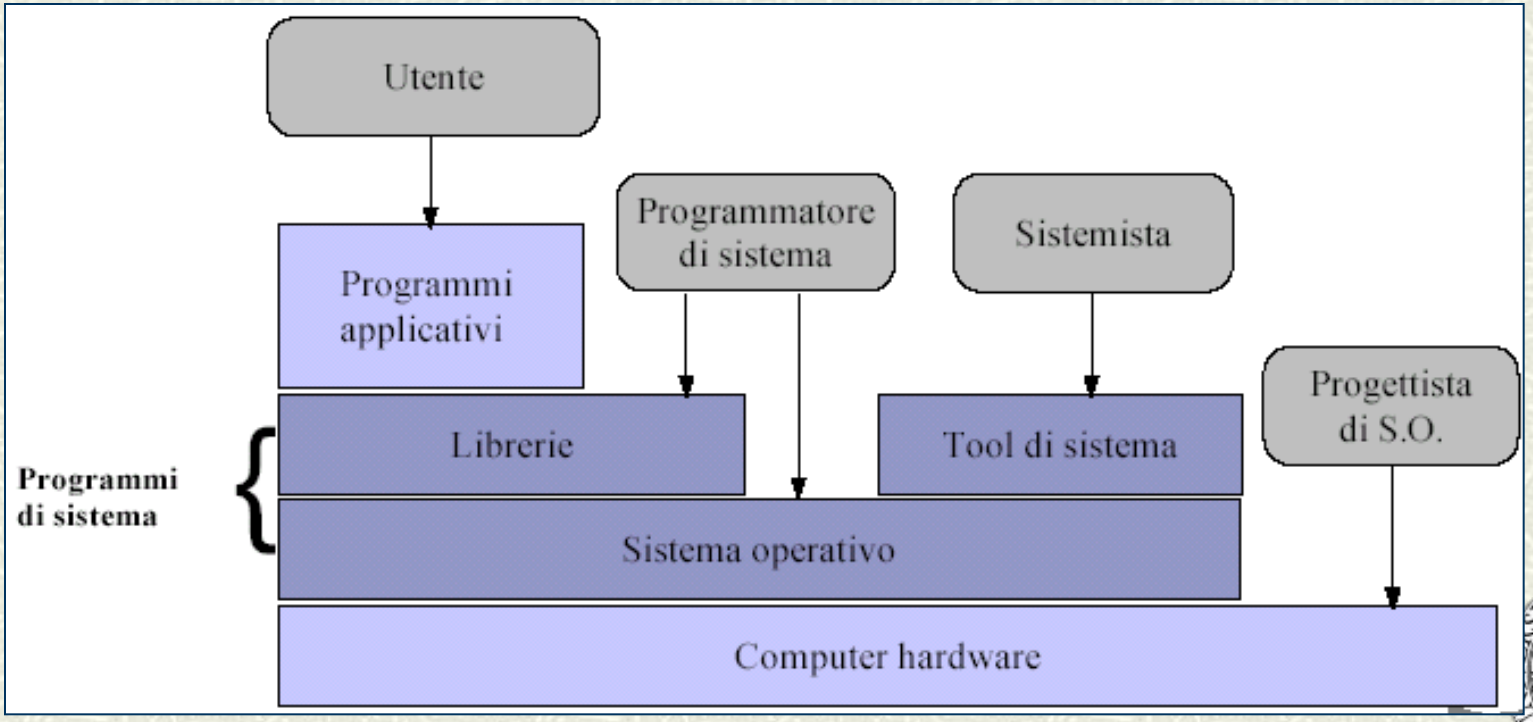
- Il capo-cuoco è il sistema operativo!
  - I clienti sono gli utenti
  - Le ricette associate ai piatti sono i programmi
  - Il menù ed il cameriere costituiscono l'interfaccia verso il sistema operativo (grafica e non)
  - Gli aiutanti sono i processi
  - La cucina è il computer; pentole, fornelli, etc. sono le componenti hardware

Esempio: il SO come gestore risorse – 3

- Problemi del capo-cuoco:
  - Esecuzione fedele delle ricette
  - Allocazione efficiente delle risorse esistenti (aiutanti, fornelli, etc.)
  - Coordinamento efficiente degli aiutanti
  - Licenziamento degli aiutanti che non si comportano secondo le regole
- Problemi del sistema operativo:
  - Esecuzione dei programmi utente
  - Efficienza nell'uso delle risorse (processori, memoria, dischi, etc.)
  - Coordinamento dei processi
  - Protezione nell'uso delle risorse

Il SO come macchina estesa – 1

- Visione a strati delle componenti hardware/software che compongo un sistema di elaborazione



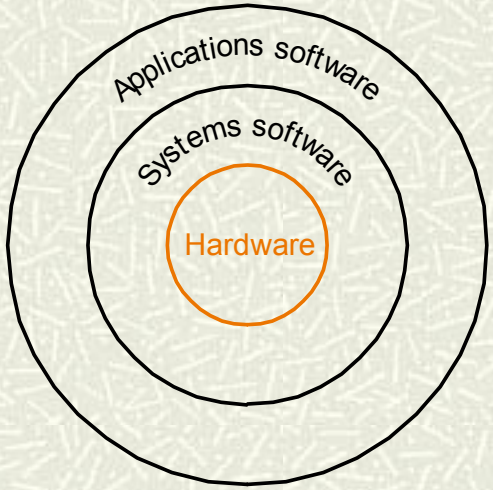
Il SO come macchina estesa – 2

- Il SO può essere inteso come uno strumento che **virtualizza** le caratteristiche dell'hardware sottostante, offrendo all'utente la visione di una macchina astratta più potente e più semplice da utilizzare di quella fisicamente disponibile
- In questa visione, un SO…
  - …nasconde a programmatori/utenti i dettagli dell'hardware e fornisce un'interfaccia conveniente e facile da usare
  - …agisce come intermediario tra programmatore/utente e hardware
- Parole chiave
  - Indipendenza dall'hardware
  - Comodità d'uso
  - Programmabilità

Il SO come macchina estesa – 3

- L'utente è in grado di utilizzare la macchina fisica senza conoscere i dettagli della sua struttura interna e del suo funzionamento

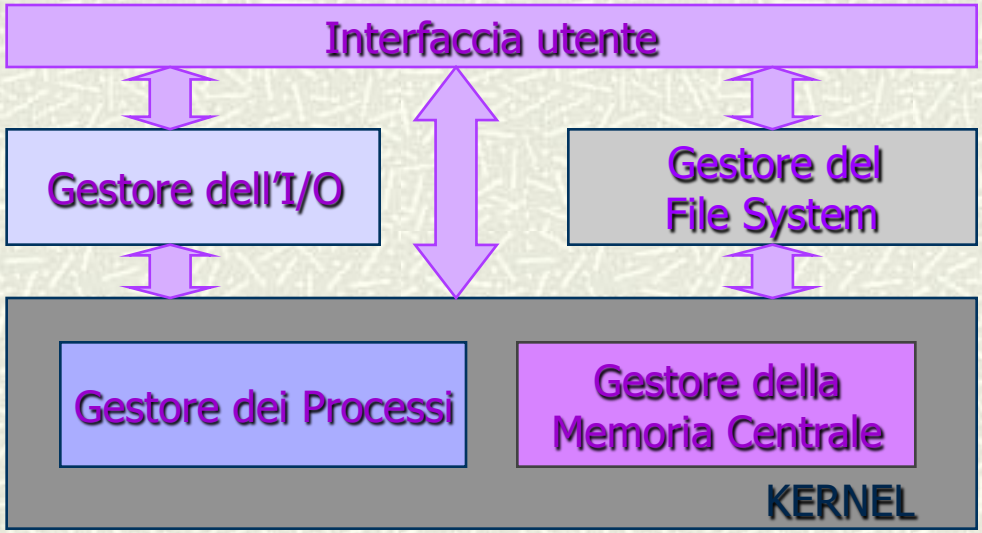
1. **Hardware** — fornisce le risorse fondamentali di calcolo (CPU, memoria, device di I/O)
2. **Sistema Operativo** — controlla e coordina l'utilizzo delle risorse hardware da parte dei programmi applicativi dell'utente
3. **Programmi Applicativi** — definiscono le modalità di utilizzo delle risorse del sistema, per risolvere i problemi di calcolo degli utenti (compilatori, database, video game, programmi gestionali)
4. **Utenti** — persone, altri macchinari, altri elaboratori



Visione "a cipolla" del sistema di calcolo

Architettura del sistema operativo

- I SO sono costituiti da un insieme di moduli, ciascuno dedicato a svolgere una determinata funzione
- I vari moduli del SO interagiscono tra loro secondo regole precise, al fine di realizzare le funzionalità di base della macchina



- L'insieme dei moduli per la gestione della CPU e della memoria centrale è il **kernel**

Ancora sul sistema operativo

- Riassumendo: Il sistema operativo fornisce un ambiente per eseguire programmi in modo *conveniente* ed *efficiente*; funge infatti da…
  - **Allocatore di risorse** — controlla, distribuisce ed alloca le risorse (in modo equo ed efficiente)
  - **Programma di controllo** — controlla l'esecuzione dei programmi utente e le operazioni sui dispositivi di I/O

Esempio: il filesystem

Si ha a che fare con file, directory, etc., e non ci si deve preoccupare di come i dati sono memorizzati sul disco

Compiti del sistema operativo

- Gestione dei processi
- Gestione della memoria principale
- Gestione della memoria di massa (file system)
- Realizzazione dell'interfaccia utente
- Protezione e sicurezza

La gestione dei processi – 1

- Un **processo** è un programma in esecuzione
  - Un processo utilizza le risorse fornite dal sistema di elaborazione per assolvere ai propri compiti
  - La terminazione di un processo prevede il recupero di tutte le risorse riutilizzabili ad esso precedentemente allocate
- Normalmente, in un sistema vi sono molti processi, di alcuni utenti, e alcuni sistemi operativi, che vengono eseguiti in concorrenza su una o più CPU
- La concorrenza è ottenuta effettuando il ***multiplexing*** delle CPU fra i vari processi

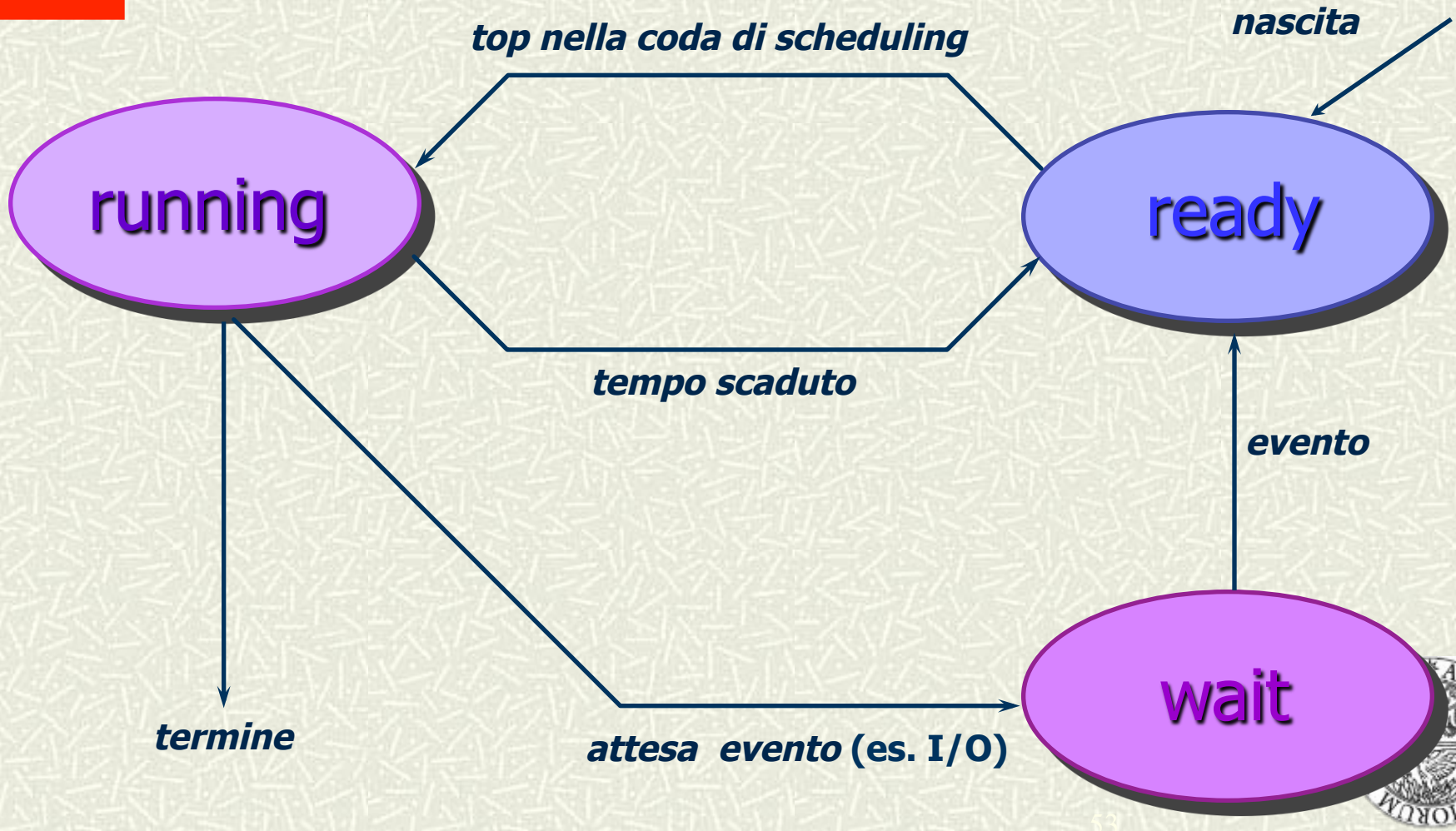
La gestione dei processi – 2

- Il sistema operativo è responsabile delle seguenti attività riguardanti la gestione dei processi:
  - creazione e terminazione dei processi
  - sospensione e riattivazione dei processi
  - gestione dei deadlock
  - comunicazione tra processi
  - sincronizzazione tra processi
- **Il gestore dei processi "realizza" una macchina virtuale in cui ciascun programma opera come se avesse a disposizione un'unità di elaborazione dedicata**

La gestione dei processi – 3

- Il gestore dei processi è il modulo che si occupa del controllo, della sincronizzazione, dell'interruzione e della riattivazione dei programmi in esecuzione cui viene assegnato un processore
- La gestione dei processi viene compiuta secondo modalità diverse, in funzione del tipo di utilizzo cui il sistema è rivolto
- Il programma che si occupa della distribuzione del tempo di CPU tra i vari processi attivi, decidendone l'avvicendamento, è chiamato scheduler
- Nel caso di sistemi multiprocessore, lo scheduler si occupa anche di gestire la cooperazione tra le diverse CPU presenti nel sistema (bilanciandone il carico)

Ciclo di vita dei processi

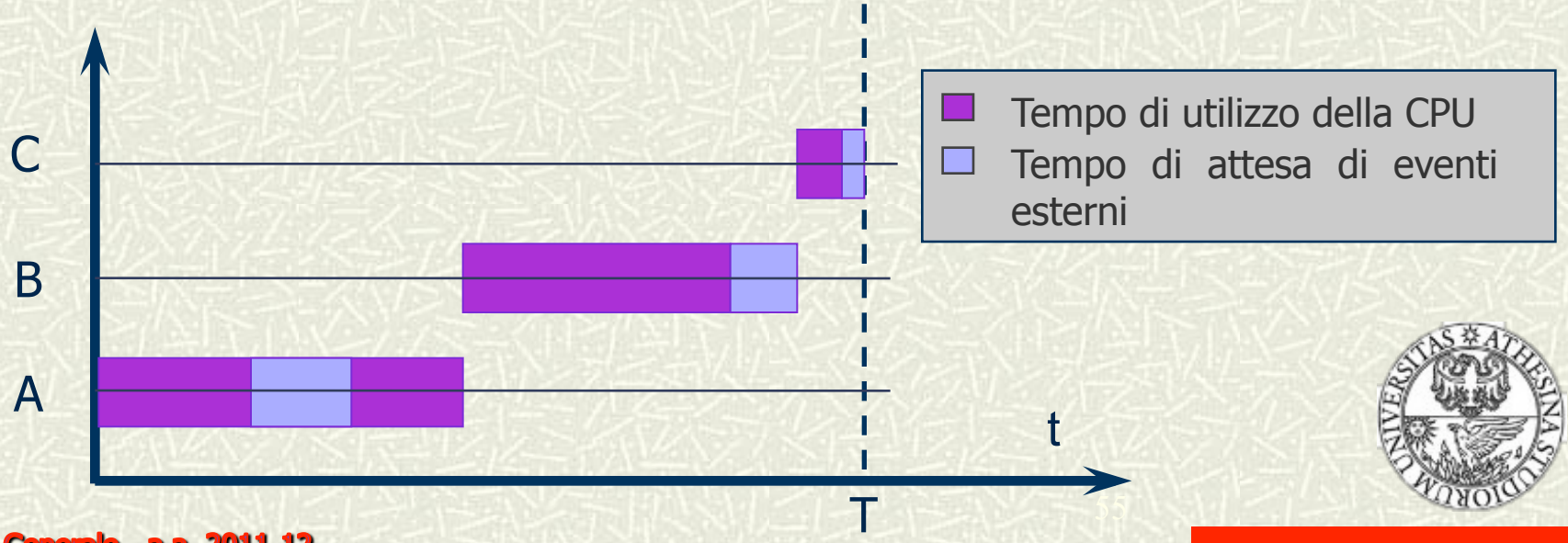


Politiche di scheduling

- Le politiche di scheduling sono raggruppabili in due grandi categorie:
  - **Preemptive**: l'uso della CPU da parte di un processo può essere interrotto in un qualsiasi momento, e la risorsa concessa ad altro processo
  - **Non preemptive**: una volta che un processo ha ottenuto l'uso della CPU, è unico proprietario della risorsa finché non ne decide il rilascio

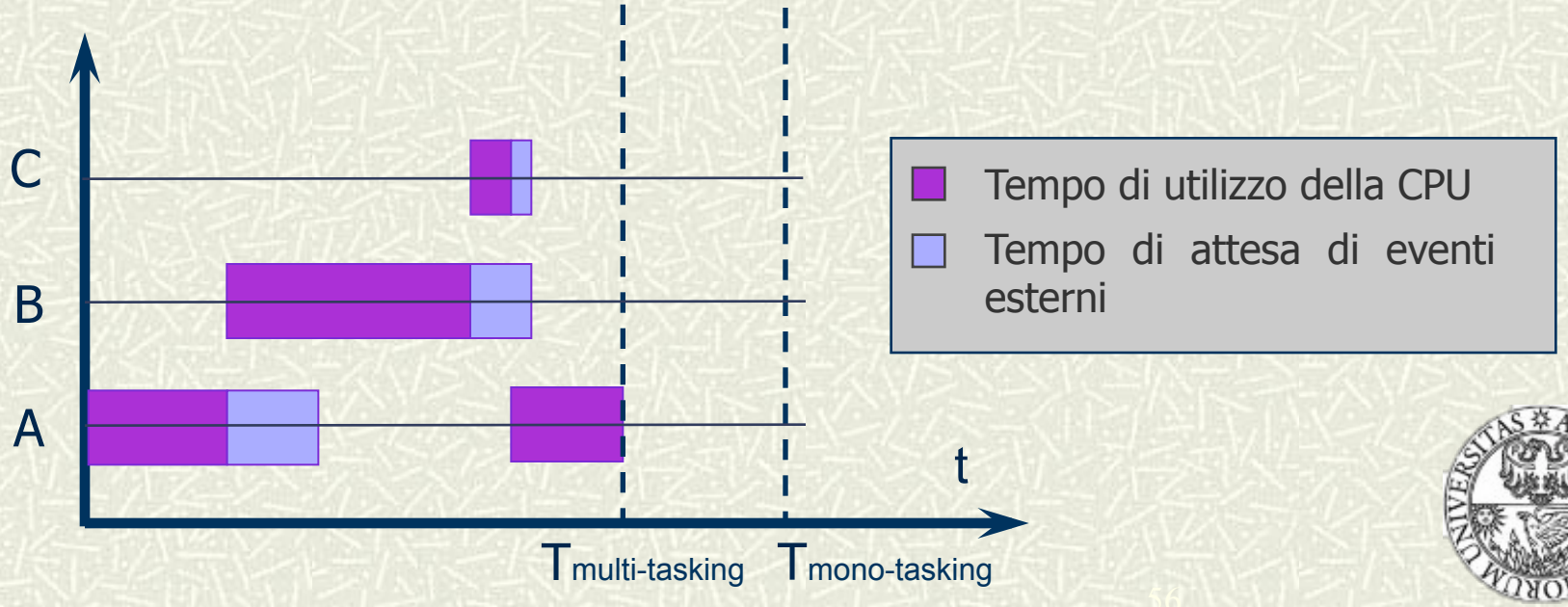
Sistemi mono–tasking

- I SO che gestiscono l'esecuzione di un solo programma per volta (un solo processo) sono detti mono-tasking
- Non è possibile sospendere un processo per assegnare la CPU ad un altro
- Sono storicamente i primi SO (es. MS-DOS)



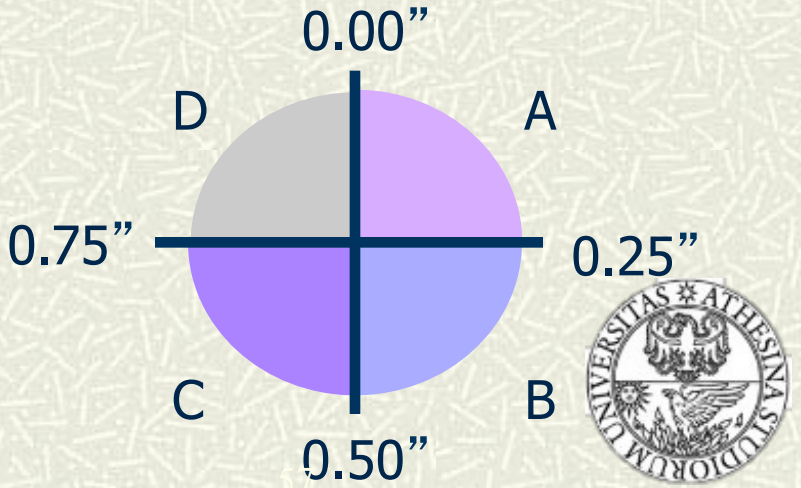
Sistemi multi–tasking

- I SO che permettono l'esecuzione contemporanea di più programmi sono detti **multi-tasking** o **multi-programmati**
- Un programma può essere interrotto e la CPU passata a un altro programma



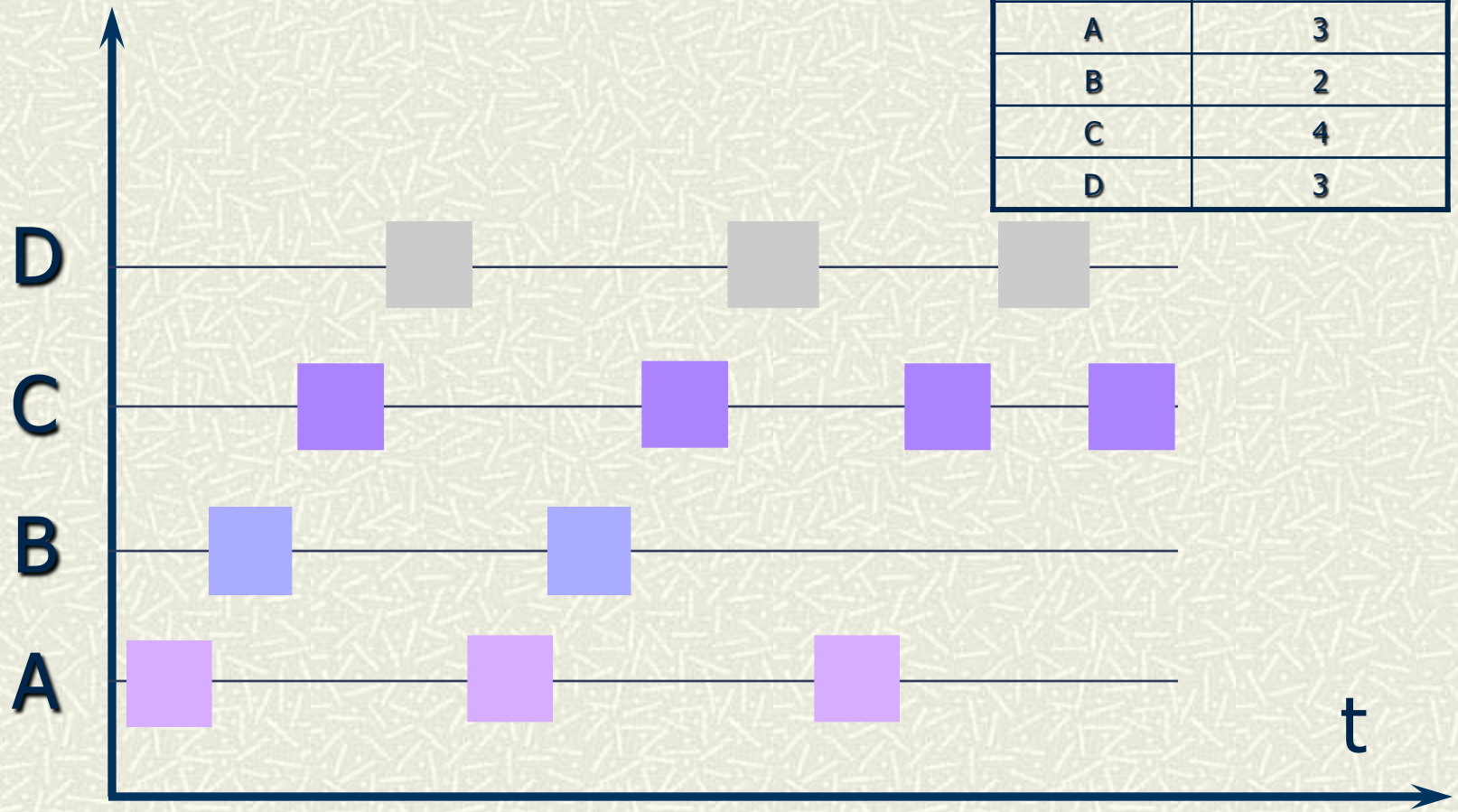
Sistemi time–sharing

- Un'evoluzione dei sistemi multi-tasking sono i sistemi time-sharing
- Ogni processo viene eseguito ciclicamente per piccoli quanti di tempo
- Se la velocità del processore è sufficientemente elevata si ha l'impressione di un'evoluzione parallela dei processi
- Esempio
  - Ipotesi: 1 MIPS, 4 processi,
    0.25 s/utente
  - Conseguenze: 0.25 MIPS/utente,
    $T_{ELA} = 4 \times T_{CPU}$



Time–sharing: diagramma temporale

Processo	Tempo di CPU
A	3
B	2
C	4
D	3



La gestione della memoria principale – 1

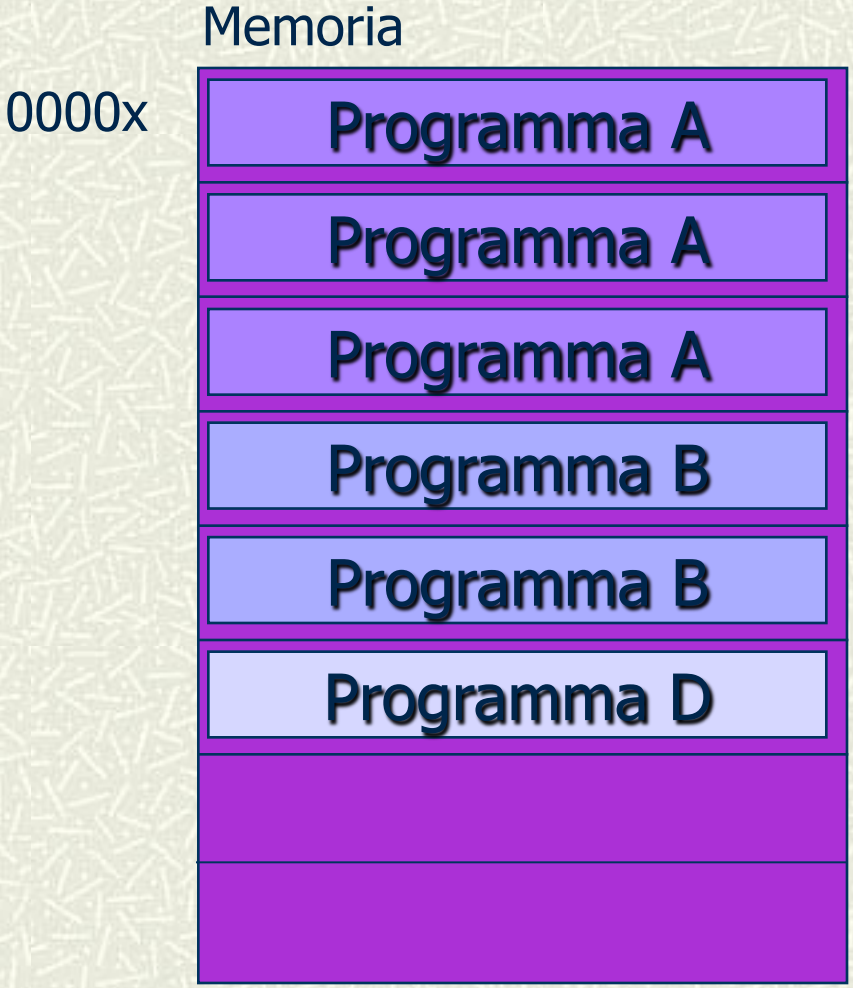
- La memoria principale…
  - …è un "array" di byte indirizzabili singolarmente
  - …è un deposito di dati facilmente accessibile e condiviso tra la CPU ed i dispositivi di I/O
- Il SO è responsabile delle seguenti attività riguardanti la gestione della memoria principale:
  - Tenere traccia di quali parti della memoria sono usate e da chi
  - Decidere quali processi caricare quando diventa disponibile spazio in memoria
  - Allocare e deallocare lo spazio di memoria quando necessario
- **Il gestore di memoria "realizza" una macchina virtuale in cui ciascun programma opera come se avesse a disposizione una memoria dedicata**

La gestione della memoria principale – 2

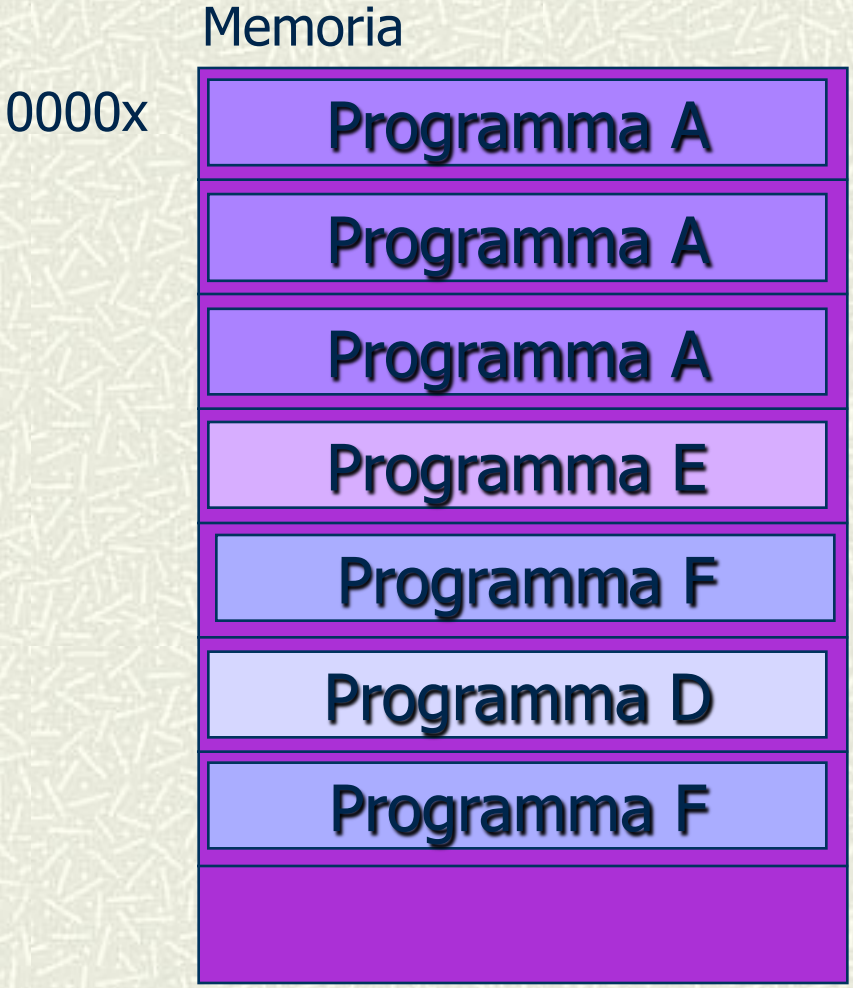
- L'organizzazione e la gestione della memoria centrale è uno degli aspetti più critici nel disegno di un SO
- Il gestore della memoria è quel modulo del SO incaricato di assegnare la memoria ai task (per eseguire un task è necessario che il suo codice sia caricato in memoria)
- La complessità del gestore della memoria dipende dal tipo di SO
- Nei SO multi-tasking, più programmi possono essere caricati contemporaneamente in memoria
- Problema: come allocare lo spazio in maniera ottimale

Allocazione lineare

Paginazione



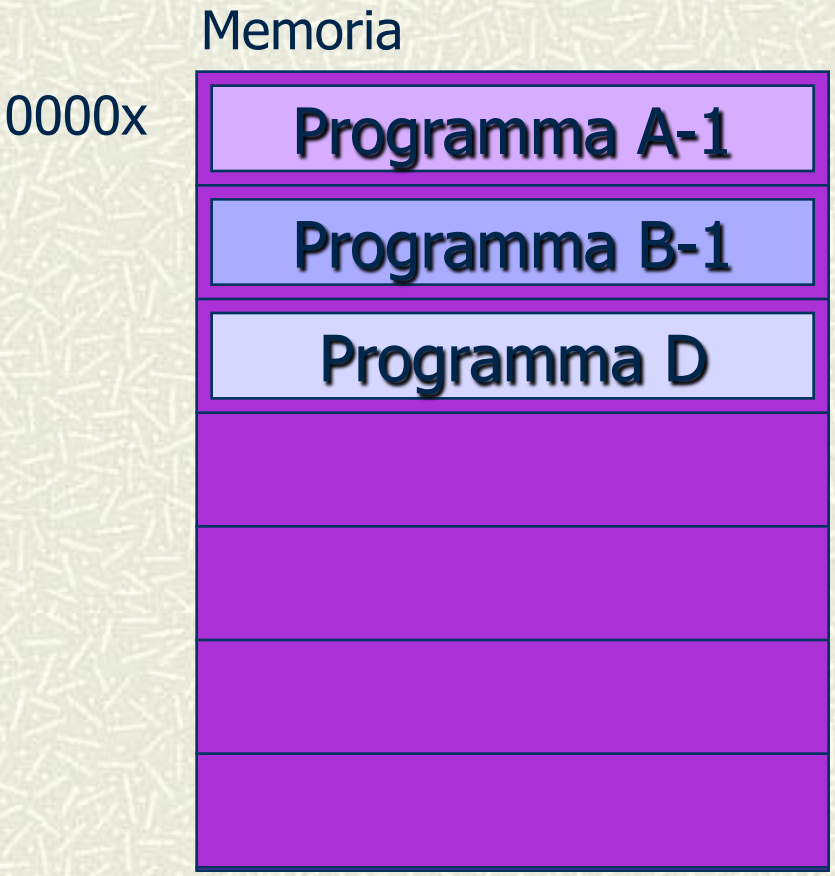
Paginazione

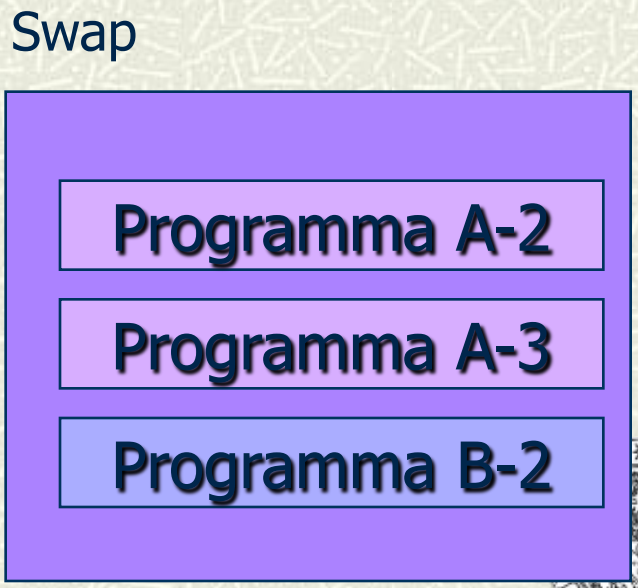


La memoria virtuale – 1

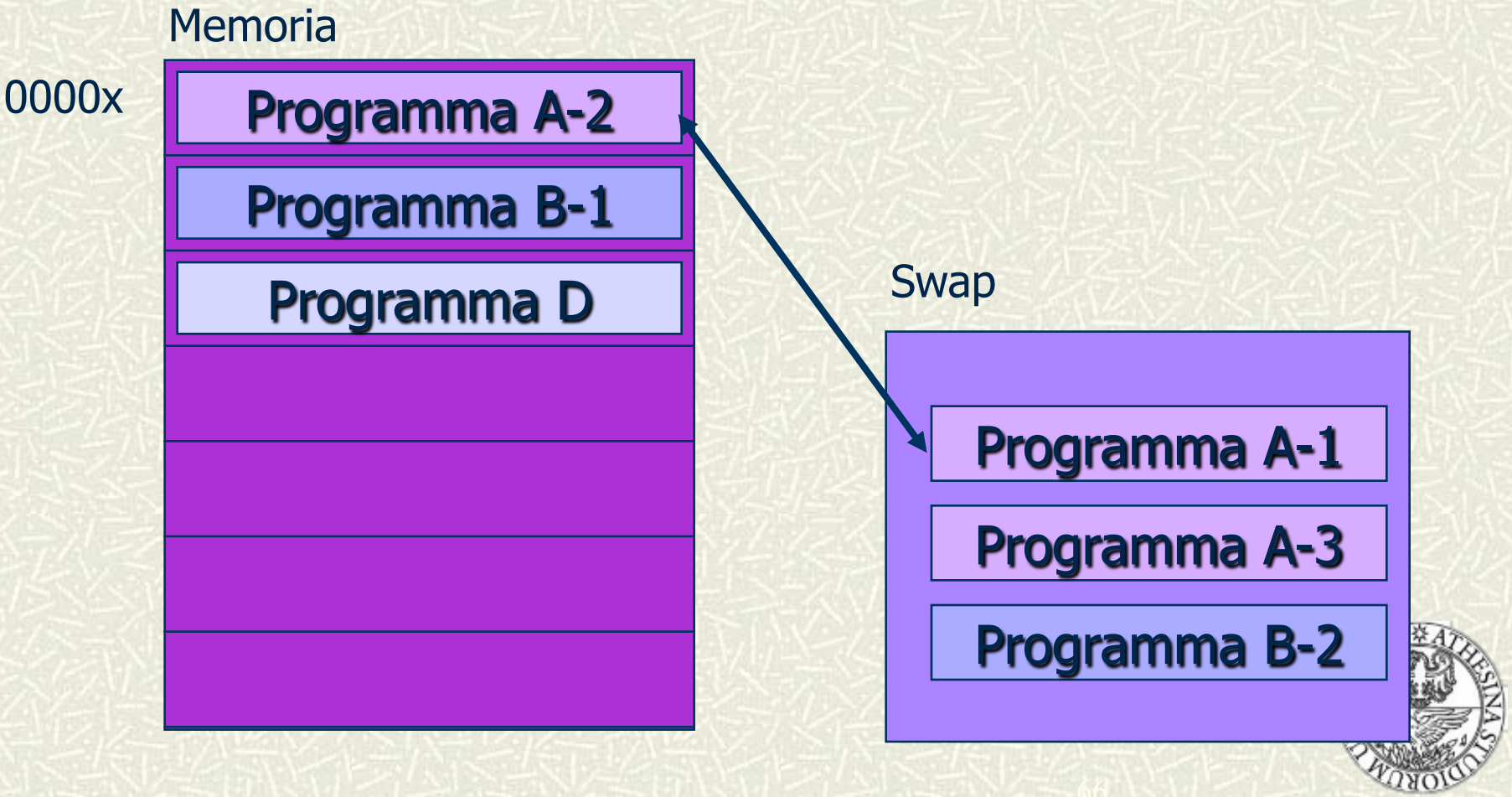
- Spesso la memoria non è sufficiente per contenere completamente tutto il codice dei processi
- Si può ***simulare*** una memoria più grande tenendo nella memoria di sistema (RAM) solo le parti di codice e dati che servono in quel momento
- Si usa il concetto di memoria virtuale
- I dati e le parti di codice relativi a programmi non in esecuzione possono essere tolti dalla memoria centrale e "parcheggiati" su disco nella cosiddetta area di swap
- I processori moderni sono dotati di meccanismi hardware per facilitare la gestione della memoria virtuale

La memoria virtuale – 2





La memoria virtuale – 2



La gestione della memoria secondaria - 1

- Poiché la memoria principale è volatile e troppo piccola per contenere tutti i dati e tutti i programmi in modo permanente, un computer è dotato di memoria secondaria
  - In generale, la memoria secondaria è data da hard disk e dischi ottici
- Il SO garantisce una visione logica uniforme del processo di memorizzazione:
  - Astrae dalle caratteristiche fisiche dei dispositivi per definire un'unità di memorizzazione logica - il *file*
  - Ciascuna periferica viene controllata dal relativo device driver, che nasconde all'utente le caratteristiche fisiche variabili dell'hardware: modalità e velocità di accesso, capacità, velocità di trasferimento

La gestione della memoria secondaria - 2

- Il SO è responsabile delle seguenti attività riguardanti la gestione della memoria secondaria:
  - Allocazione dello spazio
  - Gestione dello spazio libero
  - Ordinamento efficiente delle richieste di accesso al disco (**disk scheduling**)

La gestione del file system – 1

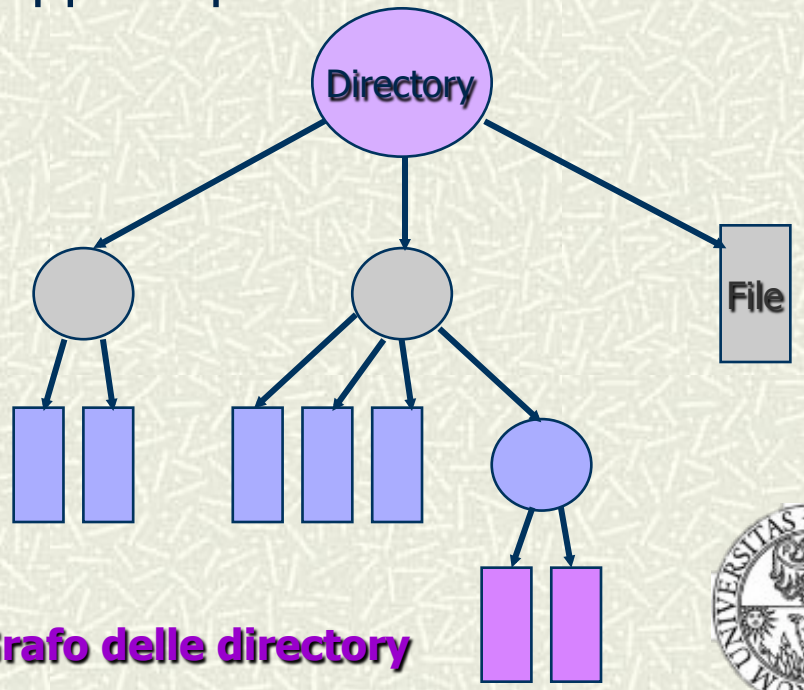
- Un file è l'astrazione informatica di un archivio di dati
  - Il concetto di file è indipendente dal mezzo sul quale viene memorizzato (che ha caratteristiche proprie e propria organizzazione fisica)
- Un file system è composto da un insieme di file
- Il SO è responsabile delle seguenti attività riguardanti la gestione del file system:
  - Creazione e cancellazione di file
  - Creazione e cancellazione di directory
  - Manipolazione di file e directory
  - Codifica del file system sulla memoria secondaria

La gestione del file system – 2

- Il gestore del file system è il modulo del SO incaricato di gestire le informazioni memorizzate sui dispositivi di memoria di massa
- Il gestore del file system deve garantire la correttezza e la coerenza delle informazioni
- Nei sistemi multi-utente, fornisce meccanismi di protezione per consentire agli utenti di proteggere i propri dati dall'accesso di altri utenti non autorizzati
- Le funzioni tipiche del gestore del file system sono:
  - Fornire un meccanismo per l'identificazione dei file
  - Fornire metodi opportuni di accesso ai dati
  - Rendere trasparente la struttura fisica del supporto di memorizzazione
  - Implementare meccanismi di protezione dei dati

Organizzazione del file system

- Quasi tutti i SO utilizzano un'organizzazione **gerarchica** del file system
- L'elemento utilizzato per raggruppare più file insieme è la *directory*
- L'insieme gerarchico delle directory e dei file può essere rappresentato attraverso un grafo (un albero nei SO più datati) delle directory



Grafo delle directory

La gestione dei dispositivi di I/O

- La gestione dell'I/O richiede:
  - Un'interfaccia comune per la gestione dei ***device driver***
  - Un insieme di driver per dispositivi hardware specifici
  - Un sistema di gestione di buffer per il caching delle informazioni
- Il gestore dei dispositivi di I/O è il modulo del SO incaricato di assegnare i dispositivi ai task che ne fanno richiesta e di controllare i dispositivi stessi
- Da esso dipende la qualità e il tipo di periferiche riconosciute dal sistema
- **Il gestore delle periferiche offre all'utente una versione astratta delle periferiche hardware; l'utente ha a disposizione un insieme di procedure standard di alto livello per leggere/scrivere da/su una periferica che "percepisce" come dedicata**

Device driver

- Il controllo dei dispositivi di I/O avviene attraverso speciali moduli software, detti *device driver*
- I device driver sono spesso realizzati dai produttori dei dispositivi stessi, che ne conoscono le caratteristiche fisiche in maniera approfondita
- I device driver implementano le seguenti funzioni:
  - Rendono trasparenti le caratteristiche fisiche tipiche di ogni dispositivo
  - Gestiscono la comunicazione dei segnali verso i dispositivi
  - Gestiscono i conflitti, nel caso in cui due o più task vogliano accedere contemporaneamente allo stesso dispositivo

L' interfaccia utente – 1

- Tutti i SO implementano meccanismi per facilitare l'utilizzo del sistema da parte degli utenti
- L'insieme di tali meccanismi di accesso al computer prende il nome di interfaccia utente
- Serve per…
  - …attivare un programma, terminare un programma, etc.
  - …interagire con le componenti del sistema operativo (gestore dei processi, file system, etc.)

L' interfaccia utente – 2

- Interfaccia testuale:
  - Interprete dei comandi (shell)
  - Esempio: MS-DOS/UNIX
- Interfaccia grafica (a finestre):
  - L'output dei vari programmi viene visualizzato in maniera grafica all'interno di finestre
  - L'utilizzo di grafica rende più intuitivo l'uso del calcolatore
  - Esempio: WINDOWS/Linux
- Differenze:
  - Cambia il "linguaggio" utilizzato, ma il concetto è lo stesso
  - Vi sono però differenze a livello di espressività

L' interfaccia grafica

- Realizza la metafora della scrivania (desktop)
  - Interazione semplice via mouse
  - Le icone rappresentano file, directory, programmi, azioni, etc.
  - I diversi tasti del mouse, posizionato su oggetti differenti, provocano diversi tipi di azione: forniscono informazioni sull'oggetto in questione, eseguono funzioni tipiche dell'oggetto, aprono directory - ***folder***, o ***cartelle***, nel gergo GUI (***Graphical User Interface***)

Protezione e sicurezza – 1

- Protezione — è il meccanismo usato per controllare l'accesso da parte di processi e/o utenti a risorse del sistema di calcolo
- Sicurezza — è il meccanismo di difesa implementato dal sistema per proteggersi da attacchi interni ed esterni
  - Denial-of-service, worm, virus, hacker
- In prima istanza, il sistema distingue gli utenti, per determinare chi può fare cosa
  - L'identità utente (user ID) include nome dell'utente e numero associato - uno per ciascun utente
  - L'user ID garantisce l'associazione corretta di file e processi all'utente e ne regola la manipolazione
  - L'identificativo di gruppo permette inoltre ad un insieme di utenti di accedere correttamente ad un gruppo di risorse comuni (file e processi)

Protezione e sicurezza – 2

- In Linux...

$ ls -l

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Sistemi operativi commerciali

- In commercio sono presenti una grande quantità di sistemi operativi diversi
- In passato, la tendenza delle case costruttrici di sistemi di elaborazione era di sviluppare sistemi operativi proprietari per le loro architetture
- La tendenza attuale è quella dello sviluppo di sistemi operativi **portabili** su piattaforme hardware diverse

Windows/Vista

- CPU Intel (da 80386), ma anche per DEC-AXP, MIPS-R4000, etc.
- Multitask
- Monoutente/Multiutente
- NTFS (NT File System)
- Microkernel, thread
- Sistema a 32/64 bit

Linux

- Nato nel '91, grazie a Linus Torvalds, studente finlandese dell'Università di Helsinki
- Sviluppato su piattaforma Intel 80386, fu distribuito da subito su Internet (**free** e **open-source**)
- Multitask
- Multiutente
- L'architettura del sistema è Unix-like: un kernel molto "piccolo" che contiene solo funzioni fondamentali per la gestione delle risorse del computer (CPU, memoria, dischi, rete, I/O) ed una larga collezione di programmi di sistema che l'utente usa per richiedere servizi al SO