v 1.0
20120919

Sistemi Operativi

II Facoltà di Ingegneria - Cesena
a.a 2012/2013
docente: Alessandro Ricci

[modulo 1b]

ARCHITETTURA DEI SISTEMI DI ELABORAZIONE - RICHIAMI

DESCRIZIONE MODULO

- Questo modulo fornisce alcuni richiami sugli elementi fondamentali di un sistema di elaborazione moderno, mettendo in evidenza in particolare quegli aspetti che sono importanti per i Sistemi Operativi
- Questi aspetti verranno approfonditi nel corso di Calcolatori Elettronici

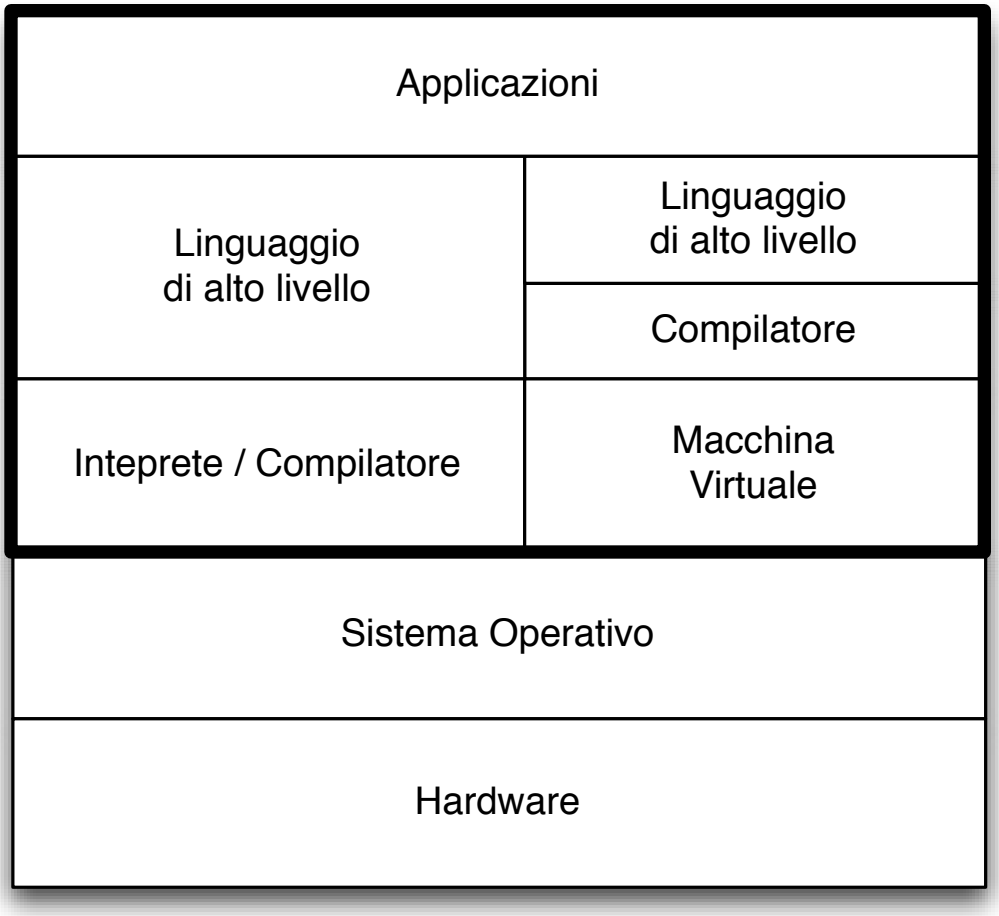
SISTEMI DI ELABORAZIONE: UNO SGUARDO D'INSIEME

- Sistemi di elaborazione moderni
  - software + hardware
    - a livello più alto: linguaggi di programmazione di alto livello, applicazioni articolate, sistemi software complessi
    - a livello più basso: circuiti elettronici, porte logiche, logica binaria
  - complessità della progettazione e realizzazione
- Organizzazione a livelli
  - decomposizione in **moduli**
    - ogni modulo confina un certo insieme di funzionalità
    - processo *di astrazione*
      - fornisce una interfaccia per fruire delle funzionalità
      - nasconde dettagli implementativi
  - organizzazione **gerarchica**
    - ogni modulo corrisponde ad un livello

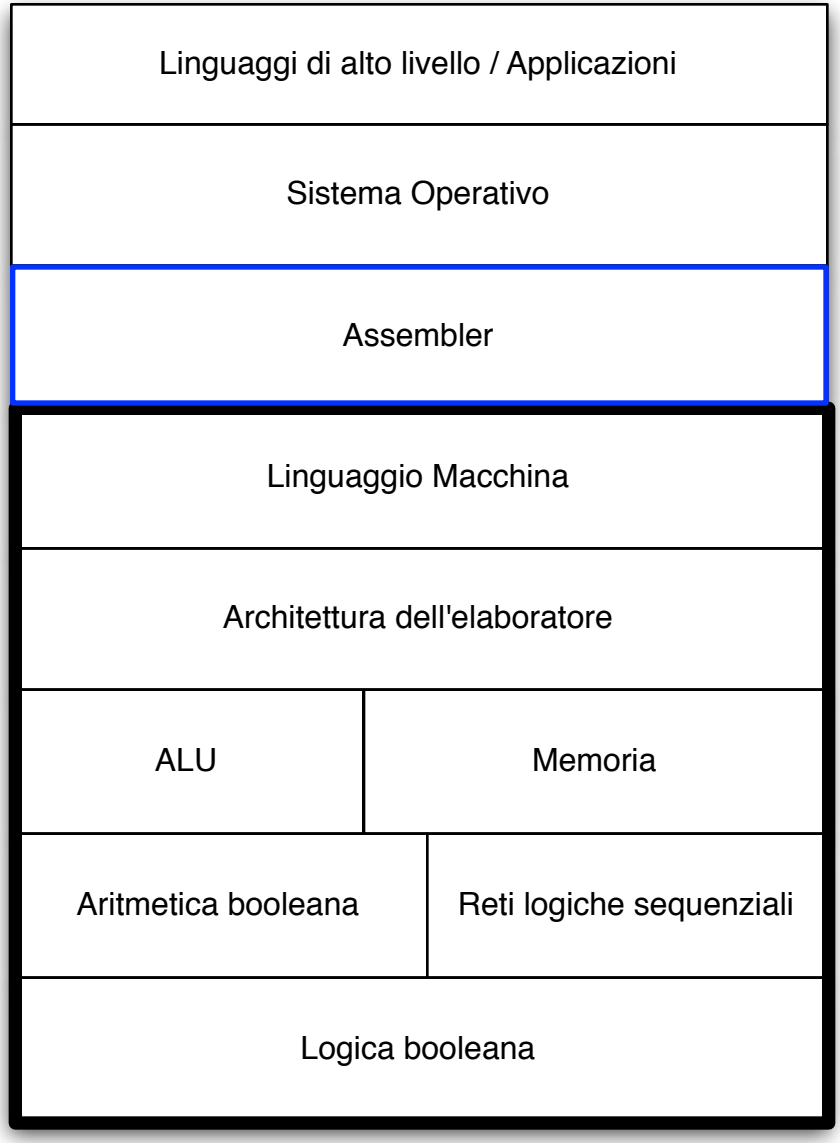
LIVELLI


Linguaggi di alto livello / Applicazioni
Sistema Operativo
Hardware

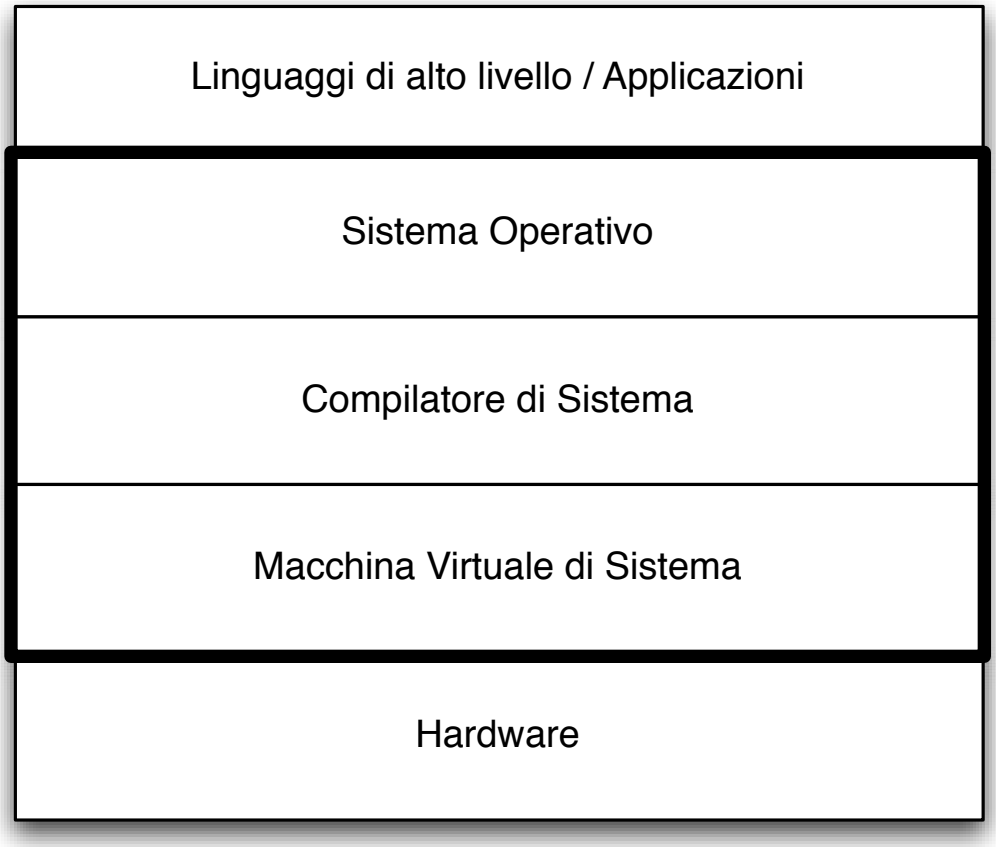
ZOOM LIVELLO LINGUAGGI/APPLICAZIONI



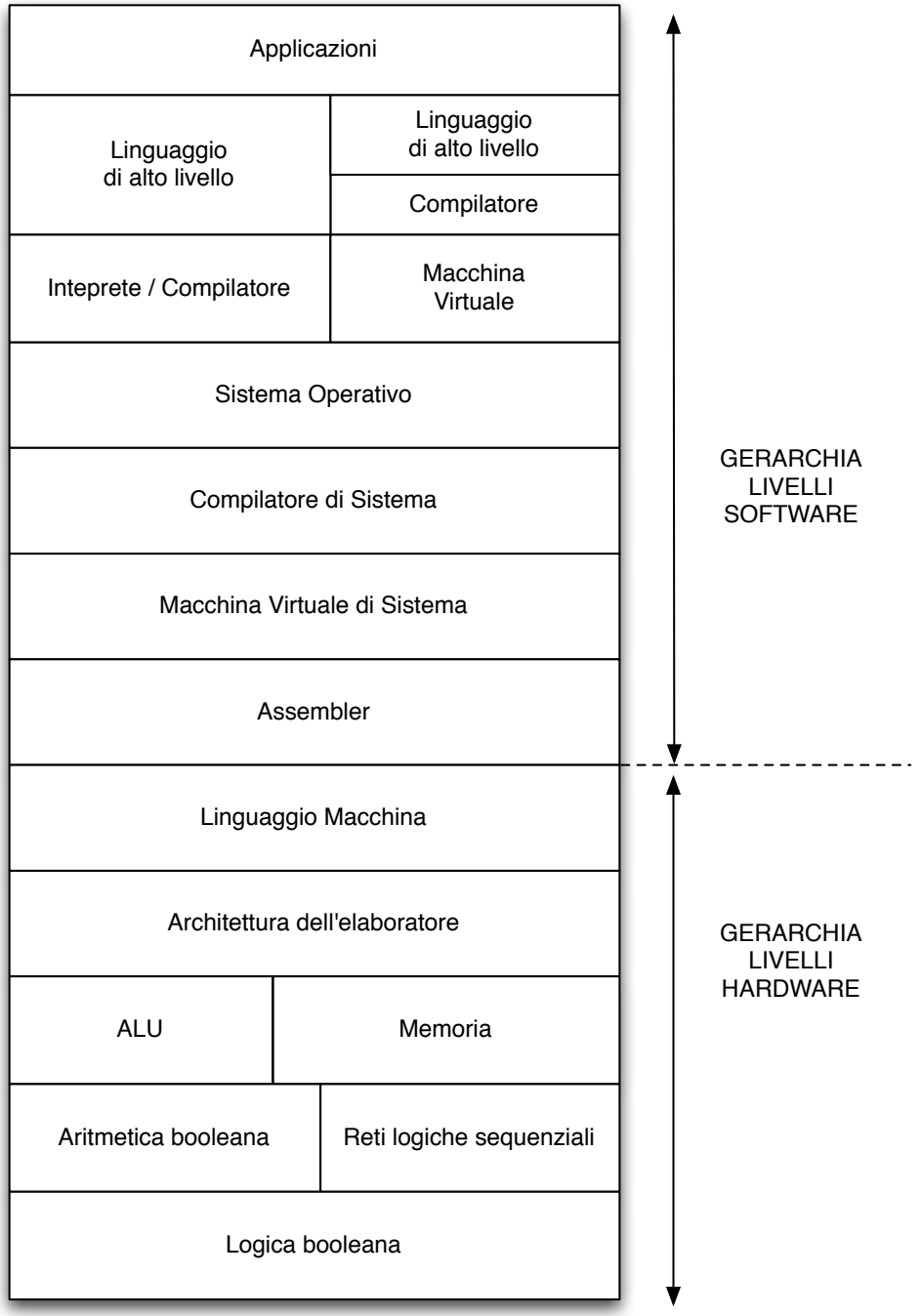
ZOOM LIVELLI HARDWARE



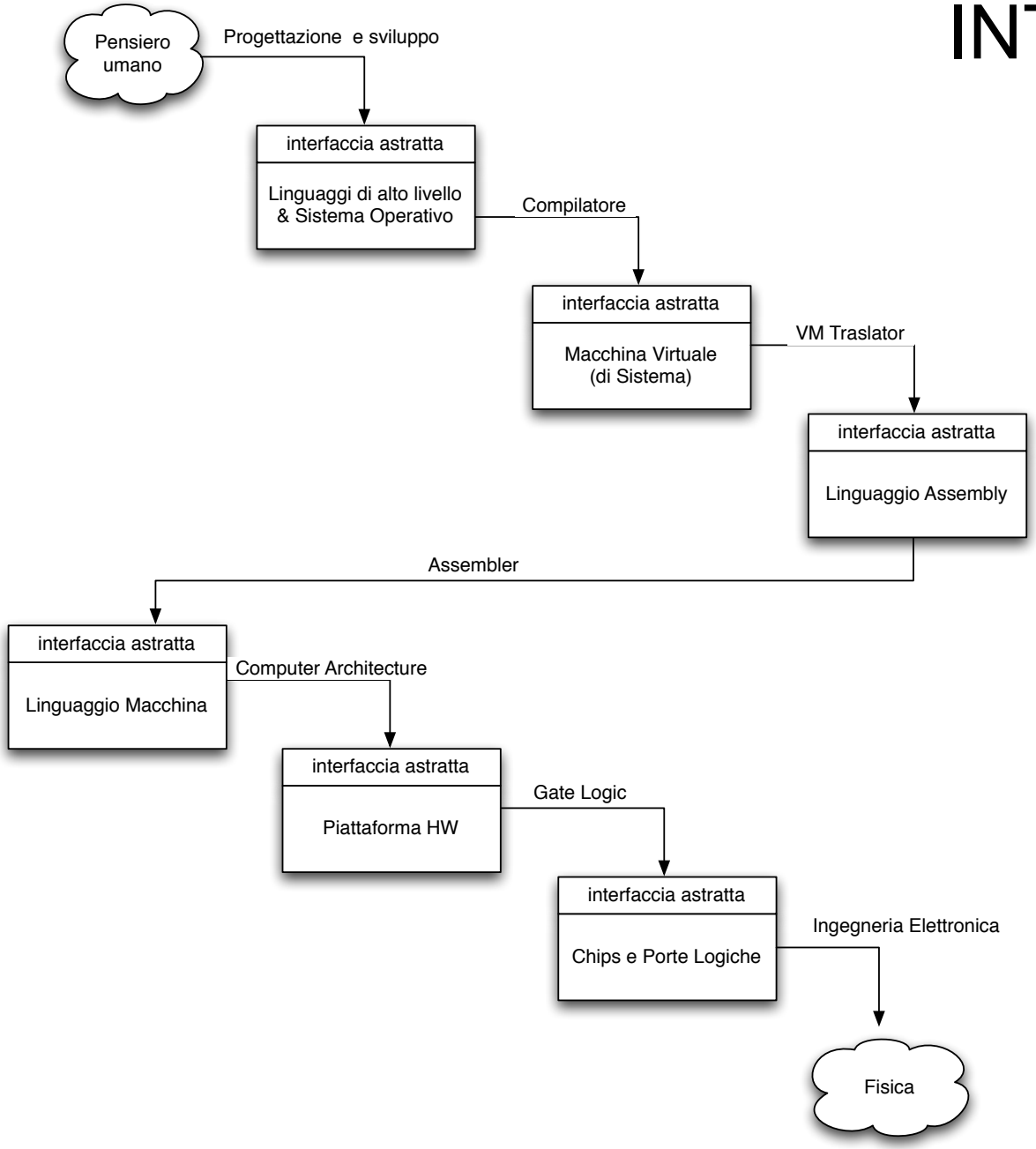
ZOOM LIVELLI VIRTUALIZZAZIONE



VISTA COMPLESSIVA DEI LIVELLI



INTERFACCE


Pensiero umano
Progettazione e sviluppo
interfaccia astratta
Linguaggi di alto livello & Sistema Operativo
Compilatore
interfaccia astratta
Macchina Virtuale (di Sistema)
VM Traslator
interfaccia astratta
Linguaggio Assembly
Assembler
interfaccia astratta
Linguaggio Macchina
Computer Architecture
interfaccia astratta
Piattaforma HW
Gate Logic
interfaccia astratta
Chips e Porte Logiche
Ingegneria Elettronica
Fisica

LIVELLI E INTERFACCE: DETTAGLIO

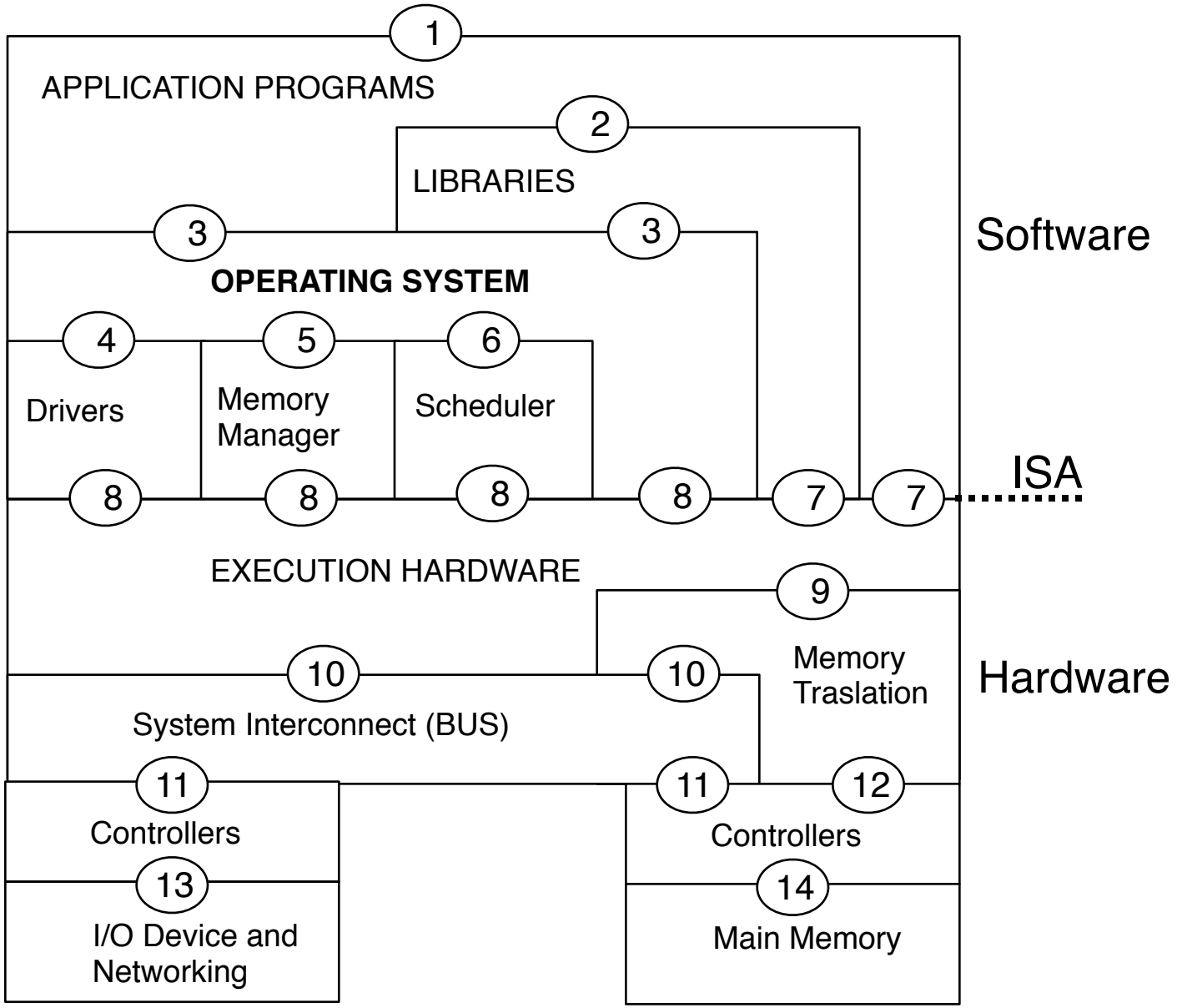
MAIN INTERFACES

ISA
- System ISA: 8
- User ISA: 7 + 8

Application Binary Interface (**ABI**): 7+3

System Call Interface: 3

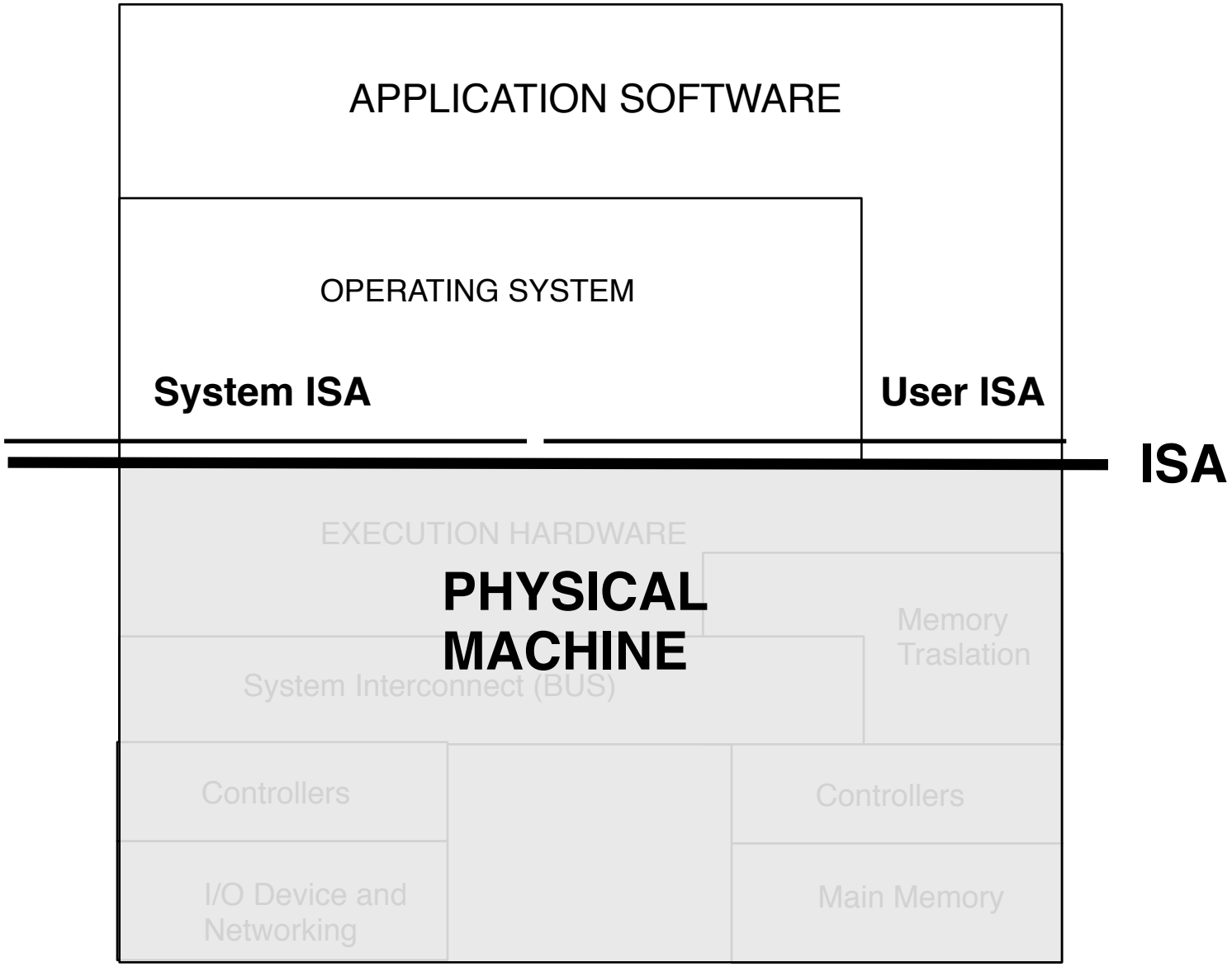
Application Programming Interface (**API**) for High-Level Languages (HLL) (es: C, Java,...): 2



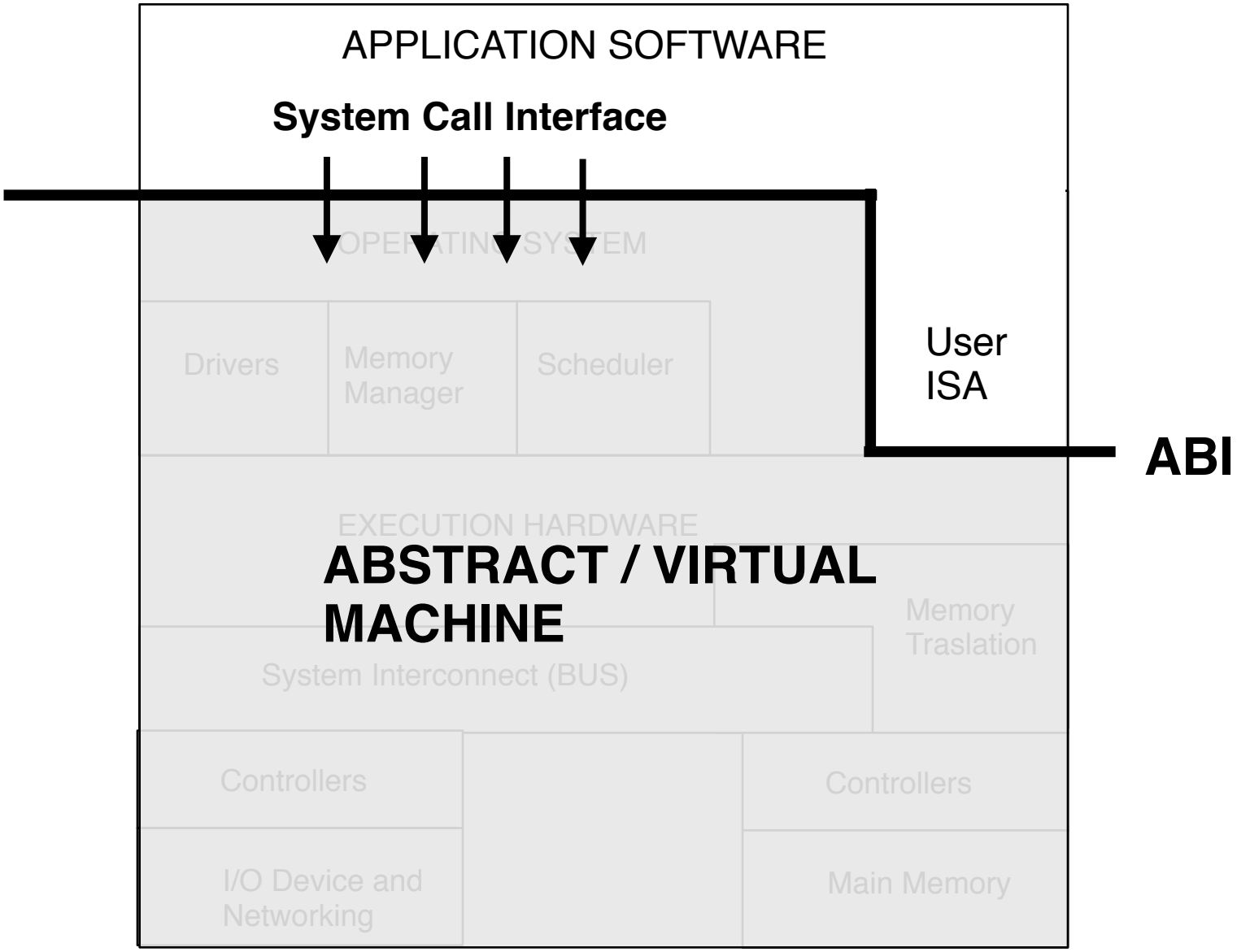
ISA, ABI e System Call Interface

- Instruction Set Architecture (**ISA**)
  - interfaccia che separa livelli HW e SW
  - *user ISA*
    - tutti gli aspetti e parti (es: istruzioni) visibili ai programmi utente
    - interfaccia 7
  - *system ISA*
    - tutti gli aspetti e parti visibili e utilizzabili solo dall'OS (es: istruzioni *privilegiate*)
    - interfaccia 8
- Application Binary Interface (**ABI**)
  - interfaccia fornita ai programmi per accedere alle risorse HW e ai servizi del sistema
  - due componenti principali: user ISA + System Call Interface
- System Call interface
  - insieme di operazioni che l'OS mette a disposizione ai programmi come servizi fondamentali
    - trasferimento del controllo, cambio livello di privilegio, ,,,

MACHINE INTERFACE: ISA

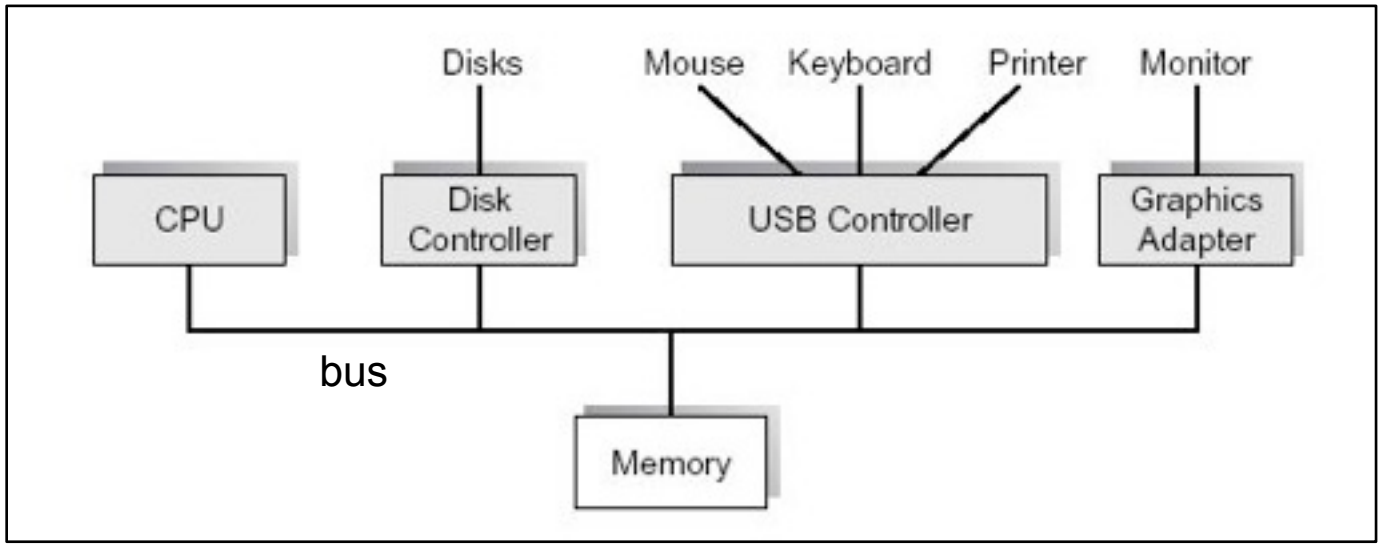


MACHINE INTERFACE: ABI



ARCHITETTURA HARDWARE

- L'architettura di un moderno e general-purpose computer è data da una **CPU** (Central Processing Unit) e da un insieme di **device controller** (controllore di dispositivi) e **adapters** connessi attraverso un **bus** comune (*bus di sistema*) che ne abilita l'interazione con la **memoria centrale** (condivisa)



- Il modello astratto è la **macchina di Von Neumann**
  - la memoria contiene dati e programma
  - la CPU carica ed esegue istruzioni dalla memoria
    - tale esecuzione comporta interazioni con la memoria (lettura e scrittura dati), con i device e manipolazione dei dati

CENTRAL PROCESSING UNIT (CPU)

- Componente che esegue le istruzioni del programma codificate **in linguaggio macchina**
- Caratterizzato da un proprio **Instruction Set Architecture** (**ISA**)
  - insieme di possibili istruzioni riconosciute dalla CPU
  - insieme di registri
    - registri generali o programmabili **(GPR)**
      - es: accumulatore
    - registri di stato e controllo
      - Program counter (PC) - l'indirizzo della prossima istruzione da eseguire.
      - Program Status Word (PS) - informazioni sullo stato del processore
- CPU a XX bit (XX = 8, 16, 32, 64)
  - numero bit utilizzati nei registri e specificare indirizzi di memoria e valori
- Esempio: architettura Intel x86 (IA32 , x86-64)
  - istruzioni: mov, add, mul, inc, cmp, jmp, jz, push, pop,....
  - registri:
    - ax, bc, cx, dx, si, di, ... (16bit),
    - eax, ebc, ecx, edx, esi, ... (32bit),
    - rax, rbx, rcx,... (64bit)

ESEMPI DI CPU: INTEL

- Famiglia Intel
  - http://en.wikipedia.org/wiki/List_of_Intel_microprocessors
- Esempi delle versioni più recenti
  - desktop: Core i7 Sandy Bridge-E (Novembre 2011)
    - tecnologia 32nm
    - 2.5 / 3.9 GHz
    - 8 physical cores/16 threads
    - 32+32 Kb (per core) L1 cache, 256 Kb (per core) L2 cache, 20 MB L3 cache
    - 2270 milioni di transistor
    - consumo: 130W
  - mobile: Core i7 Ivy Bridge (Aprile 2012)
    - tecnologia 22nm Tri-gate
    - 2.5 / 3.9 GHz
    - 4 physical cores/8 threads
    - 32+32 Kb (per core) L1 cache, 256 Kb (per core) L2 cache, 8 MB L3 cache
    - consumo: 45 W / 77 W
    - Integrated GPU Intel HD Graphics 4000

LO STACK

- A supporto dell'esecuzione della CPU c'è sempre uno *stack*
  - struttura dati di tipo LIFO, in memoria centrale
    - operazioni di push e pop
  - registro **Stack Pointer** (SP)
    - punta alla cima dello stack
- Utilizzato in caso di chiamate di subroutine (istruzioni del tipo call) o in caso di interruzioni software (tipicamente per realizzare system call)
  - memorizzazione e ripristino del valore del PC
  - memorizzazione parametri passati alla subroutine / system call
  - usato dai linguaggi di programmazione per memorizzare le variabili locali a blocchi computazionali (funzioni, procedure, metodi,...)

LA MEMORIA

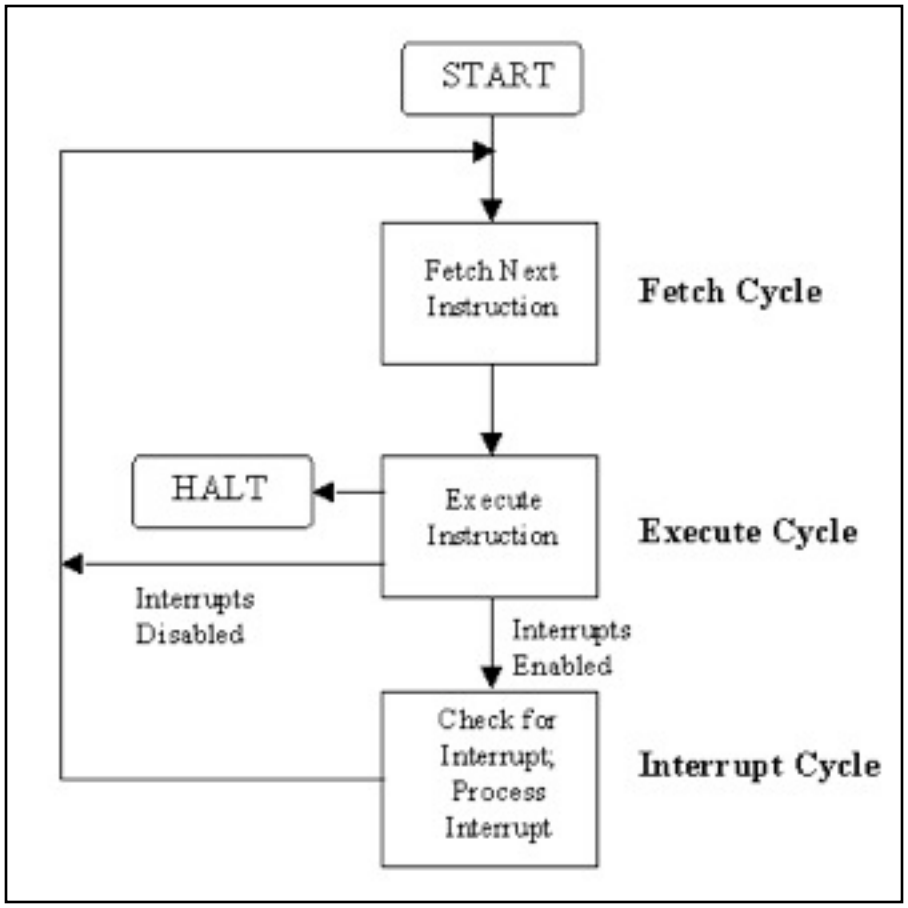
- La CPU e i device controller sono entità autonome, che competono per accedere alla *memoria condivisa*
  - la coordinazione degli accessi è fatta dal *memory controller*
  - variazioni di questo schema base includono CPU multiple, bus dedicati per interazione CPU e memoria, ..
- La memoria in generale è la risorsa con lo scopo di contenere dati, che è possibile manipolare con operazioni di lettura e scrittura
- Fra i parametri caratteristici:
  - persistenza
    - se volatile o persistente
  - capacità
    - quanti dati è in grado di contenere
  - velocità
    - definisce prestazioni in accesso e trasferimento di dati o throughput
  - costo

MEMORIA PRINCIPALE (O CENTRALE)

- I programmi devono risiedere nella memoria principale (chiamata **RAM, random access memory**) per essere eseguiti
  - fra le tecnologie più usate abbiamo la **DRAM** (**dynamic random access memory**)
- La memoria è costituita da insieme contiguo (**array**) di celle (dell'ordine dei miliardi), dette anche parole **(word)** di memoria, accessibili direttamente dalla CPU per essere lette o scritte
  - ogni parola di memoria ha un indirizzo
- La CPU interagisce con la memoria mediante delle istruzioni di **load** con cui carica nei registri una parola di un dato indirizzo, e **store** con cui si scrive il contenuto di un registro in una parola in memoria
- Oltre al load e store, la CPU carica automaticamente l'istruzione corrente da eseguire, il cui indirizzo è contenuto nel program counter

MACCHINA DI VON NEUMANN: CICLO DI ESECUZIONE

- L'**instruction-execution cycle** di un computer in quanto macchina di Von Neumann consiste nel:
  - caricamento dalla memoria (**fetch**) dell'istruzione da eseguire, depositata in un apposito registro (**instruction register**)
  - **decodifica** dell'istruzione, che può comportare il caricamento di altri operandi dalla memoria
  - **esecuzione** dell'istruzione, che può portare all'aggiornamento dei registri e alla scrittura di dati in memoria

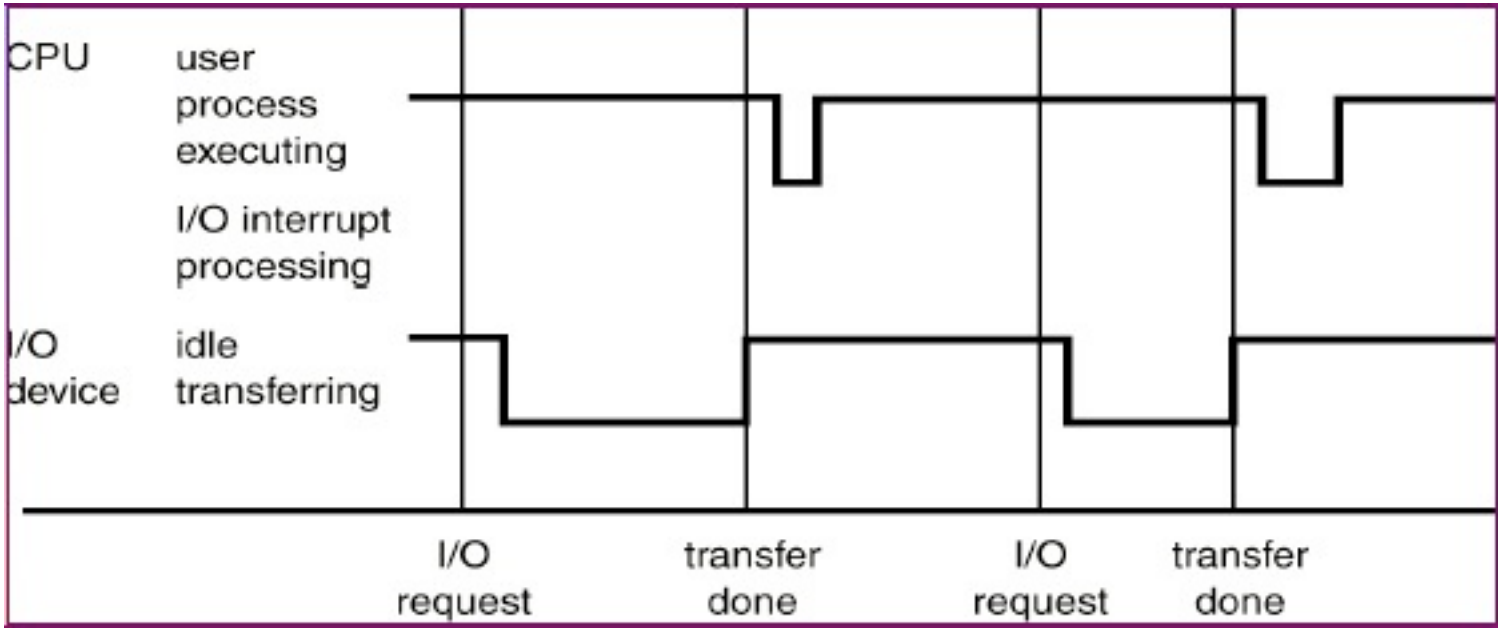


MECCANISMO DELLE INTERRUZIONI (INTERRUPT)

- Un **interrupt** in generale rappresenta la segnalazione di un certo evento a cui la CPU e il Sistema Operativo devono far fronte, sospendendo l'attività corrente
  - meccanismo *fondamentale* per i sistemi operativi
- Gli interrupt possono essere hardware o software
  - gli **interrupt hardware** sono causato da device e avvengono fisicamente come segnali inviati alla CPU tramite il bus di sistema
  - gli **interrupt software** sono generati da programmi in esecuzione
    - mediante una opportuna istruzione in linguaggio macchina
      - `int` (architetture Intel), `svc` (architetture hw IBM),...
- Uso degli interrupt software
  - per richiedere l'esecuzione di servizi propri del sistema operativo (ovvero di *system call*)
  - per manifestare situazioni di errore (*exceptions*)

INTERRUPT SERVICE ROUTINE (HANDLER)

- Per ogni tipo di interruzione, il S.O. associa una determinata porzione di programma da mandare in esecuzione come 'reazione' all'evento (**interrupt service routine** o **interrupt handler**)
- La dinamica dell'occorrenza ed esecuzione di una interruzione è esemplificato in figura (caso di output con un singolo processo)



DINAMICA DELL'ESECUZIONE DI UNA INTERRUZIONE

- All'occorrenza di una interruzione, la CPU *interrompe* ciò che stava facendo ed *traferisce il controllo* immediatamente ad una prefissata locazione di memoria
  - salvataggio sullo stack (push) del valore del PC per la prossima istruzione da eseguire, che dipende dal tipo di interruzione.
  - In tale locazione si trova poi l'indirizzo di partenza della service routine associata all'interruzione
- L'interrupt service routine viene eseguita
- Al completamento, la CPU ripristina ciò che stava facendo
  - continua l'esecuzione a partire dall'indirizzo recuperato (pop) dallo stack
- Possibilità di abilitare / disabilitare interruzioni
  - bit nel registro di stato del processore
  - STI = Set Interrupt (abilita)
  - CLI = Clear Interrupt (disabilita)

TABELLA DELLE INTERRUZIONI

- Le interruzioni sono tipicamente identificate con un numero progressivo, da zero in su
- Gli indirizzi delle routine di interruzione sono memorizzati in una tabella o array, chiamata **interrupt vector,** indicizzata per numero di interruzione, tipicamente collocata nella parte più bassa della memoria disponibile
  - contiene indirizzo dei vari interrupt handler e del valore delle PS da settare quando viene servita l'interruzione

MECCANISMI HARDWARE PER LA PROTEZIONE

- I sistemi moderni consentono l'esecuzione di più programmi, anche di utenti diversi, che mediante i servizi forniti dal sistema operativo condividono risorse quali CPU, memoria e risorse di I/O.
- E' fondamentale allora che ci siano meccanismi e politiche di protezione
  - l'eventuale non funzionamento non corretto di un programma *non causi il malfunzionamento di altri programmi o del sistema operativo stesso*
- Tale protezione viene realizzata tipicamente mediante politiche del SO, basate su meccanismi forniti dall'hardware

MODALITA' OPERATIVE CPU: USER MODE E MONITOR MODE

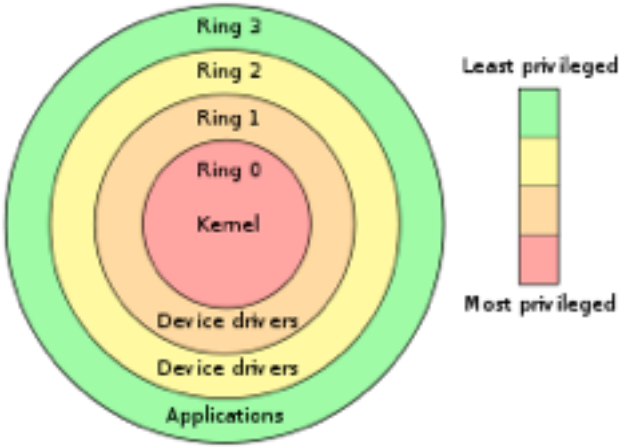
- La prima forma di protezione consiste nel proteggere il S.O. da eventuali azioni dannose o erronee eseguite da parte dei programmi utente
- Per abilitare questa forma di protezione, le CPU moderne forniscono un supporto hardware diretto, per cui il funzionamento di una CPU viene suddivisa in due modalità fondamentali:
  - **user mode**
    - la CPU può eseguire solo un certo sottoinsieme delle istruzioni possibili
  - **monitor mode**
    - chiamata anche *supervisor mode*, *system mode*, *privileged mode*
    - in monitor mode può eseguire qualsiasi istruzione
- Questa distinzione permette di discernere quando il sistema stia eseguendo codice del S.O. (monitor mode) e codice dell'utente (user mode)
  - un bit (mode bit) indica se il sistema sta eseguendo in modalità user (1) o monitor (0)

USER MODE E MONITOR MODE (2)

- Al boot l'hardware parte in monitor mode
  - nei processori Intel il boot è, in realtà, in user mode
- Il sistema operativo viene caricato e quindi i processi utente sono mandati in esecuzione in modalità user
- Non appena viene scatenata una interruzione o trap, l'hardware esegue uno switch in kernel mode
- Poi prima di restituire il controllo allo user, viene nuovamente ripristinata la modalità user

CASO CONCRETO: INTEL

- Due modalità di funzionamento del processore
  - modalità **reale** (processori 8086, 8088)
    - nessun supporto di protezione
  - modalità **protetta** (processori 80386, 80486, Pentium e successori)
    - con supporto di protezione (user mode, monitor mode)
    - supporto per paginazione e multitasking
- In modalità protetta
  - quattro livelli di privilegio (*privilege levels* o **ring**)
    - ring 0 (più privilegiato) a ring 3 (meno privilegiato)
  - il Sistema Operativo insieme ad alcuni device driver è tipicamente in esecuzione al ring 0 e le applicazioni al ring 3



x86-64: LONG MODE

Operating mode		Operating system required	Compiled-application rebuild required	Default address size	Default operand size	Register extensions	Typical GPR width
Long mode	64-bit mode	OS with 64-bit support, or bootloader for 64-bit OS	Yes	64	32	Yes	64
	Compatibility mode		No	32	32	No	32
				16	16		16
Legacy mode	Protected mode	Legacy 16-bit or 32-bit OS; or bootloader for 16, 32, or 64-bit OS	No	32	32	No	32
				16	16		16
	Virtual 8086 mode	Legacy 16-bit or 32-bit OS		16	16		16
	Real mode	Legacy 16-bit OS; or bootloader for 16, 32, or 64 bit OS					

ISTRUZIONI PRIVILEGIATE

- Istruzioni che possono essere eseguite solo in monitor mode in quanto possono influire sul comportamento dell'intero sistema
  - eseguibili quindi tipicamente solo dal sistema operativo
  - il tentativo di esecuzione di una istruzione privilegiata in user mode comporta un errore di protezione generale (GP)
    - gestito dal sistema operativo che termina il processo che ha causato l'errore
- Esempi di istruzioni privilegiate in IA32

```
CLTS - Clear Task-Switched Flag    LMSW    - Load Machine Status
HLT  - Halt Processor              LTR     - Load Task Register
LGDT - Load GDT Register           MOV CRn - Move Control Register
LIDT - Load IDT Register           MOV DRn - Move Debug Register
LLDT - Load LDT Register           MOV TRn - Move Test Register
```

SENSITIVE INSTRUCTIONS

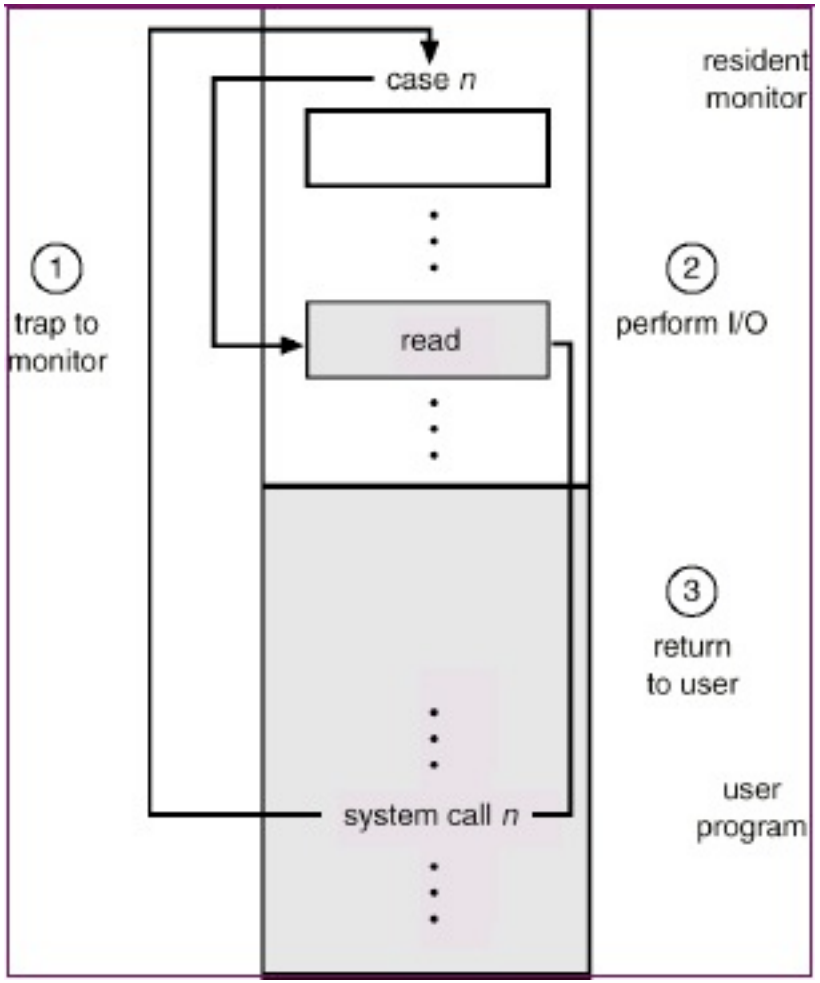
- Nelle architetture Intel si distinguono instruzioni privilegiate e *sensitive instructions* (dette anche IOPL-sensitive)
  - sono istruzioni che possono essere messe in esecuzione solo se il livello di privilegio corrente è superiore o uguale al cosiddetto *livello di privilegio di I/O* che può essere assegnato ad ogni processo (dal SO)
  - nel caso di violazione, viene generato un errore di protezione generale gestito dal sistema operativo
- Esempi di sensitive instructions

```
IN  - Input           OUTS - Output String
INS - Input String    CLI  - Clear
Interrupt-Enable Flag (IF)
OUT - Output          STI  - Set IF
```

PROTEZIONE I/O

- Le istruzioni di I/O sono trattate come istruzioni privilegiate, e possono essere eseguite solo dal sistema operativo via system call

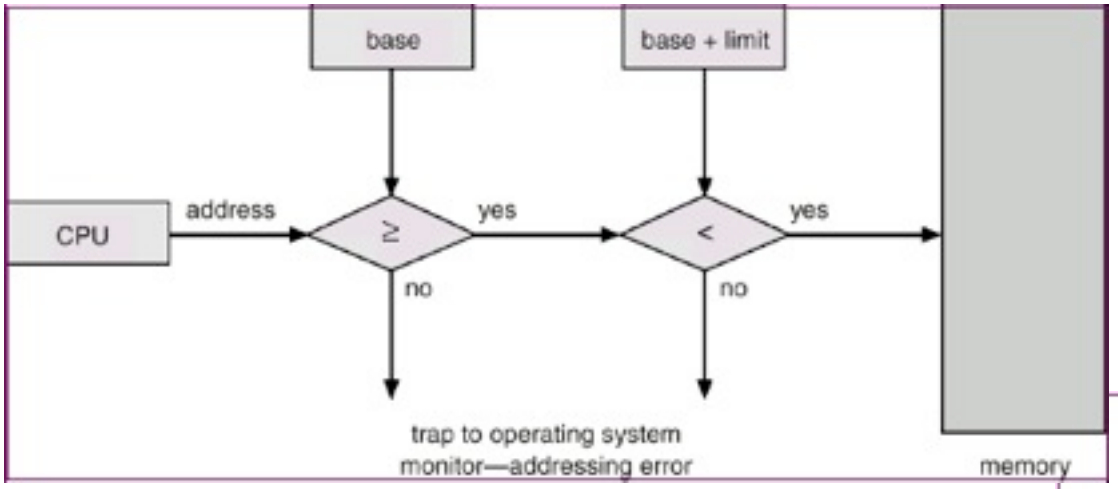
- E' fondamentale che non capiti mai che un programma utente ottenga il controllo della CPU mentre è in monitor mode, che gli consentirebbe di avere accesso completo a qualsiasi risorsa del sistema.

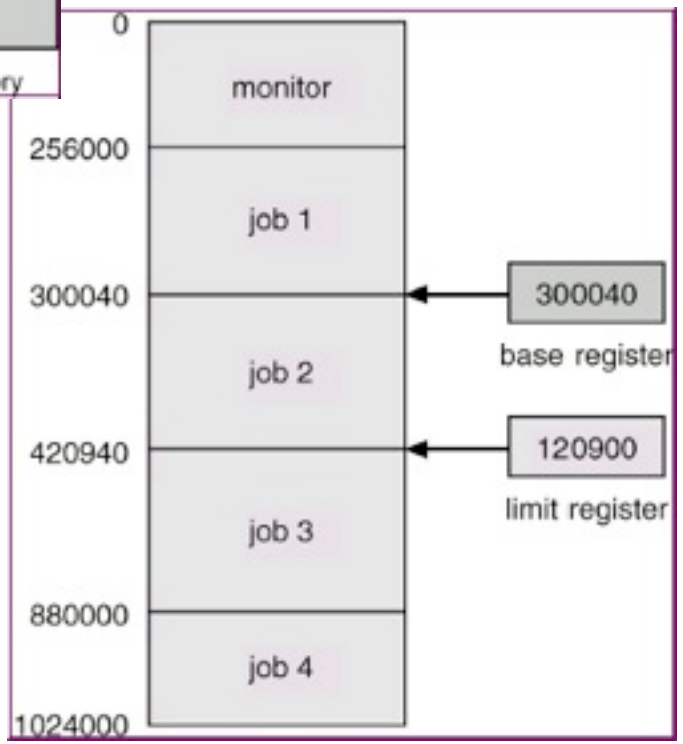


PROTEZIONE DELLA MEMORIA

- Una completa protezione include il fatto di evitare che processi (programmi in esecuzione) distinti 'sporchino' volontariamente e non dati / codice di altri processo, o del sistema operativo stesso
- Per fare questo l'hardware mette a disposizione meccanismi utilizzati dal S.O. per separare completamente lo spazio di memoria fra S.O. e processi, ed impedire accessi erronei alla memoria
  - (MMU) Memory Management Unit, parte della CPU
- Un meccanismo elementare di protezione è dato da due registri, **base register** e **limit register** che contengono indirizzo di base e dimensione della memoria accessibile da un determinato processo in esecuzione.
  - In modo user, ogni indirizzo di memoria è confrontato con quello dei registri: se esce viene generata una eccezione

ESEMPIO DI RILOCAZIONE E CONFINAMENTO





TIMER

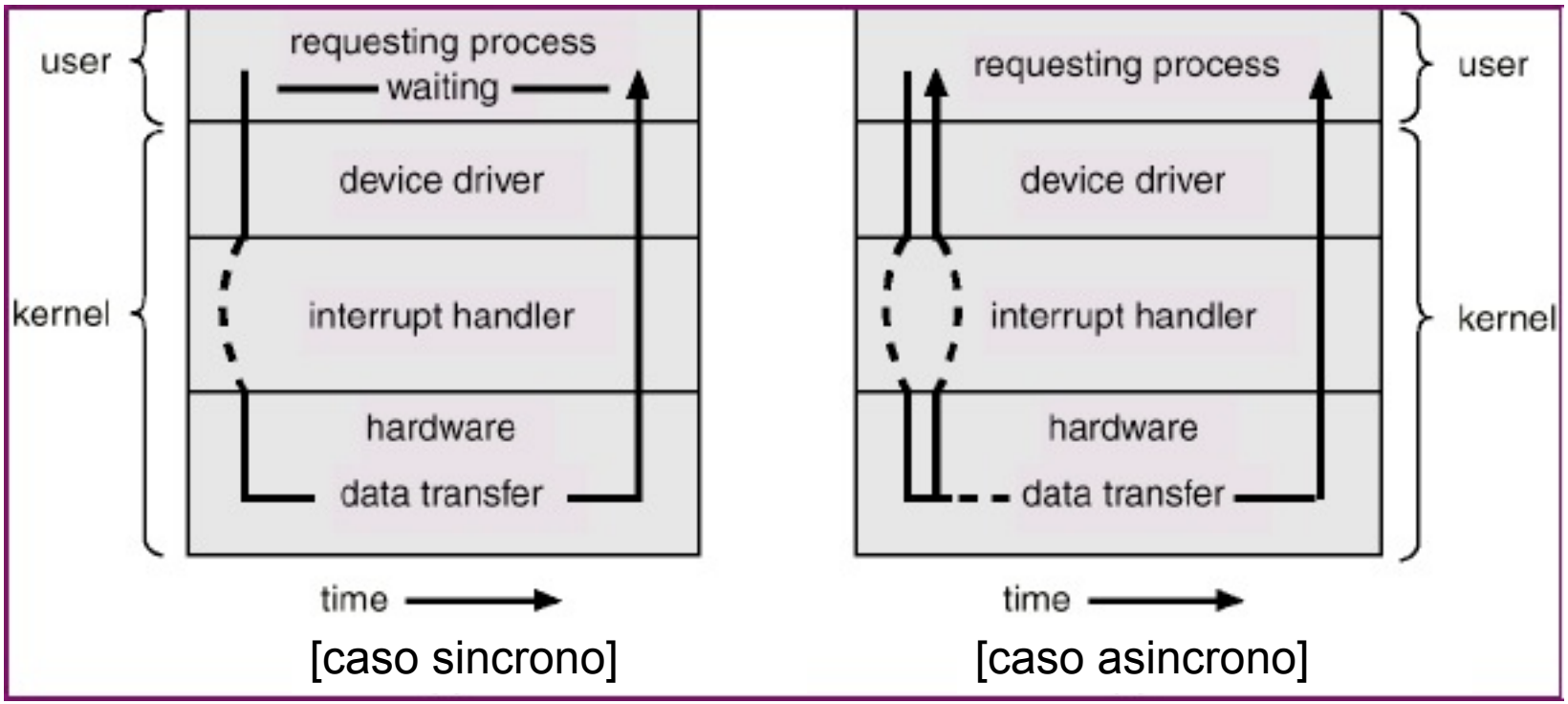
- L'altra risorsa fondamentale che va protetta rispetto ad un uso non consentito da parte dei programmi utente è la CPU, il cui controllo deve tornare costantemente al SO, a prescindere dalle istruzioni in esecuzione da parte dei processi
  - questo è fondamentale per evitare che processi malevoli o errati (es: loop infiniti) causino il monopolio della CPU
- Il meccanismo per realizzare tale protezione è dato dal **timer,** dispositivo con cui vengono generate interruzioni periodiche ogni un certo quanto di tempo, che può essere fisso (es: 1/60 di secondo) o variabile (da 1 ms a 1 s)
  - Con tale meccanismo si ha la sicurezza che ogni quanto di tempo il controllo della CPU viene dato al SO, alla routine di gestione dell'interruzione
- Questo è il meccanismo fondamentale con cui si realizza il *multitasking* e sistemi simulazione real time, come video giochi

GESTIONE I/O

- Ogni **device controller** controlla uno specifico tipo di device (disk drive, input device, I/O bus, video display..)
  - un medesimo controller può controllare più device (vedi ad esempio **SCSI**, small computer systems interface)
  - un device controller ha tipicamente un buffer di memoria locale privato e un insieme di registri specifici (*special-purpose*)
  - Il controller è responsabile per il trasferimento di dati dal device al buffer di memoria locale
- La porzione del sistema operativo che pilota uno specifico dispositivo (e il suo controller) viene chiamata **device driver**
- L'esecuzione di una generica operazione di I/O avviene in generale nei seguenti passi:
  - la CPU carica nei registri del controller determinati valori
  - il controller quindi esamina tali valori e di conseguenza esegue determinate azioni sul device, come ad esempio il trasferimento di dati
  - al completamento dell'operazione, il controller informa la CPU del completamento (es: dati trasferiti) mediante il triggering di una interruzione

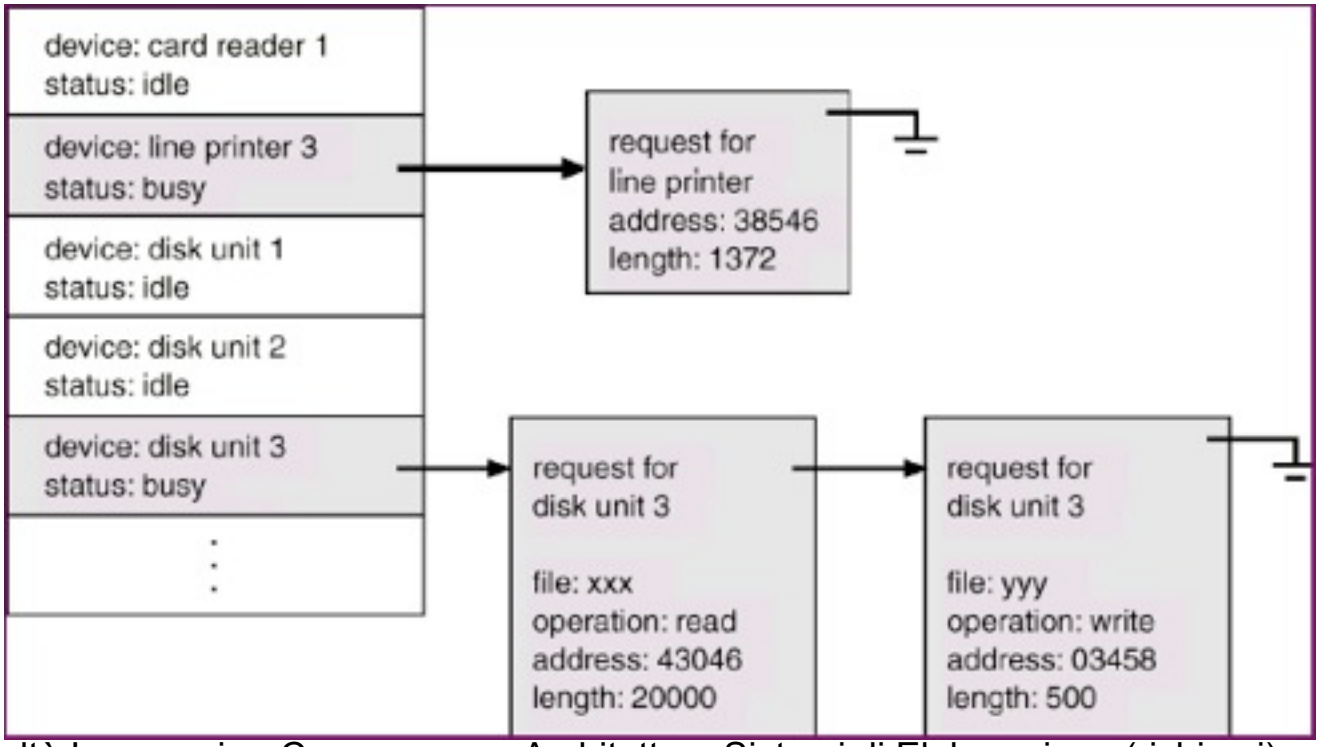
I/O SINCRONO e ASINCRONO

- La modalità di tale esecuzione può essere **sincrona** o **asincrona**:
  - nel primo caso la CPU viene bloccata fino a quando l'operazione di I/O non sia terminata
  - nel secondo caso invece prosegue, continuando le istruzione dello user program



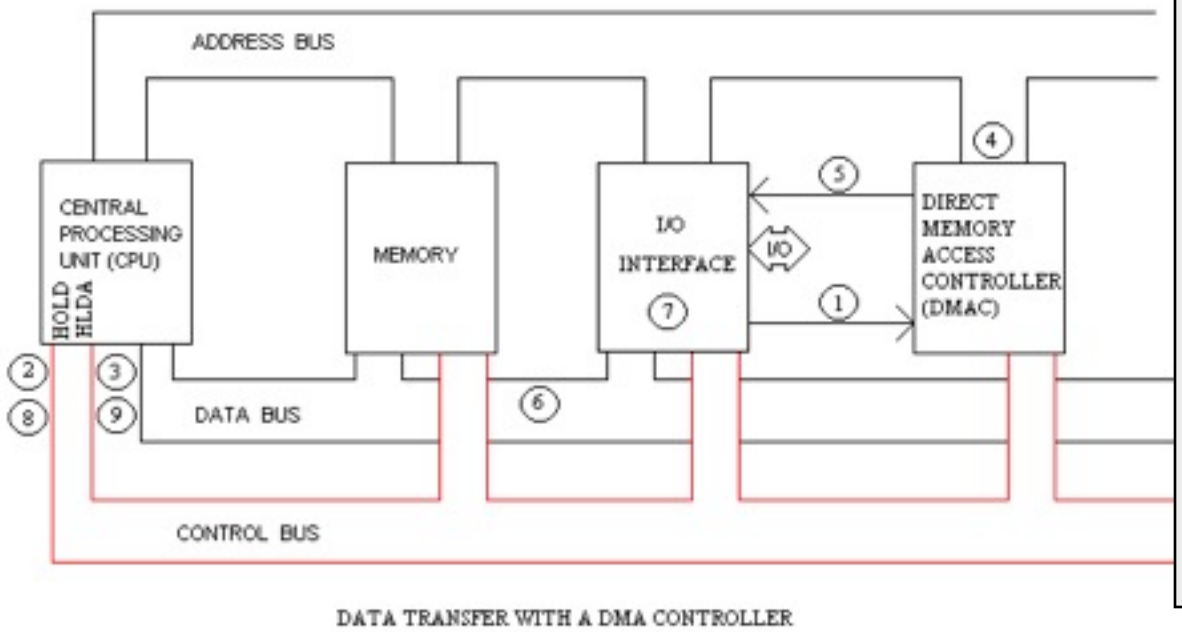
I/O SINCRONO e ASINCRONO (2)

- Nel caso sincrono la CPU tipicamente rimane in attesa mediante una istruzione di **wait**
- Il caso asincrono ha il vantaggio notevole di permette l'esecuzione simultanea di più operazioni di I/O e CPU
  - in tal caso una tabella (**device-status table**) viene mantenuta dal S.O. che tiene traccia dello stato di I/O di ogni device
  - per ogni device il S.O. mantiene tipicamente anche una coda di attesa (**wait queue**), in cui c'è la lista dei processi in attesa di risposta sullo specifico device



DMA

- Il trasferimento fra dati da memoria locale al device controller alla memoria principale, è tipicamente attuata in modalità **DMA** (**direct memory access**), ovvero senza intervento della CPU - che può continuare la propria attività - e direttamente ad opera del controller, pilotato dal device driver



DATA TRANSFER WITH A DMA CONTROLLER

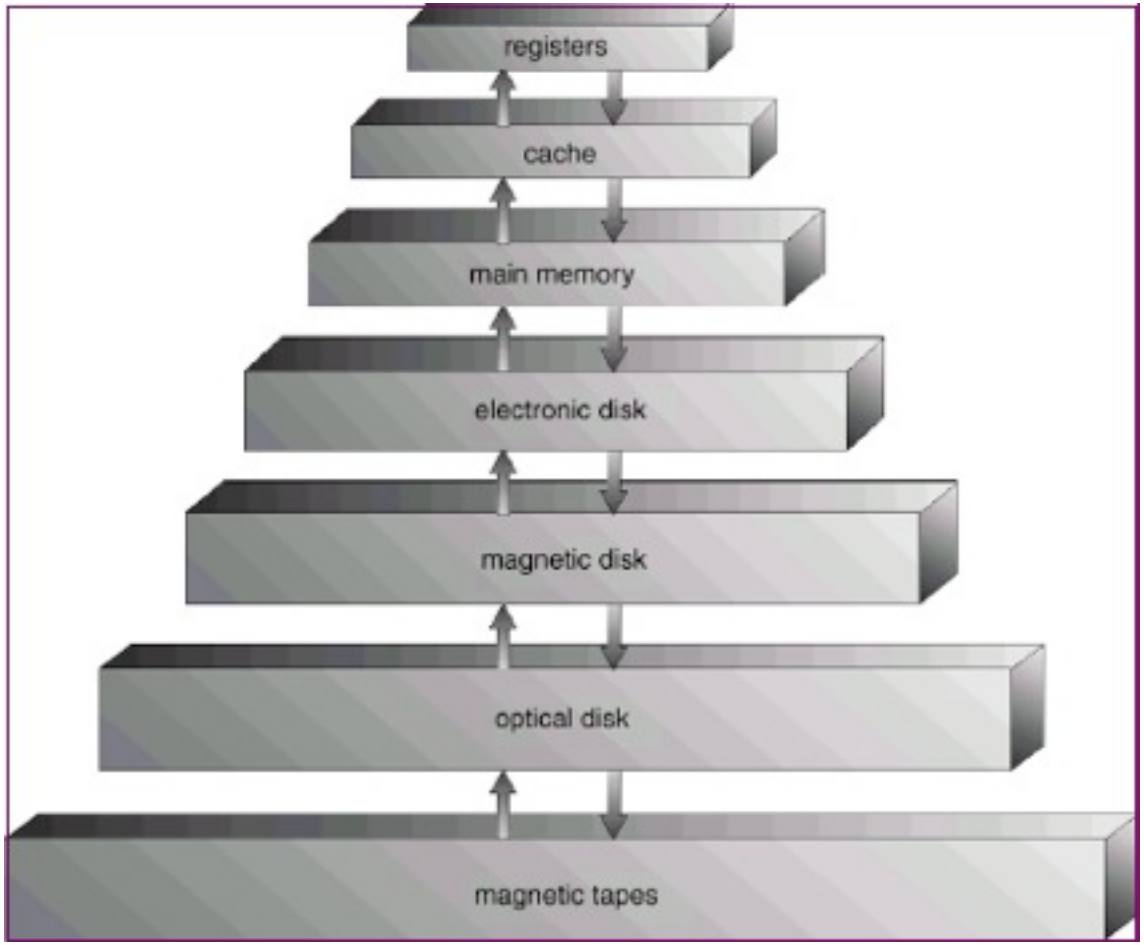
1)	Request sent to DMA
2)	Bus request toto get bus control
3)	Bus grant fromthe CPI
4)	DMAC put address register on the bus
5)	DMAC send ACK to I/O Interface
6)	Data transferred to memory
7)	Interface latches data
8)	Bus request is dropped, DMAC releases bus control
9)	Bus grant from the CPU
10)	address register incremented
11)	byte count decremented
12)	if count != 0, goto step 1) otherwise stop

MEMORY-MAPPED I/O

- Per facilitare l'interazione fra CPU è device di I/O, nella maggior parte delle architetture moderne i registri del device controller (**I/O port**) vengono mappati in memoria centrale (**memory mapped I/O**)
- la lettura e scrittura di valori ad indirizzi relativi a tale porzione (da parte della CPU) comportano la scrittura / lettura direttamente dei registri del device

GERARCHIA DEI TIPI DI MEMORIA

- Esiste una completa **gerarchia** di tipi di memoria, che rispecchia capacità / costo / velocità:

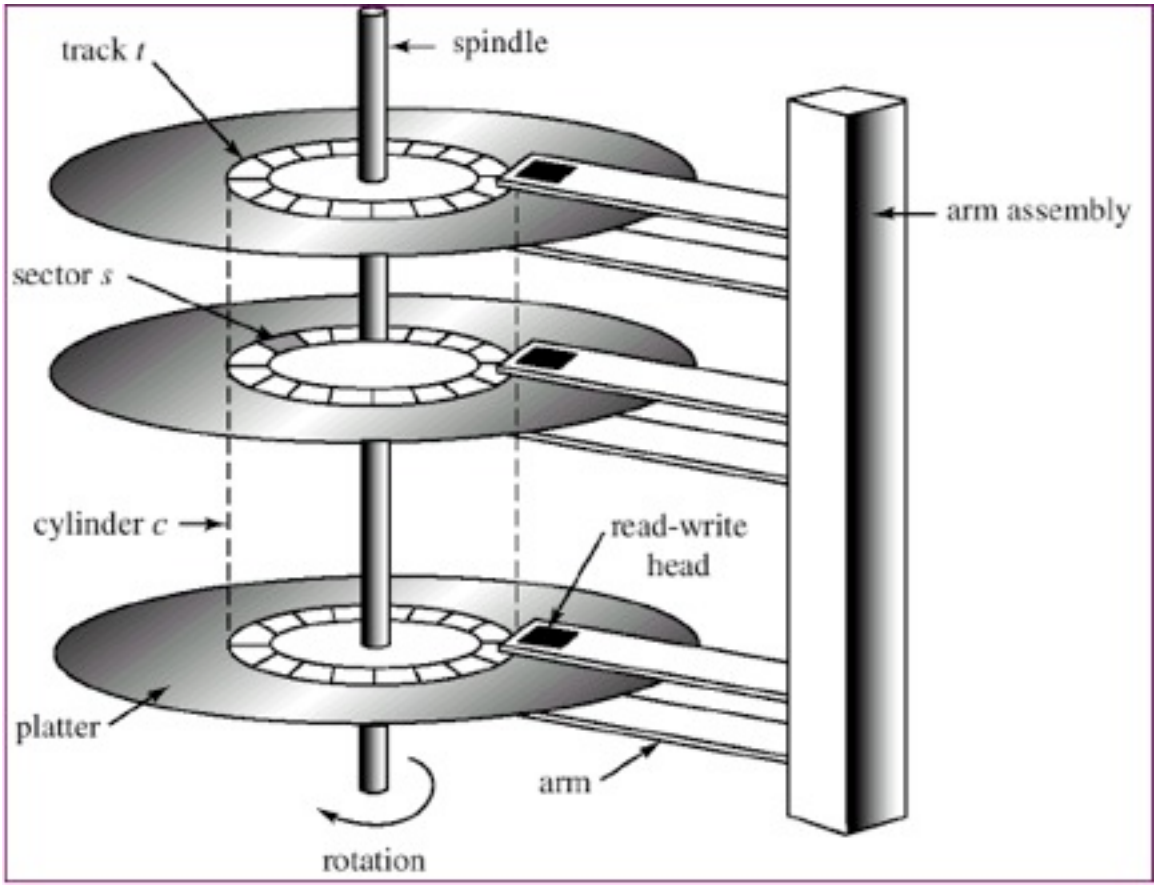


MEMORIA SECONDARIA

- La memoria principale è *volatile*, ovvero il suo contenuto non persiste allo spegnimento del computer.
- Per la memorizzazione persistente delle informazioni si utilizza la **memoria secondaria**, il cui obiettivo è mantenere in modo persistente grandi quantità di dati, superiore spesso a due ordini di grandezza rispetto la memoria principale.
- I **dischi magnetici** sono la forma più diffusa di memoria secondaria.

DISCHI MAGNETICI

- La struttura semplificata di **disk drive** magnetico - hard disk in particolare - è descritta in figura:



DISCHI MAGNETICI

- Tipicamente hard disk contengono migliaia di cilindri, e ogni traccia può contenere centinaia di settori.
  - La velocità di rotazione dell'ordine delle migliaia di giri per minuto
    - es: 7200 RPM (revolutions per minute)
- La velocità è misurata in:
  - transfer rate
    - quantità di dati trasferiti per unità di tempo - dell'ordine di svariate decine di megabyte al secondo.
  - random-access time o positioning time
    - somma del tempo impiegato per posizionare il braccio sul cilindro (**seek time**) - e del tempo impiegato per il posizionamento su settore voluto (**rotational latency**) - dell'ordine di millisecondi

DISK CONTROLLER

- Un disk drive è controllato da un **controller**, suddiviso in **host controller** (host bus adapter, HBA) che risiede sul bus di sistema e **disk controller**, componente invece collocata sul drive, dotata tipicamente di memoria cache
- Il controller interagisce con il drive mediante un **I/O bus**. Esistono vari tipi di bus
  - **EIDE** (enhanced integrated drive electronics), **ATA** (advanced technology attachment), **FC** (fibre channel), **SCSI** (small computer system interface)
- Per eseguire una operazione su disco, la CPU comunica un comando al host controller (scrivendo sui registri del controller, mediante mapped memory), l'host controller invia messaggi al disk controller, che pilota direttamente il drive.

MEMORIA CACHE

- I dati sono mantenuti in generale in memoria centrale. Tuttavia i dati che vengono usati con maggior frequenza vengono copiati in una memoria molto più veloce di quella centrale, **memoria cache**, tipicamente collocata direttamente nella CPU.
  - l'accesso e modifica a tali dati implica quindi l'accesso solo alla memoria cache, senza richiedere l'accesso alla memoria centrale
- Il caching è una tecnica fondamentale che compare a vari livelli nei sistemi informatici, come pattern di progettazione che permette di aumentare in modo significativo le performace dei sistemi

CACHING

- Più in generale, si parla di caching ogni qual volta si utilizzi un *buffer* di memoria (speciale o non) per contenere la copia di sottoinsieme di dati di un insieme molto più grande, per il quale il tempo di accesso a tali dati è molto maggiore.
  - l'insieme può essere una memoria più grande, come nel caso di memoria cache e memoria principale, oppure un insieme di dati risultati di un calcolo, di una computazione.
- Si basa su *principio di località temporale e spaziale*
  - ovvero sul fatto che nell'interazione fra componenti si possano identificare delle regolarità per cui, da un lato, sui medesimi dati frequentemente più operazioni vengono eseguite in un ristretto arco temporale, dall'altro, sia no frequenti gli accessi a dati 'vicini' secondo una certa metrica.

FORME DI CACHING

- Le forme di caching più diffuse sono:
  - memoria RAM e memoria cache / internal programmable register del processore
  - caching del codice
  - caching nei I/O controller
- Ma anche a livello più alto, di applicazioni:
  - cache di file in memoria centrale
  - cache di una pagina web

GESTIONE DELLA CACHE

- Affinché un sistema di caching funzioni correttamente ed efficacemente, è necessario progettare in modo opportuna dimensione della cache e le strategia con cui si gestiscono (aggiornano, leggono) i dati contenuti (*cache management*)
- Il trasferimento di informazioni dalla memoria principale alla cache può essere fatta esplicitamente o implicitamente, a seconda dell'hardware di supporto e del controllo del sistema operativo
- In generale le strategie devono garantire coherency (coerenza) o **consistency** (consistenza) delle informazioni replicate in cache / memoria centrale.
- Tale problema si presenta in generale su ogni qualvolta ci siano informazioni replicate, che possono essere accedute e modificate da più parti (es: più processori), quindi si può presentare per livelli diversi dell'intera gerarchia di memorie viste in precedenza.
  - Esempi:
    - contenuto di un file su disco, caricato in RAM
    - pagina web visualizzata da un web browser

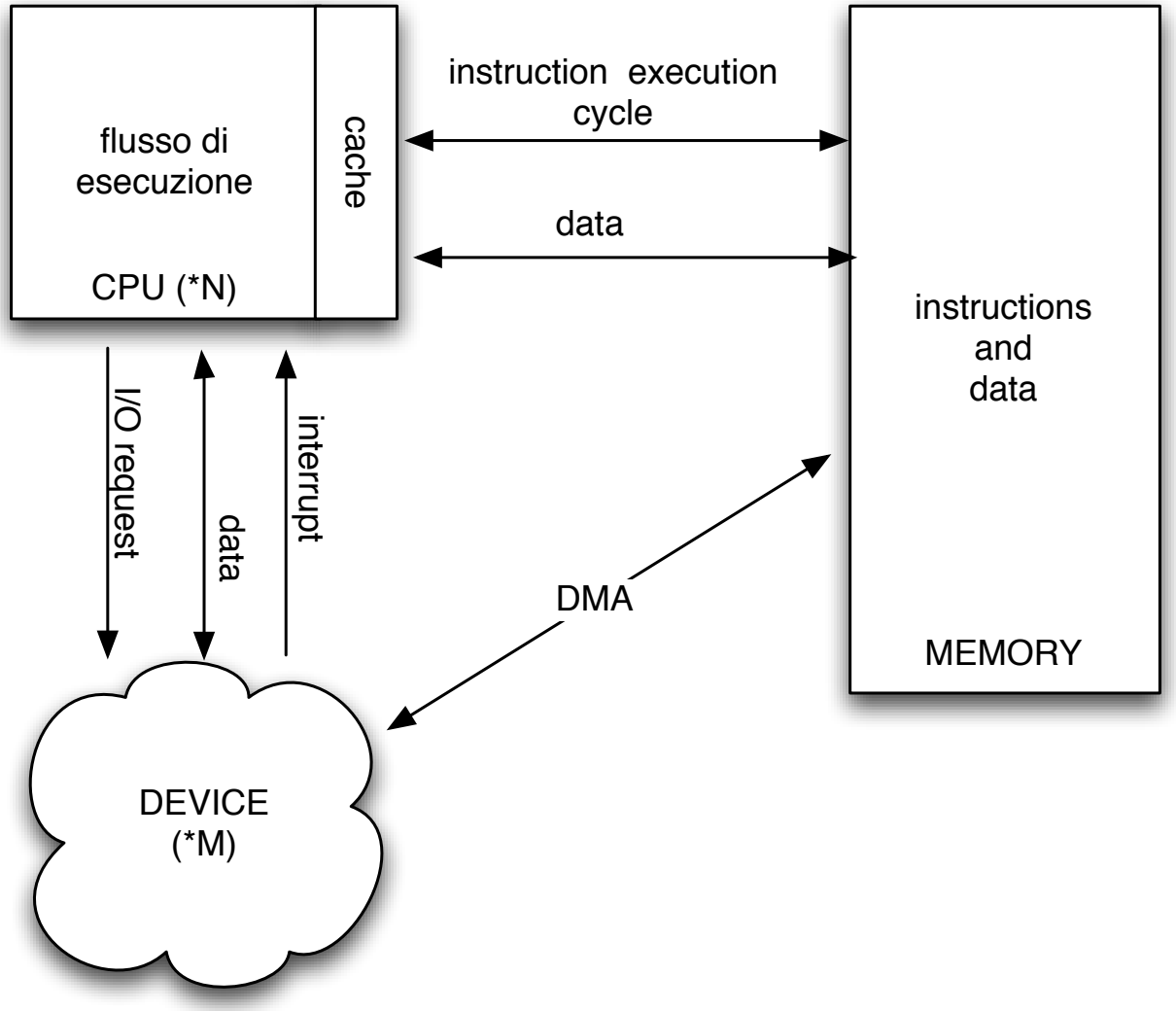
CACHE COHERENCY

- Per **cache coherency** si intende l'allineamento del valore replicato dell'informazione: tutte le copie repliche della medesima informazione devono essere aggiornate non appena una venga cambiata.
  - ad esempio: se un I/O device cambia il valore di dati presenti in cache nella CPU, mediante trasferimenti DMA.
  - o ancora: se N processori hanno in cache lo stesso dato di memoria, la modifica di tale valore da parte di un processore sia coerentemente applicata a tutte le repliche delle altre cache.
- La cache coherency è tipicamente gestita a livello hardware.

PERFORMANCE DEI LIVELLI DI MEMORIZZAZIONE

Livello	1	2	3	4
Nome	registri	cache	main memory	disk storage
Dimensione tipica	< 1KB	< 16MB	< 64 GB	> 100 GB
Tecnologia implementativa	custom memory con porte multiple, CMOS	on-chip o off-chip CMOS SRAM	CMOS DRAM	dischi magnetici
Tempo di accesso (ns)	0.25-0.5	0.5-25	80-250	5000000
Bandwidth (MB/sec)	20000-100000	5000-10000	1000-5000	20-150
Managed by	compilatore	hardware	sistema operativo	sistema operativo
Backed by	cache	main memory	disk	CD o nastri

FUNZIONAMENTO COMPLESSIVO

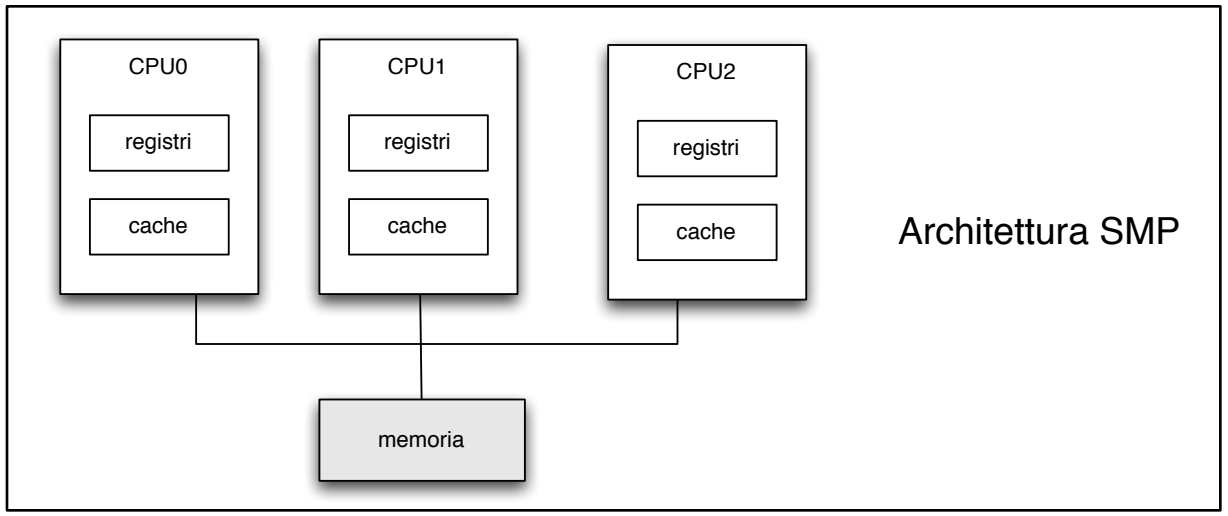


ARCHITETTURE MULTI-PROCESSORE

- La maggior parte dei sistemi di elaborazione è a singolo processore
  - una CPU in grado di eseguire un insieme di istruzioni general-purpose
  - processori special-purpose per compiti dedicati
    - processori di dispositivi specifici (es grafici), gestione I/O,..
    - set di istruzioni limitate
- Recente sviluppo e diffusione di architetture **multi-processore**
  - due o più processori con condivisione di risorse che dipende dall'architettura specifica
    - memoria, bus, clock
  - vantaggi
    - **incremento del throughput**
      - parallelismo nell'esecuzione delle attività computazionali
    - **economia di scala**
      - costo sistemi multiprocessore < costo sistemi multipli a processor singolo
    - **incremento dell'affidabilità**
      - graceful degradation e sistemi fault tolerant

TIPI DI ARCHITETTURE MULTI-PROCESSORE

- **Asymmetric Multi-Processing**
  - ad ogni processore è assegnato un task specifico
  - un processore master controlla il lavoro degli altri processori
  - approccio utilizzato in sistemi special purpose
- **Symmetric Multi-Processing (SMP)**
  - ogni processore può eseguire qualsiasi task/processo del sistema operativo
  - una copia del sistema operativo è in esecuzione su ogni processore
  - tutti i sistemi operativi moderni supportano architetture SMP



ARCHITETTURE MULTI-PROCESSORE SU SINGOLO CHIP

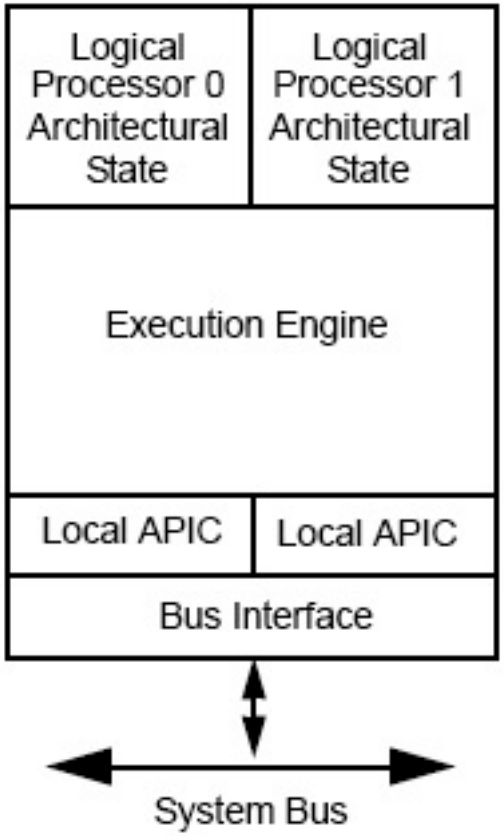
- Sviluppo recente di tecnologie che permettono di includere più unità di processamento (logiche o fisiche) nel medesimo chip
  - vantaggi rispetto alle architetture multi-chip
    - riduzione del consumo di potenza
    - aumento efficienza (comunicazione on-chip)
- Tecnologie di riferimento
  - **hyper-threading**
  - architetture **multi-core**

TECNOLOGIA HYPER-THREADING

- Introdotta da Intel per aumentare le performance nell'esecuzione di programmi *multi-threaded* e single-threaded in ambiente multi-tasking
- Singolo processore in grado di eseguire due o più flussi di esecuzione concorrentemente sfruttando risorse di esecuzione condivise

➡ Risultato: **due o più *processori logici*** ognuno con il proprio stato

- registri
- advanced programmable interrupt controller (APIC)

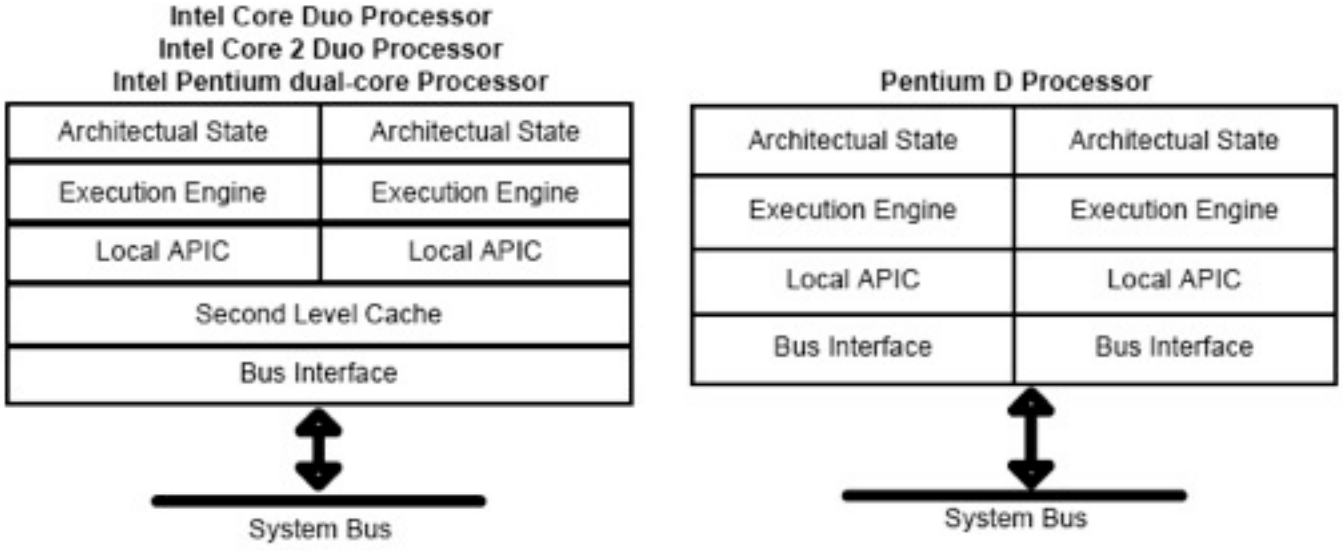


ARCHITETTURE MULTI-CORE

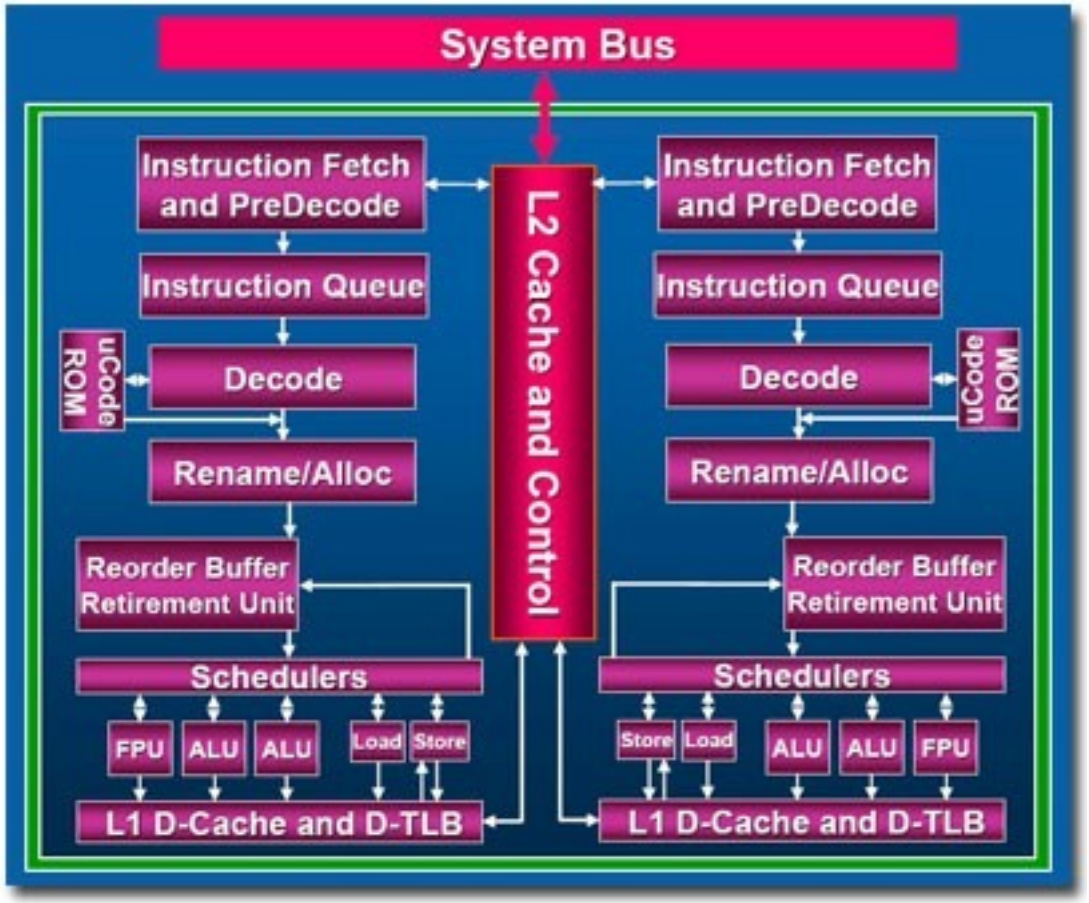
- Disponibilità di due o più nuclei di esecuzione (**core)** in un singolo package
  - ogni core ha le proprie risorse micro-architetturali identiche alle implementazioni con singolo processore, incluse il motore di esecuzione
  - topologia della gerarchia della memoria cache (condivisione o meno dei vari livelli di cache) dipende dalle specifiche implementazioni

➡ Risultato: due o più processori *logici* ognuno con il proprio stato e gestione del flusso di esecuzione

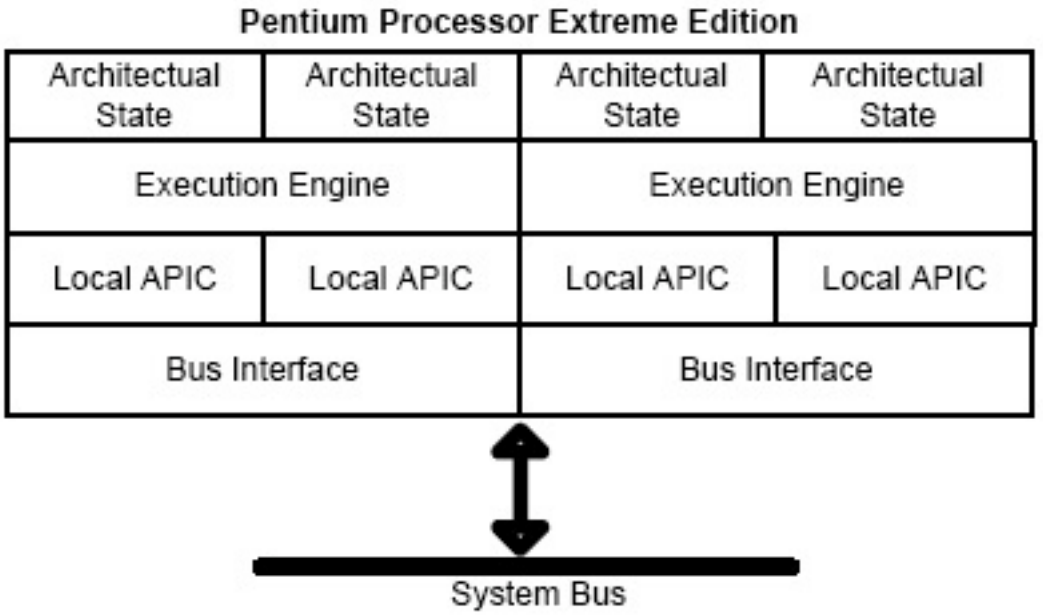
- esempi: AMD Phenom, AMD Opteron, Intel Core Duo, Intel Core Duo 2



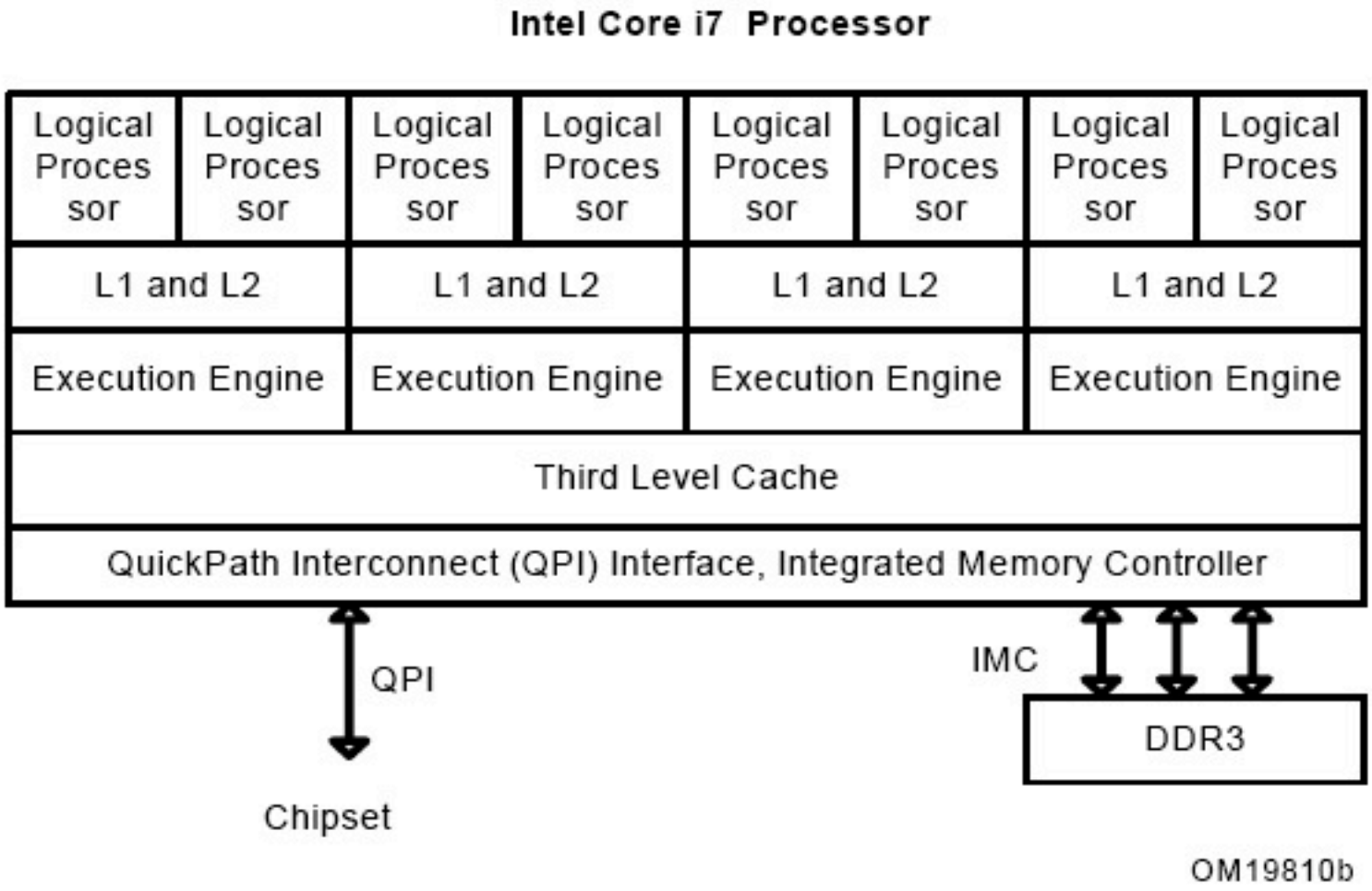
ARCHITETTURA DI UN DUAL-CORE



MULTI-CORE + HYPER-THREADING



QUAD-CORE + HT


Intel Core i7 Processor
Logical Processor
Logical Processor
Logical Processor
Logical Processor
Logical Processor
Logical Processor
Logical Processor
Logical Processor
L1 and L2
L1 and L2
L1 and L2
L1 and L2
Execution Engine
Execution Engine
Execution Engine
Execution Engine
Third Level Cache
QuickPath Interconnect (QPI) Interface, Integrated Memory Controller
QPI
Chipset
IMC
DDR3
OM19810b

ARCHITETTURE MULTI-CORE E "CONCURRENCY REVOLUTION"

- Limiti nell'incremento della potenza di calcolo del singolo processore
  - limiti fisici della tecnologia
  - consumo potenza
  - ...
- ➡ **Investimenti massicci nelle architetture multi-core**
  - prototipi con 80 core (Intel)
- Impatto sui modelli di programmazione delle applicazioni
  - programmazione concorrente diventa mainstream
    - "Software and Concurrency Revolution" (Sutter, Larus) ACM Queue, 3(7),2005
  - quali metodologie di programmazione? quali linguaggi? quali approcci per la costruzione del software (concorrente)?
  - investimenti massicci: Microsoft, Intel

SUPPORTI HW PER VIRTUALIZZAZIONE

- CPU con estensioni specifiche a supporto della virtualizzazione
  - Intel VT-*x*, AMD-V
- Possono essere sfruttate dalle tecnologie software per la virtualizzazione
  - miglioramento performance
- Esempio: estensione **VMX** di Intel (fornita in Intel VT)
  - modalità di funzionamento VMX **root operation**
    - esecuzione del VMM (Virtual Machine Monitor)
    - istruzioni specifiche di gestione delle macchine virtuali
      - VMXONper entrare nella modalità di virtualizzazione
      - VMLAUNCH/VMRESUME per entrare nelle/uscire dalle singole macchine virtuali
      - VMXOFF per chiudere la modalità di virtualizzazione
  - modalità di funzionamento per VMX **non-root operation**
    - esecuzione delle VM, quindi dei relativi sistemi operativi

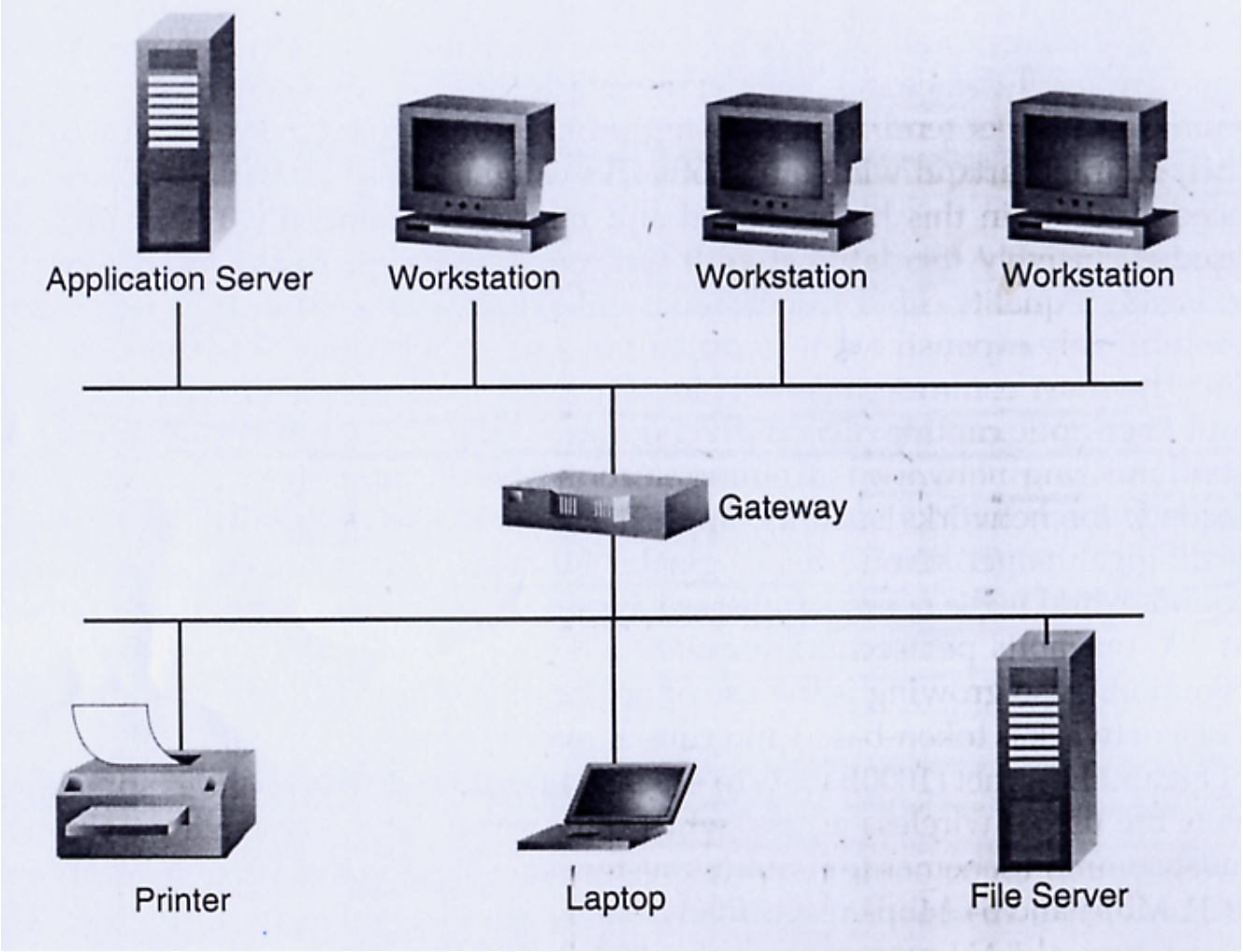
NETWORKING

- Una parte fondamentale dei sistema informatici moderni è data dalla rete (network):
  - sistemi software / hardware più o meno complessi sono realizzati da un insieme più o meno grande e strutturato di computer connessi in rete, in grado comunicare.
    - sistemi distribuiti
  - sistemi a **cluster**
    - insieme di elaboratori in rete integrati in modo supportare l'esecuzione parallela di determinati task
      - es. processo di indicizzazione informazioni (Google), servizi Web, ..
    - es. *Beowulf* clusters
- Tipi di rete principali e più diffusi:
  - **LAN** (Local Area Network)
  - **WAN** (Wide Area Network)
    - Internet
  - **SAN** (Storage Area Network)
    - condivisione della memoria di massa

LAN (LOCAL AREA NETWORK)

- Una LAN è data da un insieme relativamente piccolo di computer e device - es: stampanti - connessi in rete, e localizzati nella medesima area geografica
  - esempi tipici sono uffici, imprese, laboratori.
  - emerse nei primi anni 70.
- Il tipo più diffuso di rete si chiama **Ethernet**,
  - in cui ogni computer / device è connesso a tutte le altri, senza punti di centralizzazione
  - la velocità delle reti ethernet varia da 1Mb (sistemi wireless) a 10/100Mb a 1Gb (fibra ottica)

STRUTTURA DI UNA LAN



WAN (WIDE AREA NETWORK)

- Una WAN (es: **Internet**) connette una moltitudine di computer, sottoreti, LAN distribuite in ampie aree geografiche
  - emerse alla fine degli anni 60, in ambito accademico e di ricerca: il progetto ARPANET è partito nel 1968 con 4 computer, ed è cresciuto in quello che oggi è la rete delle reti, ovvero Internet, con milioni di computer
- Data la mole di computer, queste reti sono strutturate in sottoreti connesse tra loro da appositi link di comunizazione detti **communication processor**, responsabili di interfacciare fra loro reti diverse
- Nel caso di Internet, i communication processor prendono il nome di **router**, che connettono una sottorete ad un'altra

STRUTTURA DI UNA WAN