Calcolatori Elettronici

CDL in Ingegneria Elettronica

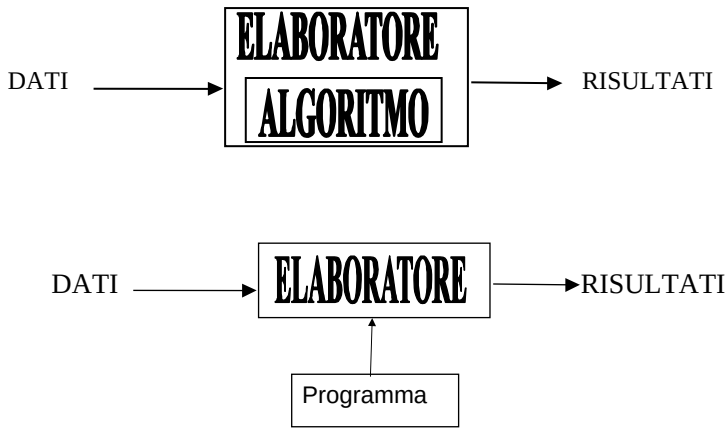
Facoltà di Ingegneria,
Università degli Studi di Firenze

Nuovo Ordinamento

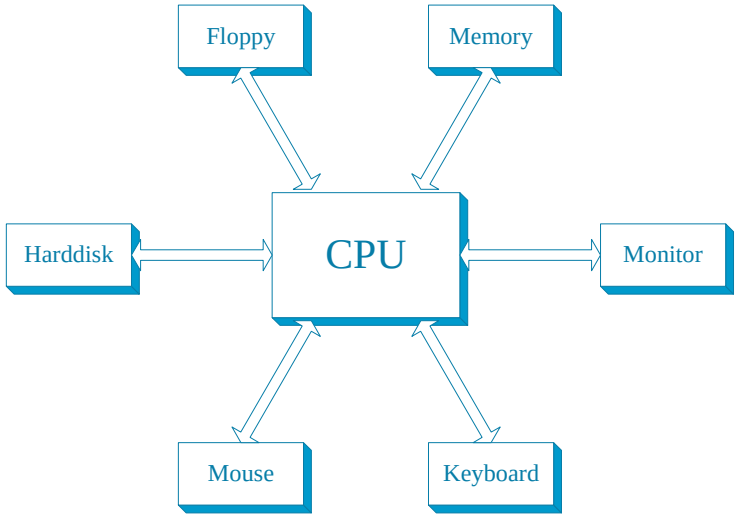
Parte 4, Le architetture degli elaboratori

Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
nesi@dsi.unifi.it
2006

L'elaboratore



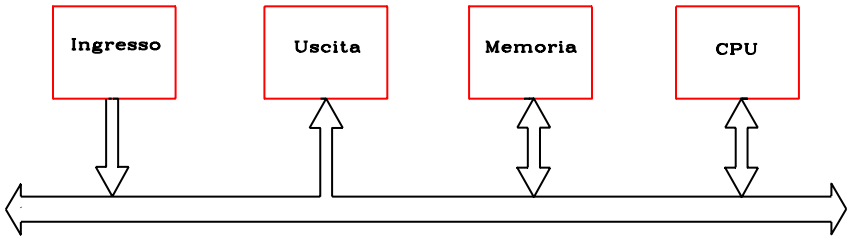
CPU e relazioni funzionali !



Paolo Nesi, Univ. Firenze, Italy, 2003-06 226

Schema di riferimento

- Per Il momento lo schema di riferimento sarà quello di figura
- Corrisponde allo schema dei PC anni 80
- Tuttora in largo uso nei sistemi di controllo
- Le unità sono collegate da un bus di sistema

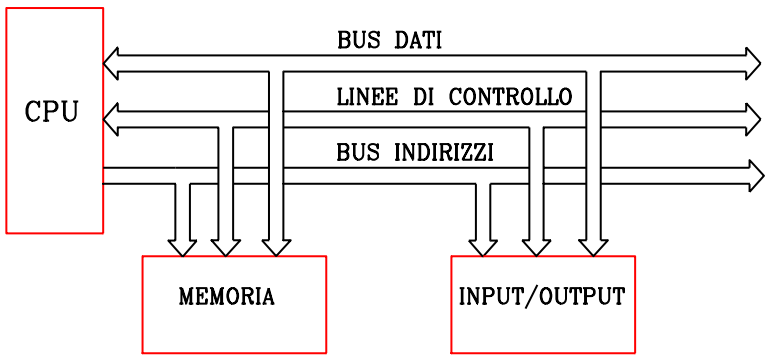


Paolo Nesi, Univ. Firenze, Italy, 2003-06 227

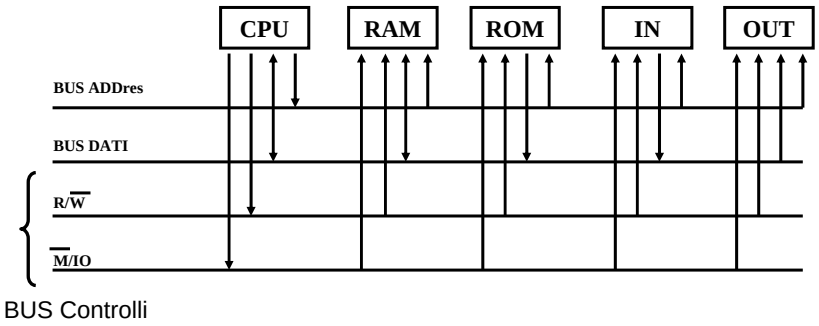
Alcuni Cenni alle periferiche

- **I/O, Input/Output, Ingressi/Uscita**
  - Input: keyboard (tastiera), mouse, scanner
  - Output: monitor, printer
- **Memory, Memoria**
  - Di base: RAM, ROM, stato solido….
  - Di Massa: dischi (HD, FD), nastri,
- **CPU (Central Processing Unit), microprocessore**
  - ALU,
  - unità di controllo,
  - registri,
  - etc.

La Struttura del BUS di Sistema

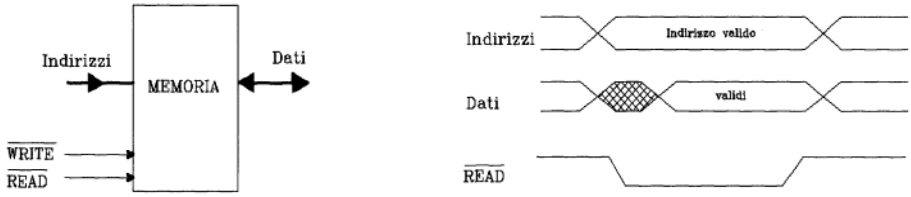


Direzione dei Segnali sul BUS

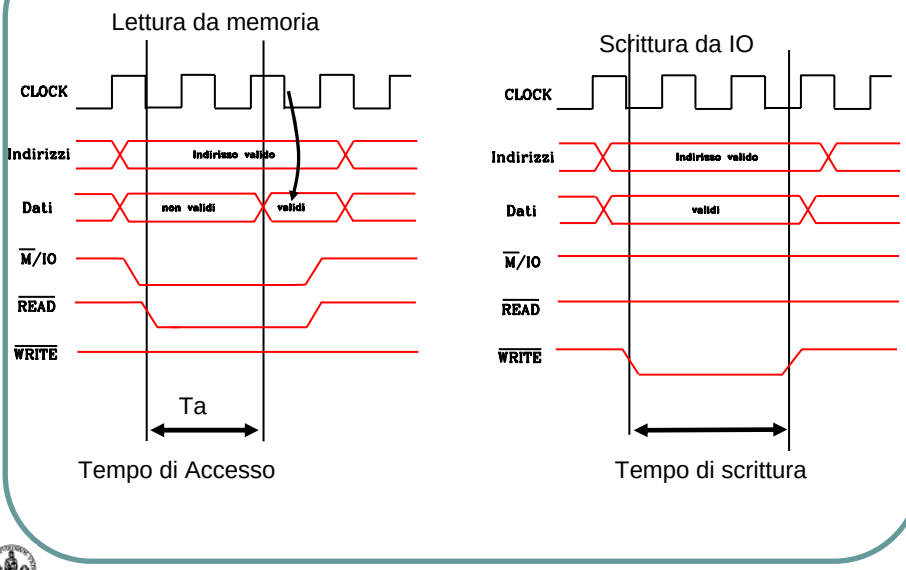


La Memoria e l'elaboratore

- Bus Indirizzi: Bit di Indirizzamento, N bit
- Bus Dati, M bit
- Bus Controlli Segnali di Controllo, Read, Write, Select….
- Tempo di lettura, tempo di accesso
- Si ha una capacita di $M*2^N$ bit



Transazioni sul bus, Ciclo macchina

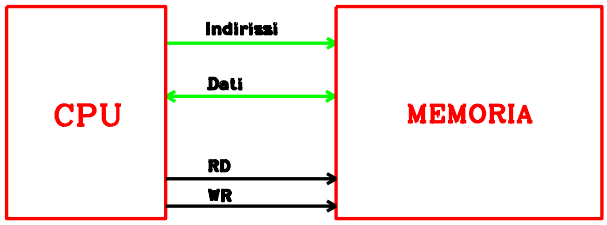


Selezione, Memoria/IO R/W

Not M/IO	not R	Not W	Azione
0	0	0	----none
0	0	1	Read Memory
0	1	0	Write Memory
0	1	1	----impossibile
1	0	0	----none
1	0	1	Read IO
1	1	0	Write IO
1	1	1	---- impossibile

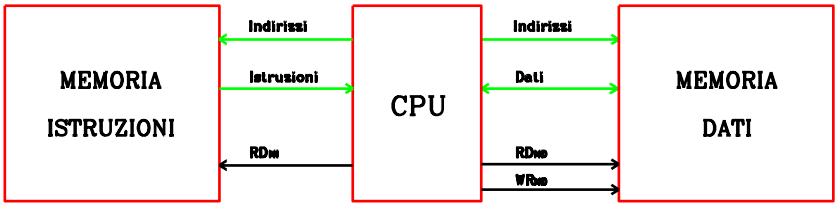
Architettura di Von Neuman

- memoria indifferenziata per dati o istruzioni, solo l'interpretazione da parte di CPU stabilisce se una data configurazione di bit è da riguardarsi come un dato o come un'istruzione



Architettura Harward

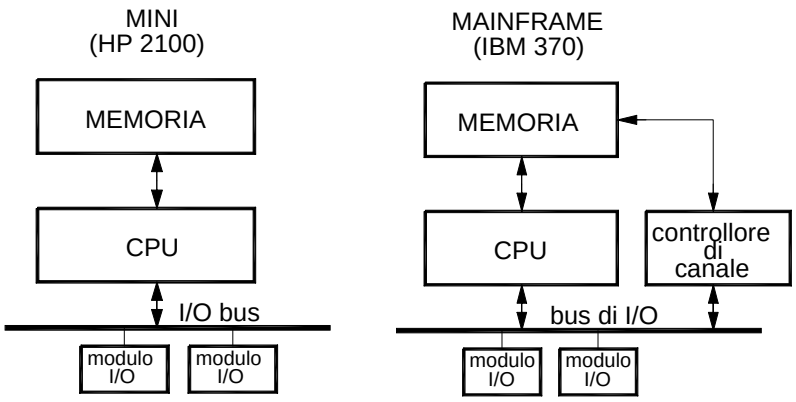
- Due memorie distinte: la memoria istruzioni e la memoria dati.
- Il comando di lettura della memoria istruzioni è superfluo, in quanto si può immaginare che questa memoria sia sempre e soltanto letta



Confronto

- **Von Neuman**
  - Accesso a istruzioni e dati nella stessa memoria
  - Flessibilità nello sfruttamento della memoria
  - Rischio di manipolazione del codice
  - Minore costi di realizzazione
- **Harward**
  - Robustezza alla manipolazione del codice
  - Accesso contemporaneo a codice e dati
  - Costi maggiori di realizzazione
  - Minore flessibilità

Struttura (anni 70)



Un Micro degli anni 70, 4004, 4 bit

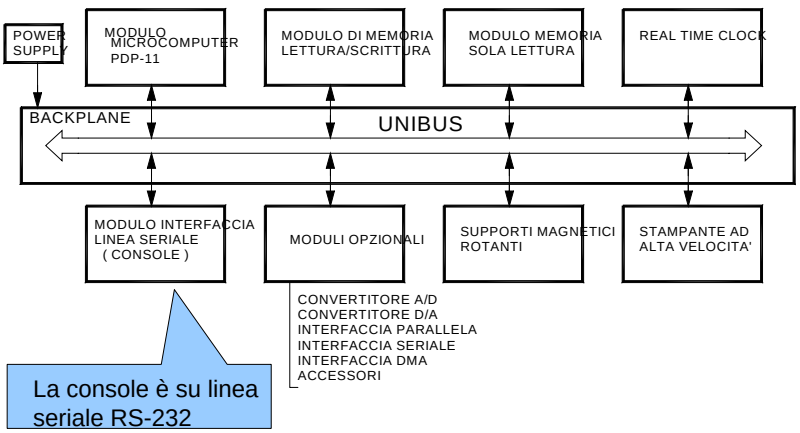
intel MCS-4 MICRO COMPUTER SET

NOVEMBER 1971

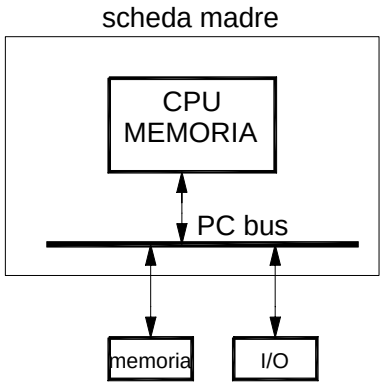
- Microprogrammable General Purpose Computer Set
- 4-Bit Parallel CPU With 45 Instructions
- Instruction Set Includes Conditional Branching, Jump to Subroutine and Indirect Fetching
- Binary and Decimal Arithmetic Modes
- Addition of Two 8-Digit Numbers in 850 Microseconds
- 2-Phase Dynamic Operation
- 10.8 Microsecond Instruction Cycle
- Easy Expansion – One CPU can Directly Drive up to 32,768 Bits of ROM and up to 5120 Bits of RAM
- Unlimited Number of Output Lines
- Single Power Supply Operation (V_{DD} = –15 Volts)
- Packaged in 16-Pin Dual In-Line Configuration

http://smithsonianchips.si.edu/ice/4004.htm

Struttura PDP-11 (anni 70-80)

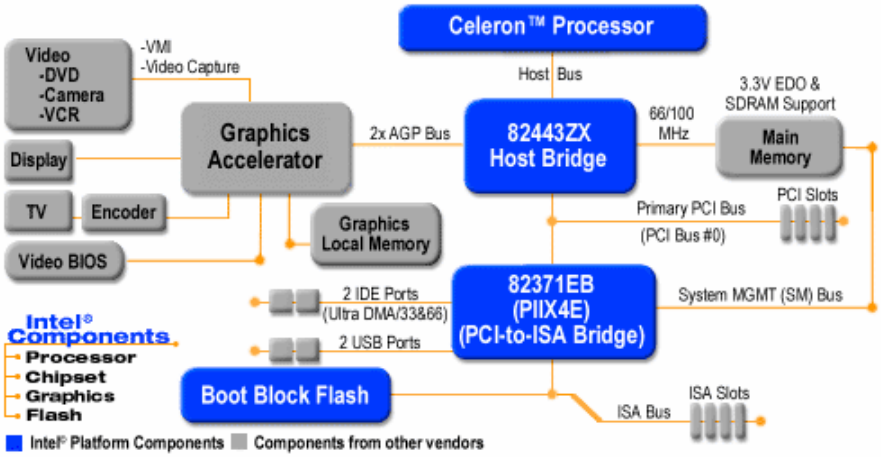


Struttura PC (anni 80)



- E' sparita la console RS232
- Il Video e la tastiera sono collegati direttamente alla scheda madre.
- Per il video c'è una RAM apposita
- Memoria di espansione e periferici sono su schede collegate sul bus

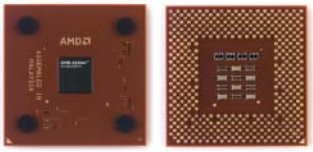
Struttura PC corrente



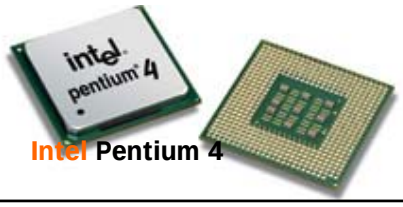
Microprocessore

Un microprocessore o CPU è un dispositivo elettronico ad elevata integrazione, che può svolgere operazioni aritmetiche, logiche e di controllo, su dati generati dal microprocessore stesso o forniti dall'esterno.

Il microprocessore per funzionare, deve essere inserito in una struttura in grado di utilizzarne le potenzialità, fornendo al dispositivo, attraverso programmi definiti dall'utente, le informazioni necessarie, relative al tipo di operazione da eseguire ed i dati su cui operare.

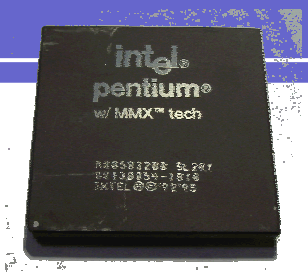


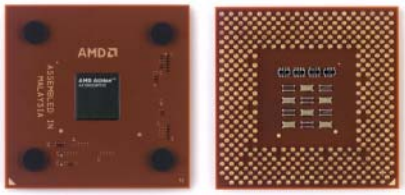
AMD Athlon XP

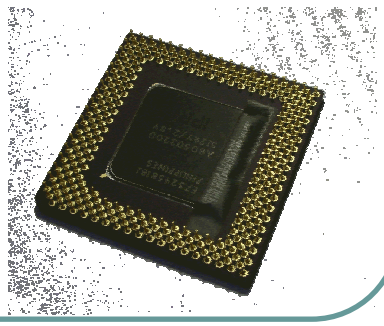


Intel Pentium 4

CPU

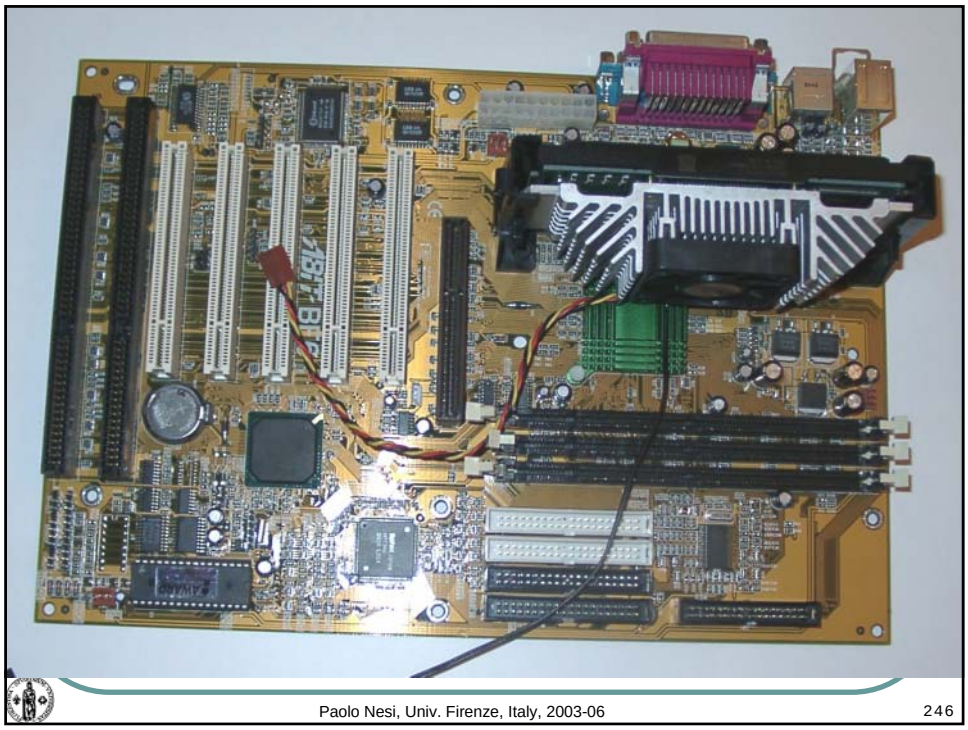






CPU


intel
i386
intel
i486 DX
ADVANCED
MICRO
DEVICES
Am386 DX /DXL-40



Scheda Madri

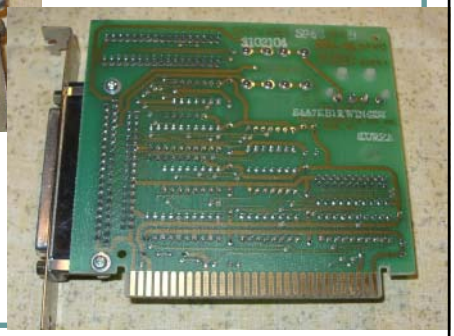
- Si noti:
  - La CPU
  - I BUS: EISA, PCI, Local Bus pe rscheda video,
  - Slot per la memoria
  - Quarzo per il Clock
  - Connettori per IO controllori di disco
  - Connettori/zoccoli per IO porta parallela, seriale, tastiera etc.

Scheda Madre

La sche da madre precedente

- Si noti:
  - Zoccolo per la CPU
  - I BUS: EISA, PCI,
  - Slot per la memoria: due tipi diversi
  - Quarzo per il Clock
  - Connettori per IO controllori di disco
  - Connettori/zoccoli per IO porta parallela, seriale, tastiera etc.

Isa Board



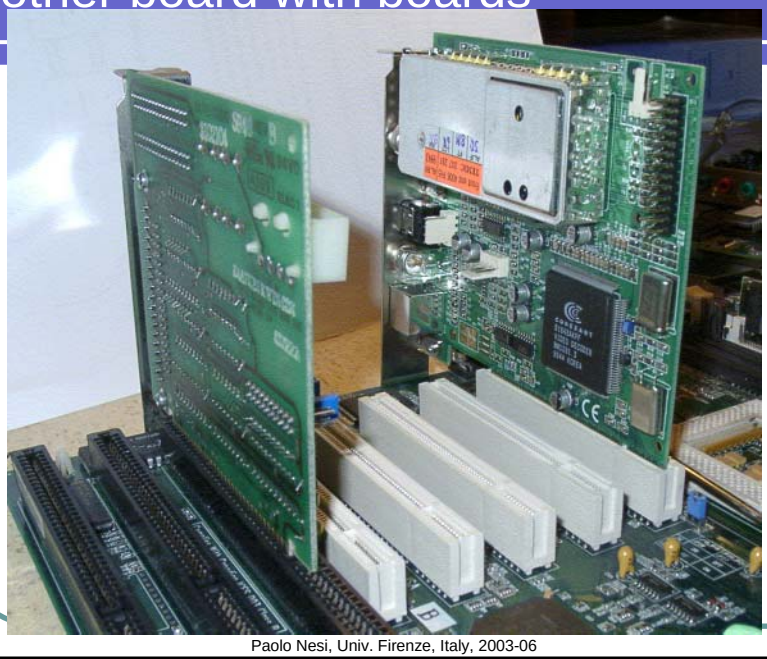
Paolo Nesi, Univ. Firenze, Italy, 2003-06 250

EISA Board

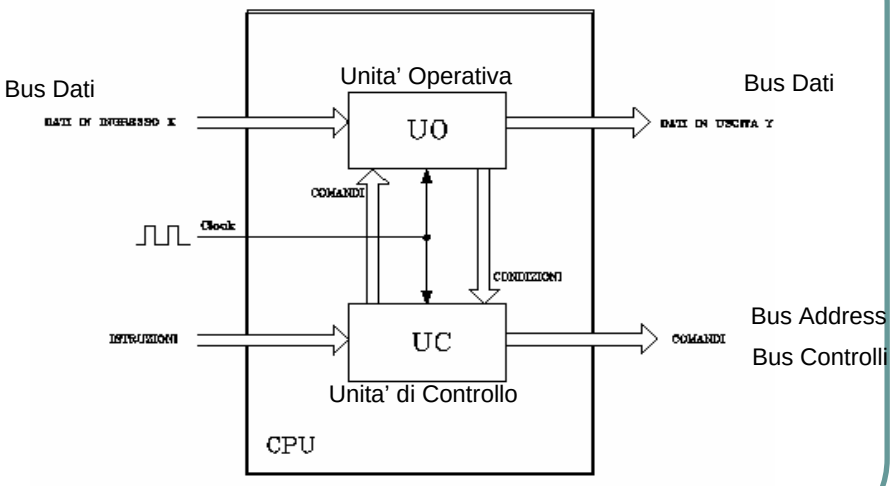
Paolo Nesi, Univ. Firenze, Italy, 2003-06 251

PCI Board

Mother board with boards

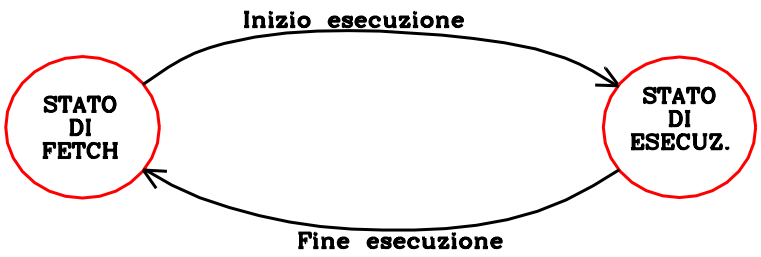


Schema aggregato della CPU

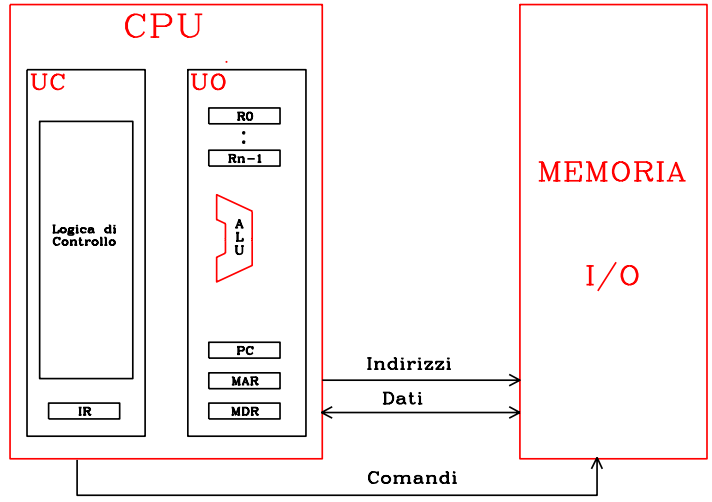


Fetch-Esecuzione

- **Fetch:** Prelievo e decodifica dell'istruzione
  - Fase comune a tutte le istruzioni
- **Esecuzione:** Fase in cui vengono eseguite le azioni previste dal codice di operazione
  - Fase diversa da istruzione a istruzione



Componenti essenziali



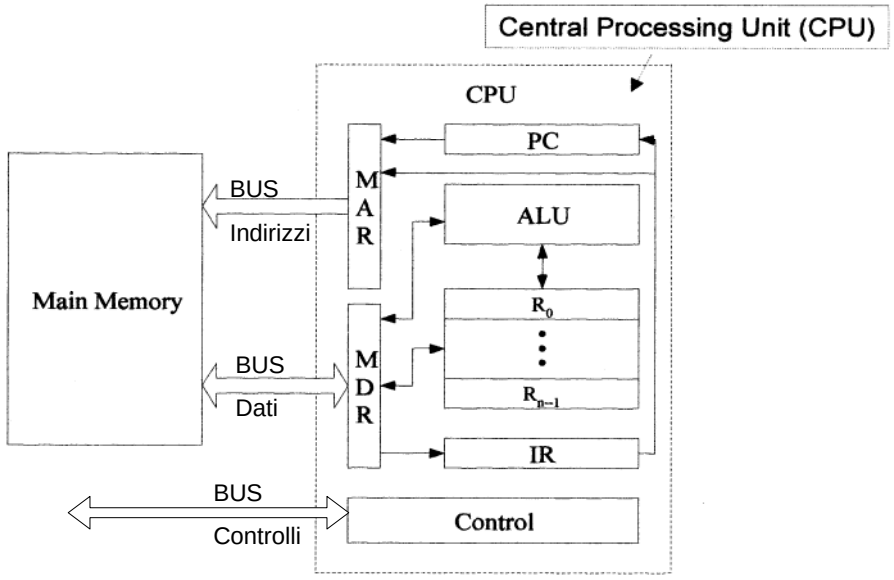
Paolo Nesi, Univ. Firenze, Italy, 2003-06 256

La Gerarchia delle Memorie

- I registri interni sono delle memorie molto veloci ai quali si fa riferimento direttamente dalla ALU, tempi di accesso dell'ordine di 10 nsec
- La struttura della Memoria a stato solido e' stata gia discussa
- RAM
  - più lenta dei registri interni
  - velocità dell'ordine dei 40 – 80 nsec, nanosecondi
  - DRAM, RAM
  - per i programmi applicativi
- ROM
  - per garantire il funzionamento all'accenzione,
  - contiene le prime istruzioni che vengono eseguite all'accensione

Paolo Nesi, Univ. Firenze, Italy, 2003-06 257

Maggiore Dettaglio relaz. M-CPU



Paolo Nesi, Univ. Firenze, Italy, 2003-06 258

Registri di CPU

- **MAR: Memory Address Register**
  - contiene l'indirizzo della locazione di memoria da leggere o scrivere.
  - La dimensione di MAR determina l'ampiezza dello spazio di memoria fisica; dalla fine degli anni '80 vengono prodotti microprocessori con bus indirizzi a 32 bit
- **MDR: Memory Data Register**
  - registro attraverso il quale viene scambiata l'informazione tra la memoria e la CPU
  - Tradizionalmente MDR dà la misura del grado di parallelismo della macchina (8, 16, 32, 64 bit) La dimensione di
- **R0, R1,...Rn: registri di uso generale**
  - Registri di uso generale,
  - elevata velocità,
  - memorizzazione temporanea dei dati,
  - realizzati come serie di FF

Paolo Nesi, Univ. Firenze, Italy, 2003-06 259

Elementi delle CPU

- **UC: Unità di Controllo, Control Unit**
  - Decodifica le istruzioni contenute nell'IR e genera i segnali di controllo
  - Controlla le altre unità al fine di completare l'istruzione data in IR
  - Produce i segnali che escono dalla CPU
  - Legge alcuni segnali che entrano nella CPU, per esempio il Clock…..
- **UO: Unità operativa**
  - Contiene i registri
  - Contiene la ALU

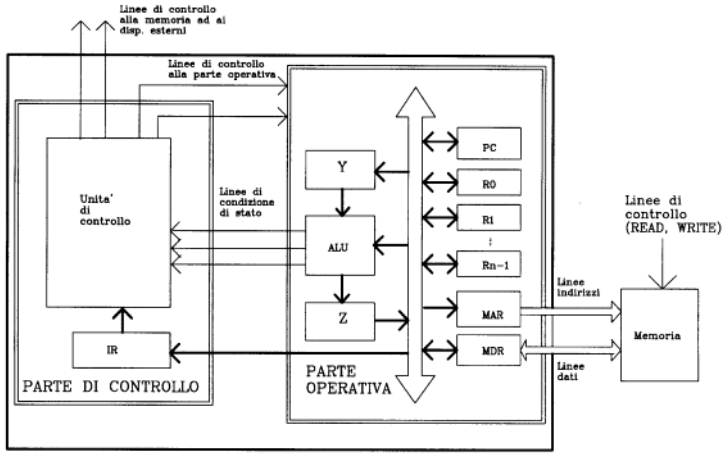
Registri di CPU

- **IR: Instruction Register**
  - usato per contenere l'istruzione in corso di esecuzione.
  - Caricato in fase di fetch dalla memoria.
  - Rappresenta l'ingresso che determina le azioni svolte durante la fase di esecuzione.
  - Dall'IP viene decodificata l'istruzione
- **PC: Program Counter**
  - tiene traccia dell'esecuzione del programma
  - Contiene l'indirizzo di memoria della prossima istruzione
  - Viene aggiornato per indirizzare l'istruzione successiva o parti di questa

Esempio di sequenza di istruzioni

- M[3]=M[1]+M[2]
  - R6 ← M[1] 6 ck
  - R7 ← M[2] 6 ck
  - R8 = R6 + R7 3 ck
  - M[3] ← R8 6 ck
  - 21ck

Maggiore Dettaglio vicino a ALU

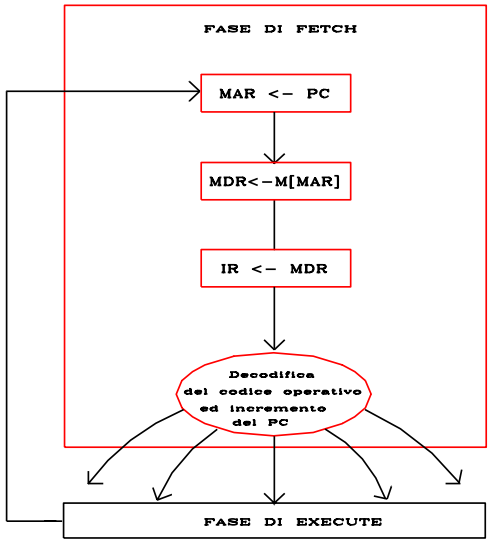


- + File Register con flag di: zero, carry, overflow, sign, etc., per mantenere lo stato della ALU
- BUS Interno

ALU, Arithmetic Logic Unit

- Esegue operazioni aritmetiche e logiche
- I risultati sono scritti nei registri o in memoria
- I dati di partenza sono letti dai registri o dalla memoria
- I due precedenti punti possono avere delle restrizioni dipendentemente dalla architettura per esempio architetture con registro accumulatore, etc.

Fase di Fetch



MAR ← PC

Not READ

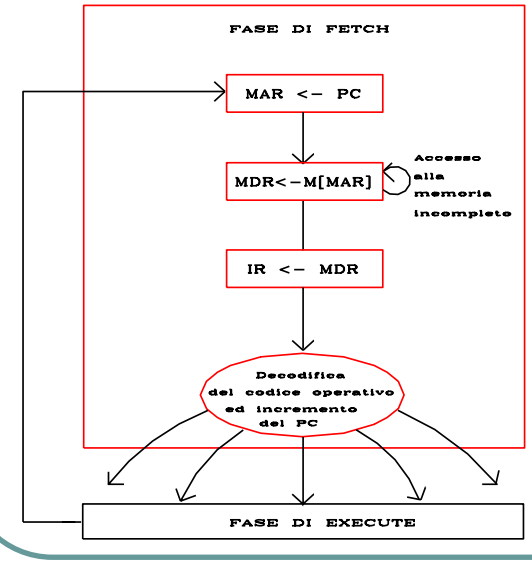
MDR ← M[MAR]

IR ← MDR

Decodifica dell'isttruz.

PC ← PC + 1

Fase di Fetch, IST multibyte



MAR ← PC

Not READ

MDR ← M[MAR]

IR ← MDR

Decodifica dell'isttruz.

PC ← PC + 1

……

PC ← PC + 1

Su più letture per completare la lettura dell'istruzione che può essere su piu' byte o word…..

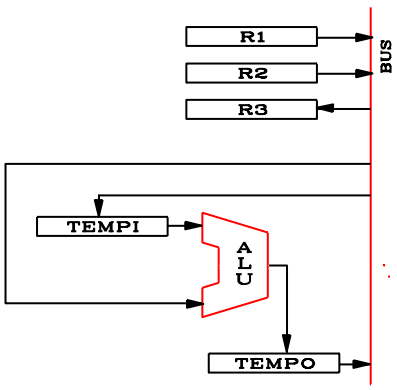
L'Esecuzione dell'istruzione

- **Fetch dell'istruzione,** IF
  - Fetch per come e' stato visto
- **Decodifica dell'istruzione**, decode, ID
  - Decodifica dipendentemente dal tipo di codifica
- **Esecuzione** dell'istruzione, execution, EXE
  - Può richiedere ALU or gestione generale
  - Può implicare la lettura da memoria
- **Scrittura dei Risultati,** write back, WB
  - Scrittura dei risultati in memoria o su I/O

A che punto siamo, Cosa vediamo ora

- Abbiamo visto
  - Instruction fetch, l'acquisizione delle istruzioni
- Vediamo ora
  - L'esecuzione delle istruzioni
- In seguito
  - La decodifica delle istruzioni

Data Path - 1 bus



Esecuzione in più passi.
Esempio: **ADD R3,R1,R2**
"Cioe' R3=R1+R2"

Richiede almeno **tre periodi di clock**:

- R1_out; TEMPI_in
- R2_out; TEMPI_out; ADD; TEMPO_in
- TEMPO_out; R3_in

Segnali per buffer Tri-State

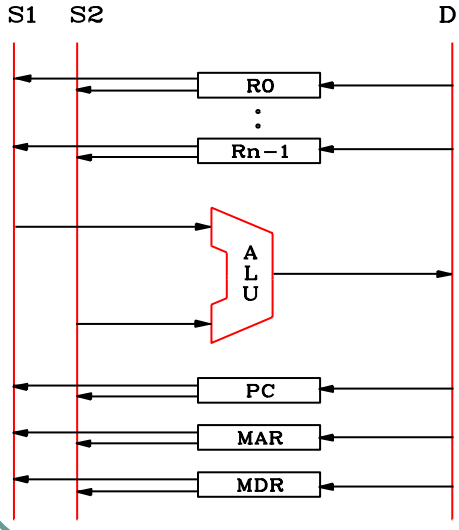
Decodifica dell'istruzione, idea

- Supponiamo che per ogni fase si possano fare 16 cose diverse (per esempio comandare 16 segnali diversi: registri, etc., R1in, R1out, etc.), allora si ha bisogno di 4 bit per ogni fase.
- Se ho tre fasi si arriva ad una istruzione di 3*4 bit, 12 bit.
- Il decoder deve accettare istruzioni di N bit, se ho 2^N istruzioni diverse.
- Per ogni istruzione viene prodotta la decodifica come per esempio i 12 bit di cui sopra.
- Non e' detto che se ho 12 bit, ho 2^{12} istruzioni diverse, certe combinazioni sono impossibili o da evitare per evitare danni.

Istruzioni con indirizzi

- ADD M[A1],R3,M[4F]
- Che significa: M[A1] = R3 + M[4F]
  - Implica avere dentro l'istruzione anche l'indirizzo
  - Se per ogni termine si puo' avere un indirizzo e lo spazio di indirizzamento e' di 12 bit, si hanno istruzioni con almeno 36 bit.

Data Path - 3 bus



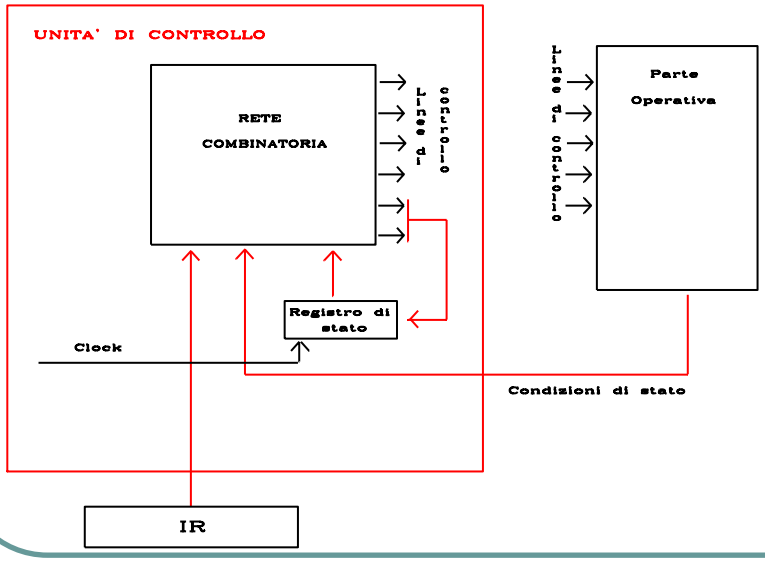
L'istruzione

ADD R3,R2,R1

viene eseguita in un **solo clock**:

- R1_out; R2_out; ADD; R3_in

Logica cablata



Logica Cablata

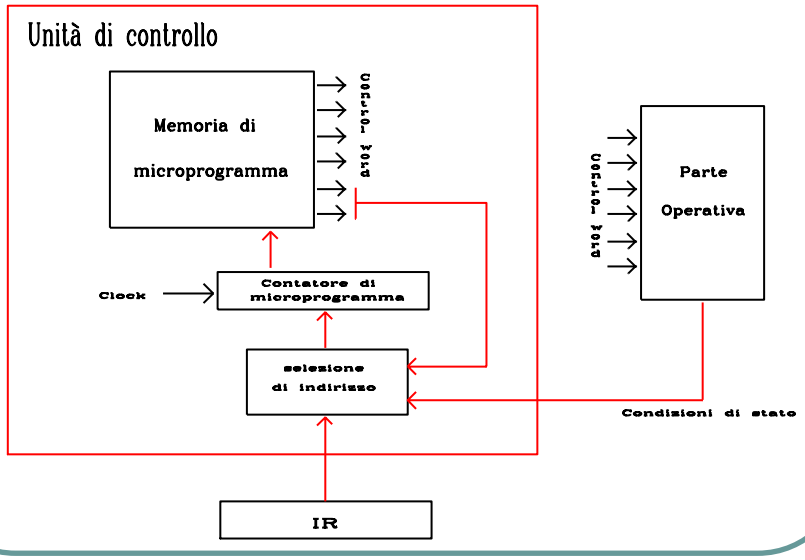
- Tabella con ingressi:
  - IR +
  - registro di stato (incluso contatore delle fasi) +
  - variabili di stato relative alle uscite
- Contatore per le fasi (per esempio 4 bit controllare 16 registri/azioni)

Ist	cod1	cod2	cod3
000	0010	0100	0000
001			
010			
	F1	F2	F3

Logica cablata

- Progetto
  - Come sintesi di rete sequenziale
    - Sintesi per 1 (+di*), pochi 1, ROM sparsa
    - ingressi: IR, stato di UO
    - uscite: comandi
  - Uso di ROM. La rete combinatoria ha come
    - ingressi (indirizzi alla ROM): IR, stato di UO, stato di UC (ALU flag register)
    - uscite: comandi che sono ingressi di eccitazione dei FF di stato, stato futuro
  - Logica programmabile (PLA)
  - Progettazione con CAD per VLSI
- Misura della complessità di UC:
  N_stati x N_ingressi x N_uscite

UC microprogrammata



UC microprogrammata

- Tecnica affermatasi negli anni 70
- UC è una sorta di calcolatore nel calcolatore
- La memoria di controllo contiene le microistruzioni:
- **mPC**: contatore di microprogramma. Contiene l'indirizzo della prossima microistruzione
  - All'inizio della fase di fetch mPC contiene l'indirizzo (I_0) del tratto di microprogramma corrispondente al fetch
  - Alla fine della fase di fetch mPC viene aggiornato con il contenuto (o ad una opportuna decodifica) di IR in modo da puntare alla microroutine che effettua le azioni richieste dalla particolare istruzione
  - Al termine, mPC viene di nuovo caricato con (mI0)

UC microprogrammata

- Differenti soluzioni:
  - sequenza di micro *control word* (CW)
  - microprogramma con subroutine
  - microistruzioni che contengono codificato al loro interno l'indirizzo della prossima microistruzione
  - *nanoprogrammazione:* le microistruzioni sono a loro volta interpretate da una unità nanoprogrammata (68000)
- Microprogrammazione orizzontale:
  - ogni bit di una CW corrisponde ad una linea di comando
- Microprogrammazione verticale
  - le CW contengono i comandi in forma convenientemente codificata
- *Bit sliced microprocessors* (AMD 29000)

Esempio, ADD R3,R2,R1

1a IST	R1out	00001
	TEMPIin	00101
2a IST	R2out	01001
	TEMPIout	……..
	ADD	……..
	TEMPOin	……..
3a IST	TEMPOout	……..
	R3in	……..

N Bit di codifica della mIST, 2^N possibili mIST,
Buffer da controllare

Cablata o microprogrammata?

- Fino a fine anni '60: logica cablata (PDP8, HP 2116)
- Anni '70: microprogrammazione (VAX, Z80, 8086, 68000)
  - Repertorio di istruzioni molto esteso e variato: **CISC**
  - Il VAX 11/789 (Digital) e il 370/168 (IBM) avevano oltre 400.000 bit di memoria di controllo
- Dagli anni '80 si è tornati alla logica cablata;
  - Affermazione delle macchine **RISC**
- Istruttivo è esaminare l'evoluzione dell'architettura Intel: da CISC a (praticamente) RISC

Ragioni per le CISC

- Un repertorio di istruzioni esteso è preferibile perché:
  - Istruzioni potenti semplificano la programmazione
  - Riduce il gap tra linguaggio di macchina e linguaggio di alto livello
  - *Software crisis*
- L'uso efficiente della memoria (all'epoca era costosa) era la preoccupazione principale:
  - meglio avere codici compatti
- Essendo (allora) la memoria di controllo molto più veloce della memoria centrale, portare funzionalità nella prima avrebbe migliorato le prestazioni della macchina

...Tuttavia

- Memorie RAM: molto più veloci delle precedenti a nuclei
- *Cache:* riducono ulteriormente i tempi di esecuzione
- Comportamento dei programmi:
  - l'80% delle istruzioni eseguite corrispondeva al solo 20% del repertorio.
  - ⇒ *conviene investire nella riduzione dei tempi di esecuzione di quel 20%, anziché aggiungere raffinate istruzioni, quasi mai usate, ma responsabili dell'allungamento del tempo di ciclo di macchina*
  - ⇒ *conviene costruire processori molto veloci, necessariamente con repertori semplici, e contare sull'ottimizzazione del compilatore*

Calcolatori Elettronici

CDL in Ingegneria Elettronica

Facoltà di Ingegneria,
Università degli Studi di Firenze

Nuovo Ordinamento

Parte 5, L'architettura Software e la Scelta

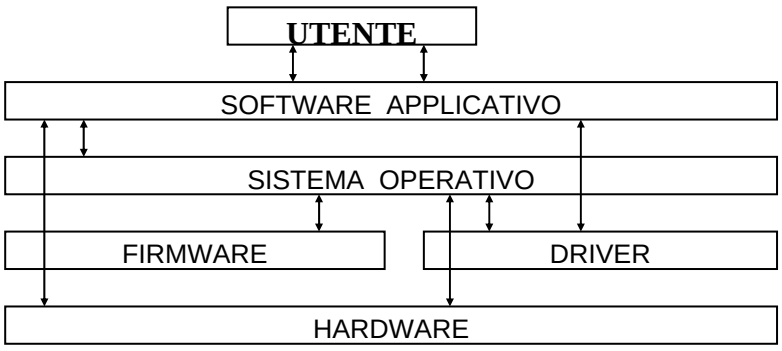
Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
nesi@dsi.unifi.it
2006

Sommario della parte 5

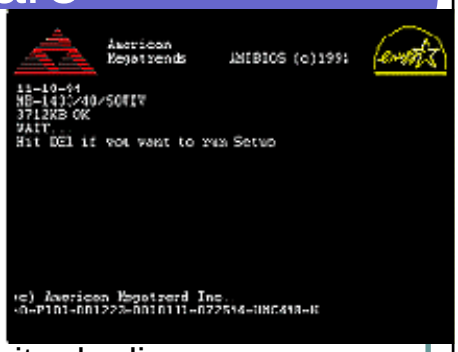
- **Hardware e Software**
- **L'architettura Software**
  - Il firmware, il sistema operativo
  - I driver
  - Il Boot
  - i programmi applicativi
- **Le Misure di un Elaboratore**
- **Le memorie di Massa**
- **Le prestazioni dei sistemi a microprocessore**

Hardware e Software

L'elaboratore è composto dall'hardware, i dispositivi fisici che lo costituiscono, e dal software, quelle procedure e istruzioni che ne dirigono le operazioni.



Hardware e Software



Il software è parte integrante del calcolatore poiché ne permette una maggiore o minore adattabilità alle richieste dell'utente stesso.
La struttura software è costituita da diverse categorie di programmi:

B.I.O.S. (basic input output system)
O.S.(operating system): KERNEL, UTILITY
SOFTWARE APPLICATIVO.

Hardware e Software

Il BIOS

Una particolare categoria di software è il BIOS (o firmware), un insieme di programmi, inseriti nell'hardware alla costruzione e copiati in memoria centrale all'avviamento (bootstrap). Sono utilizzati dal sistema operativo per accedere alle risorse hardware del sistema. Grazie a questi programmi il sistema operativo può non considerare la struttura interna dell'elaboratore. All'acquisto di un personal computer si ha l'hardware ed il BIOS su cui si può implementare il sistema operativo più adatto a gestire, col software applicativo, le applicazioni più varie.

Hardware e Software

IL SISTEMA OPERATIVO (OPERATING SYSTEM) è un insieme di programmi, che agisce da intermediario tra il calcolatore e l'utente, cosicché questi non debba interagire direttamente con l'hardware. Il Sistema Operativo, infatti, rende possibile l'esecuzione del software applicativo in modo trasparente all'utente. Questi infatti, consegna i dati richiesti al sistema operativo, che scende all'hardware, senza che l'utente ne conosca i complicati meccanismi.

Esempi di sistemi operativi sono:
MS DOS, WINDOWS 9x/ME/NT/2000/XP
LINUX, UNIX VMX
IBM OS2, MAC OS, ecc.

Le misure di un Elaboratore

- **Architettura:**
  - struttura della CPU: VN, HW, RISC, CISC, DSP
  - #Bit di CPU: tipico 32 o 64 bit
  - #bit di BA: tipico 32 bit
  - #bit di BD: tipico 32 bit
  - Clock consentito
- **Scheda Madre:**
  - per CPU singola o multipla,
  - come sfrutta la CPU, il CHIP set: DMA, PIC, Bus Control, etc.
  - periferiche integrate: Disk Controller, raid, audio, USB, etc.
  - Memoria cache (min e max): tipico 512 Kbyte
  - Memoria RAM (min e max): tipico 256 Mbyte, 512 Mbyte
  - Clock massimo e minimo
  - Tipo o tipi di BUS e loro velocità (clock):
    - SA, EISA, VESA, etc. tipico 100 Mbyte
    - Numero di slot liberi

Elaboratore e le periferiche

- **Memoria di Massa**:
  - HD tipico 80 Gbyte, Raid, etc.
  - HD, CD, FD, ZIP, Mcard, etc.
  - DVD+R, DVD+-RW, etc.Tapes
- **user Interface:**
  - video, audio, joistick, mouse
- **I/O:**
  - USB, parallela, seriale, IRDA, 1394, etc.
- **Comunicazione:**
  - Rete fissa o wireless, velocità: 10-100 Mbps
  - Protocolli di comunicazione: TCP/IP, PPP, etc.
  - Modem: ADSL, GPRS, etc.
  - etc.

Memoria di Massa:

- Mobili
  - FD, Floppy disk
  - ZIP,
  - CD, Compact Disk
  - DVD: DVD+R, DVD+-RW, etc.
  - Tapes
  - Memory card, etc.
- Fissi
  - HD tipico 80 Gbyte,
  - Raid, etc.
- Etc..

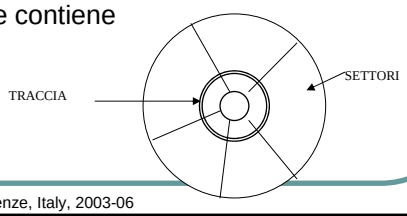
Memorie di massa - Floppy Disk

•Sono di materiale plastico ricoperti di materiale magnetico.

•Ognuno dei due lati è diviso in settori, divisi a loro volta in tracce. I settori sono degli spicchi mentre le tracce sono concentriche. I dischi magnetici sono letti per tracce concentriche. Capacità di 1,44 MB.

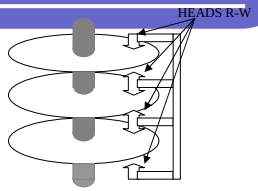
Il tempo di accesso ad una traccia è in funzione della sua posizione sulla superficie del dischetto ed è mediamente di circa 25 millisecondi (l'accesso all memoria RAM è di circa 30 nanosecondi).

Si ha un accesso diretto (quando questa passa sotto la testina) alla traccia e sequenziale all'interno della traccia. Una traccia tipicamente contiene 512 o 1024byte.



Memorie di massa - Hard Disk

Composto da dischi di materiale metallico, rigidi, generalmente d'alluminio, raccolti in una pila e sigillati all'interno di un contenitore dotato di un dispositivo di rotazione e di testine di lettura/scrittura.

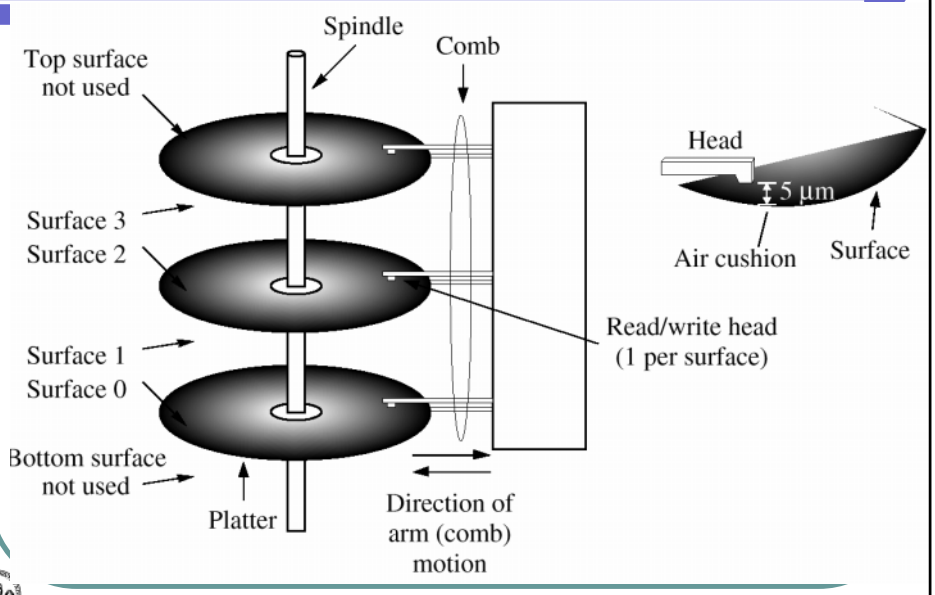


Nell'hard disk, per trovare un'informazione è necessario conoscere la traccia, il settore, il lato/side e il disco sulla quale essa si trova.

Hard Disk Maxtor Diamondmax Plus 9, **80gb Eide**, **Ultra Ata/133**, **8.7ms** Tempo Acc, **2mb Cache**, **7200 Rpm**

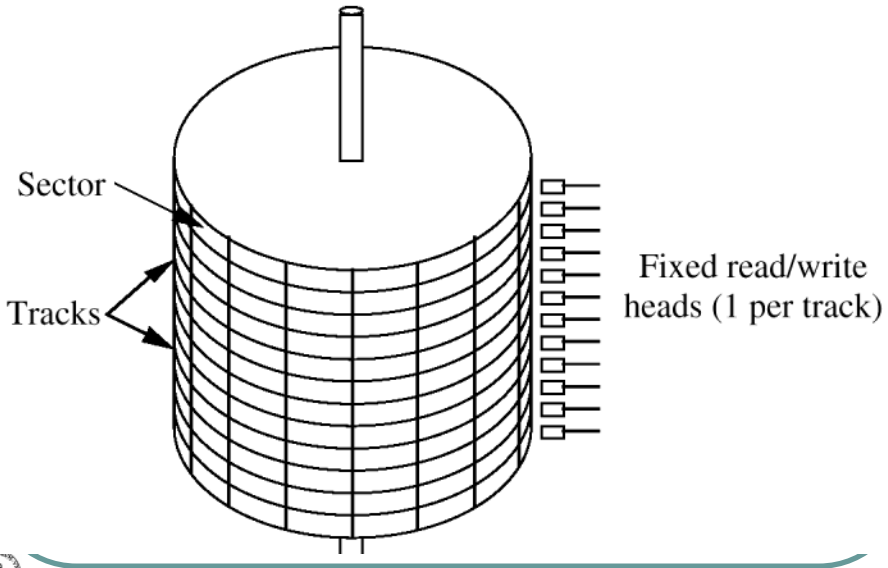
Poiché i dischi sono di materiale rigido, essi sopportano velocità più alte (RPM) ed il tempo di accesso per leggere un'informazione sull'hard disk è inferiore ai 10 millisecondi. La tecnologia impiegata è magnetica, le capacità raggiungibili fino a ~120GB.

Hard Disk



Paolo Nesi, Univ. Firenze, Italy, 2003-06 296

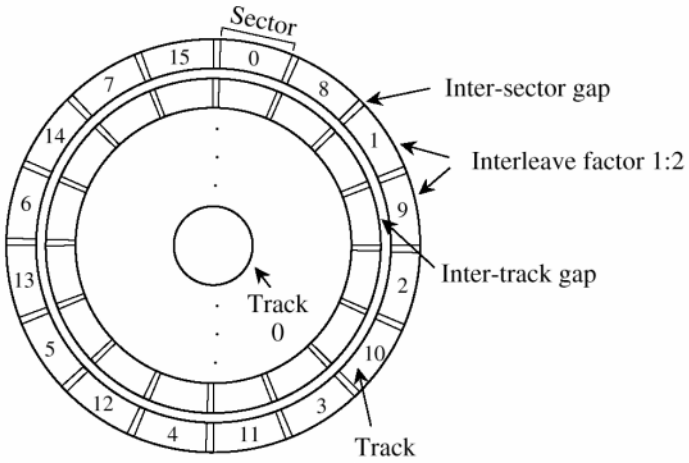
Dischi Magnetici



Paolo Nesi, Univ. Firenze, Italy, 2003-06 297

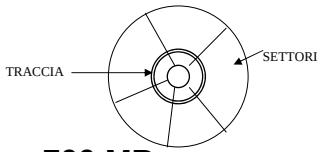
Organizzazione dei dati nei dischi

Organization of a Disk Platter with a 1:2 Interleave Factor



Data Cd (Cd-Rom, Cd-r/w, DVD)

Il CD si legge per piste concentriche come il floppy, andando dall'interno verso l'esterno del disco. Il tempo di acceso è in funzione del driver in dotazione ma è dell'ordine di 1/(2-10) volte meno quello di un Hard Disk. La velocità di lettura si misura con multipli della velocità di lettura dei CD Audio (170 kbytes/sec = 1x) . Oggi sono in commercio modelli in grado di leggere fino a 50x.



- Tecnologia impiegata: ***ottica***
- Capacità di memorizzazione:**CD, fino a 700 MB; DVD, fino a ~9GB**
- Velocità di accesso ai dati: ***media**, nell'ordine delle centinaia di ms*

Unità di Backup o a nastro

Le memorie a nastro sono supporti magnetici solitamente utilizzati per copie di riserva (backup). Il nastro non è una memoria ad accesso diretto ma ad accesso sequenziale: devo cioè svolgerlo ed avvolgerlo fino a trovare il dato che mi interessa (quindi il tempo di accesso può essere elevatissimo). La capacità dei nastri va da 60 Mbyte a 40 Gbyte circa.

I loro punti di forza stanno nella capacità di memorizzazione e nel basso costo.

Unità a nastro (DAT)

Tape Library

Data Cartridge

Memorie di massa - Altri supporti removibili

IOMEGA ZIP

- Tecnologia impiegata: ***magnetica***
- Capacità di memorizzazione: ***100Mb***
- Velocità di accesso ai dati: ***media***

USB Memories ……

- Tecnologia impiegata: flash memory
- Capacità di memorizzazione: ***1Gbyte***
- Velocità di accesso ai dati: ***medio alta***

Interfacce utente

- **Output**
  - **Adattatori Video**
  - **Registratori TV**
  - **Stampanti, Printer**
  - **Etc.**
- **Input**
  - **Mouse**
  - **Joistick**
  - **Mouse**
  - **tavoletta**
  - **Hand traking**

I/O, Input/Output, Comunicazioni

- **Comunicazioni 1:1**
  - USB,
  - parallela,
  - seriale,
  - IRDA,
  - 1394,
  - etc.
- **Comunicazioni N:M**
  - Rete fissa o wireless, velocità: 10-100 Mbps
  - Protocolli di comunicazione: TCP/IP, PPP, etc.
  - Modem: ADSL, GPRS, etc.
  - etc.

Le prestazioni

- MIPS (Milioni di istruzioni al secondo)

$$MIPS = N\_ist / (T\_CPU * 10^6)$$

E' un indice che ha poco significato

- MFLOPS (Milioni di istruzioni in virgola mobile al secondo)

$$MFLOPS = N\_vm / (T\_CPU * 10^6)$$

- Ci sono indici migliori
  - SpecINT 95
  - SpecFP 95

Il processore: prestazioni

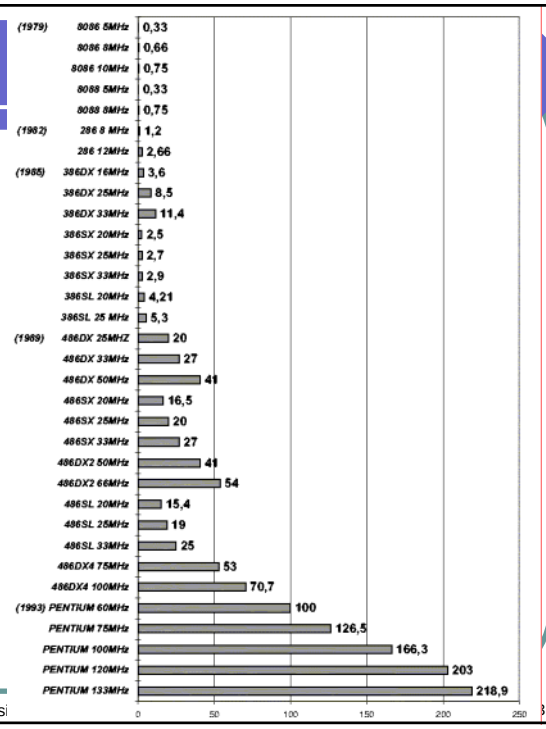
Esistono diversi criteri per valutare la *performance* di un processore.

Sono tutti più o meno discussi o discutibili.

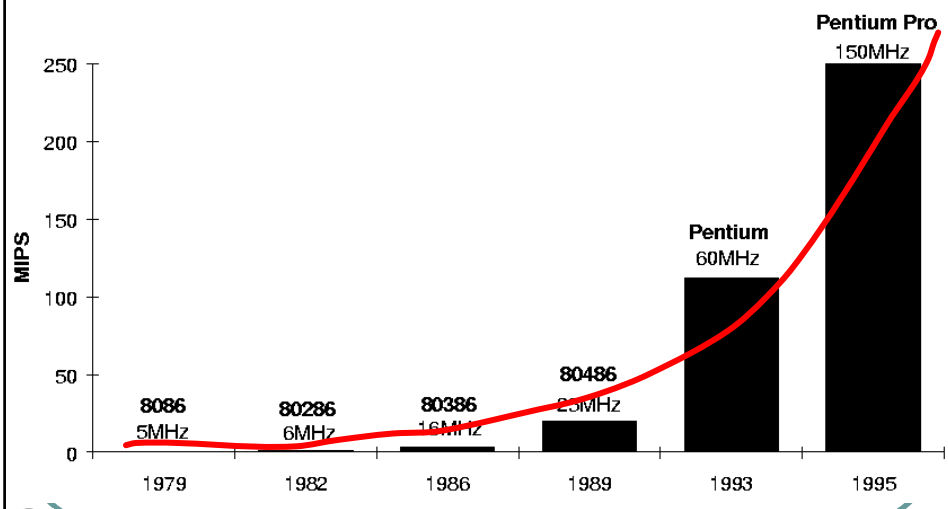
- Velocità di clock (GHz)
- Mips (millions instructions per second)
- MFlops (millions floating-point operations per second)
- iCOMP, iCOMP 2.0 (Intel COmparative Microprocessor Performance)
- P-rating, SPECint, SPECfp (real-world benchmarks)

MIPS

Da 8086 a Pentium Pro



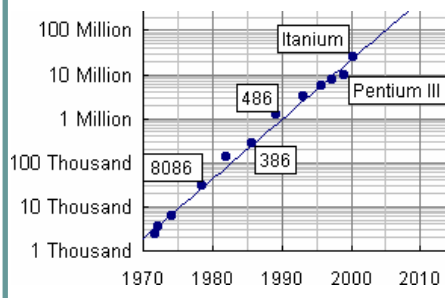
Da 8086 a Pentium Pro

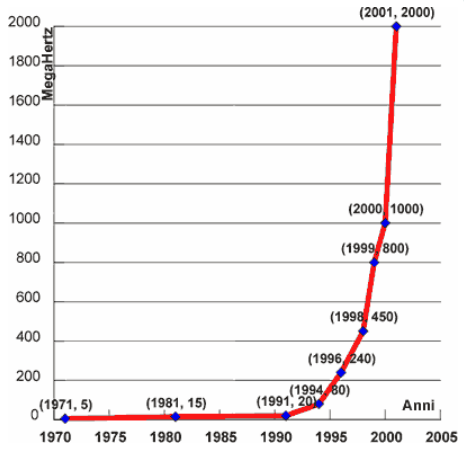


Paolo Nesi, Univ. Firenze, Italy, 2003-06 308

Evoluzione dei Microprocessori

Numero di transistor impiegati nei processori





Frequenza di clock dei processori

Paolo Nesi, Univ. Firenze, Italy, 2003-06 309

Calcolatori Elettronici

CDL in Ingegneria Elettronica

Facoltà di Ingegneria,
Università degli Studi di Firenze

Nuovo Ordinamento

Parte 6, La CPU 8086

Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
nesi@dsi.unifi.it
Gennaio 2006

L'8086, il data sheet

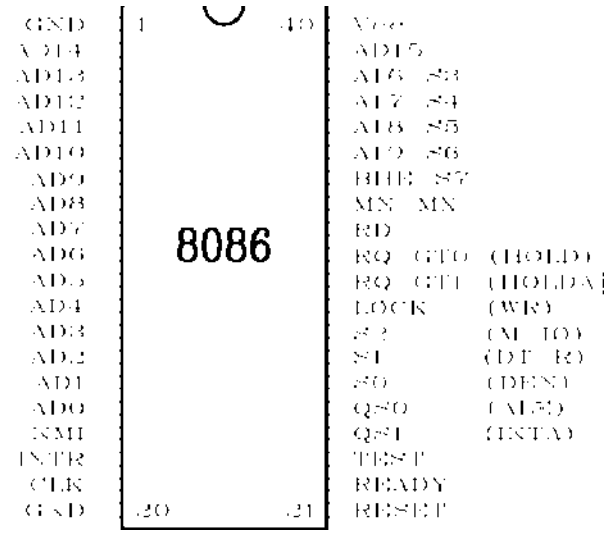
Features

- Compatible with NMOS 8086
- Completely Static CMOS Design
  - DC 5MHz (80C86)
  - DC 8MHz (80C86-2)
- Low Power Operation
  - ICCSB ..... 500μA Max
  - ICCOP ..... 10mA/MHz Typ
- 1MByte of Direct Memory Addressing Capability
- 24 Operand Addressing Modes
- Bit, Byte, Word and Block Move Operations
- 8-Bit and 16-Bit Signed/Unsigned Arithmetic
  - Binary, or Decimal
  - Multiply and Divide
- Wide Operating Temperature Range
  - C80C86 ..... 0^{o}C to $+70^{o}$C
  - I80C86 ..... -40^{o}C to $+85^{o}$C
  - M80C86 ..... -55^{o}C to $+125^{o}$C

Si consiglia di fare il "download" dal sito WEB del corso, si veda prima pagina, dei DATA SHEET (dei manuali tecnici operativi) dei componenti.

Tali documenti sono stampabili come il resto delle dispense e dei lucidi

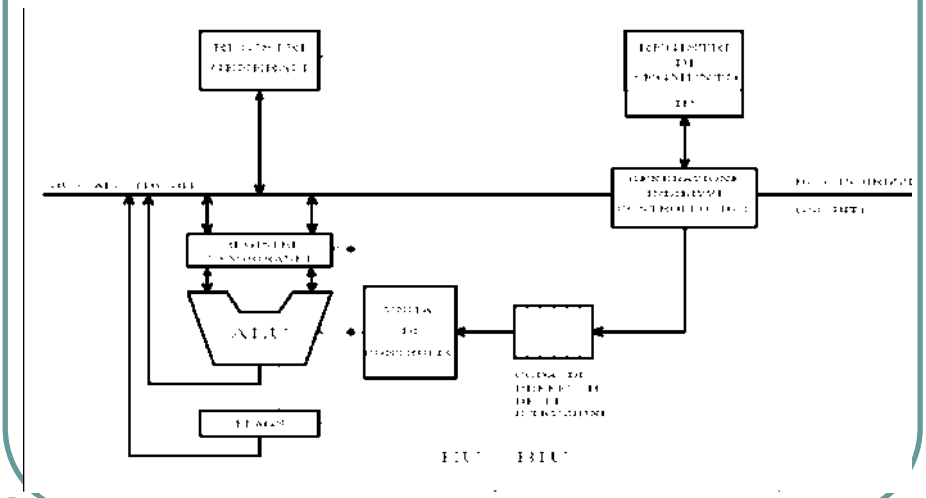
Piedinatura dell'8086



Caratteristiche

- 20 bit BUS address, 20 bit di indirizzamento
  - 1 Mbyte, da 00000H a FFFFFH
- 16 bit BUS dati, 16 bit per il BUS dati
  - Dati e istruzioni, architettura VN
- 16 bit ALU, operazioni algebrico e logiche native a 16
  - Complemento a 2 per interi, virgola mobile, etc.
- BUS Multiplato nel tempo
- Memoria Segmentata
- BUS con gestione del Tri-State
- Due Modalità di funzionamento:
  - modo minimo e modo massimo
  - Gestione semplice ed avanzata del BUS

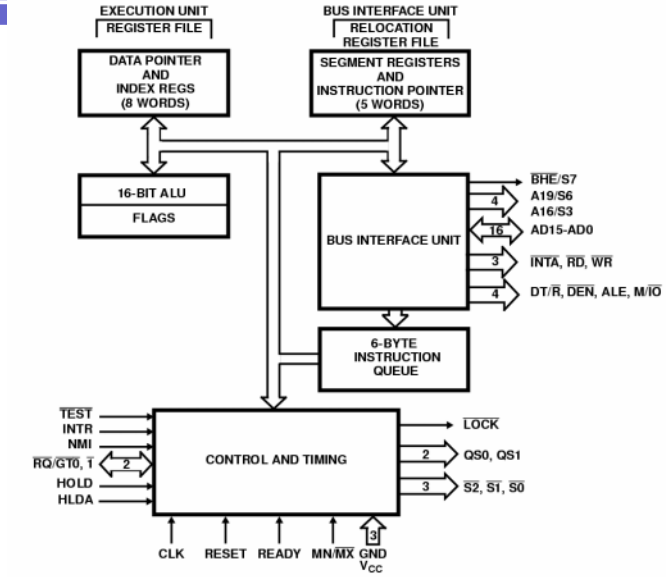
Architettura interna dell'8086



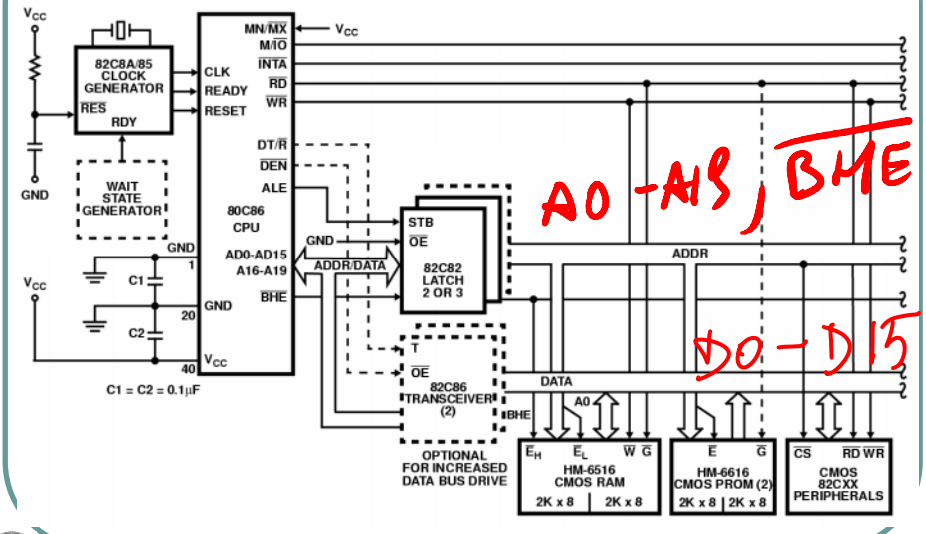
Componenti principali

- EC: Execution Unit
  - Esecuzione delle operazioni, ALU e Registri
- BIU: Bus Interface Unit
  - Accesso alla memoria per istruzioni e dati
  - Gestioni dei registri di indirizzamento
    - Calcolo dell'indirizzo fisico da quello logico di segmento
  - Controllo logico del BUS
  - Acquisizione delle istruzioni ed inserimento di queste in una coda

Diagramma Funzionale



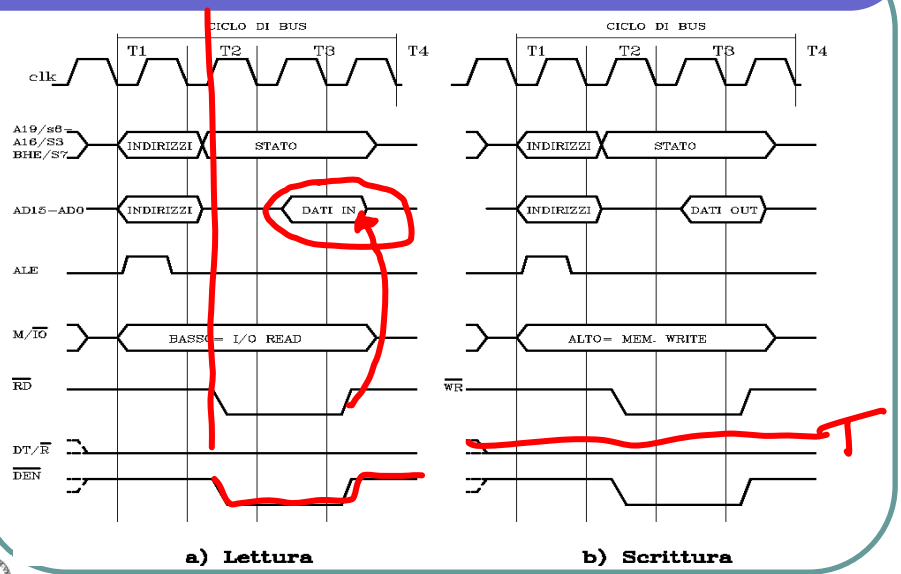
Modo Minimo dell'8086



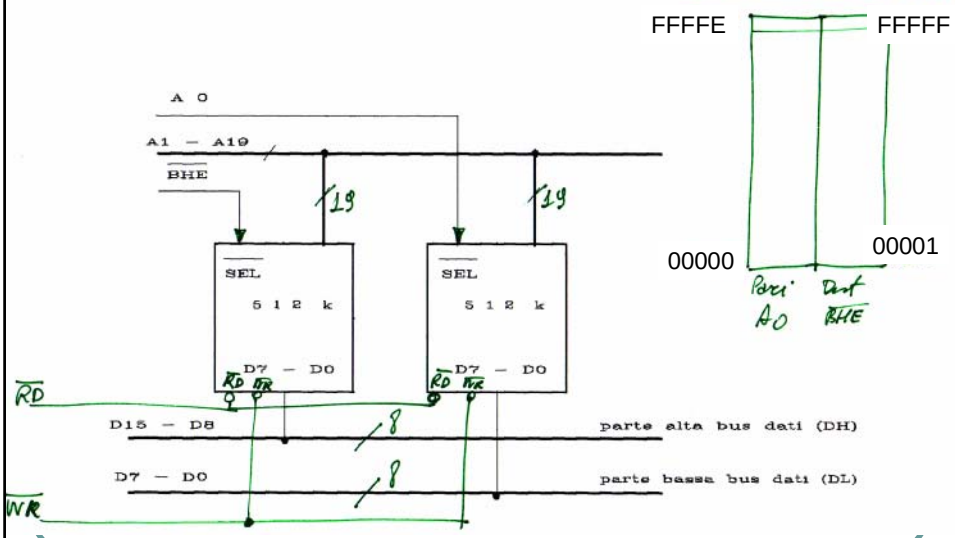
BUS Multiplato, modo minimo

- 16 bit vengono presentati nella prima parte del ciclo macchina sui pin ADxx
- Sugli stessi pin si ha il BUS dati nella seconda parte del ciclo macchina
- I bit di indirizzo da 17 a 20 sono presentati su pin diversi che nella prima parte del ciclo macchina presentano tali indirizzi, mentre nella seconda lo stato della CPU

Ciclo Macchina, Accesso alla Memoria



Accesso a BUS con BHE

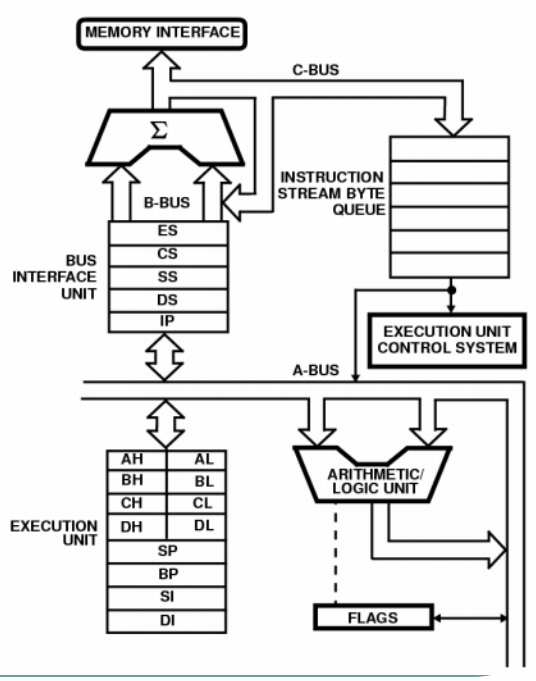


Tipo di Lettura sul BUS

- Not BHE, BUS High Enable, abilitazione della parte alta del bus, segnale attivo basso
- Letture di Byte e Word da indirizzi pari e dispari ?

not BHE	A0	dati	start	#cicli
1	0	byte	pari	1
0	1	byte	dispari	1
0	0	word	pari	1
1	1	--	--	--

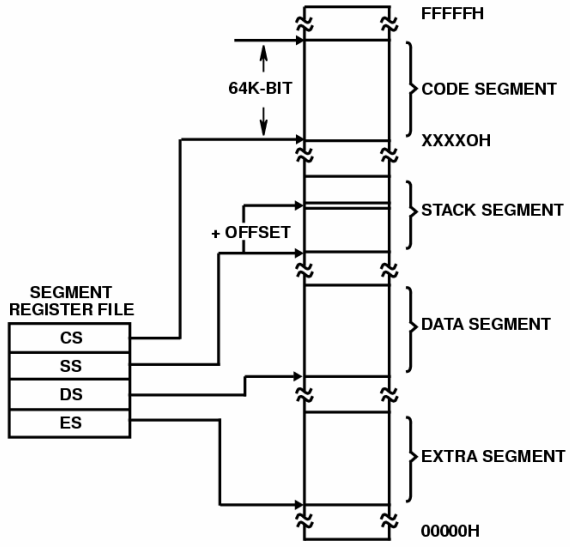
Registri Interni



Registri della BIU,

- **IP: Instruction Pointer (offset)**
  - Non e' semplicemente l'offset dell'istruzione successiva poiche' ho la coda di istruzioni.
- **CS: Code Segment**
  - Segmento che contiene codice, il programma, corrente o meno
- **SS: Stack Segment**
  - Segmento che contiene dati temporanei. In tale segmento viene salvato lo stato della CPU e del programma quando si hanno dei cambi di contesto per ripristinarlo quando serve, se ne parlerà piu' in dettaglio in seguito
- **DS: Data Segment**
  - Segmento che contiene dati per l'elaborazione
- **ES: Extra Segment**
  - Segmento aggiuntivo per i dati
- Tutti questi registri sono a 16 bit mentre la memoria e' a 20 bit
- Servono per comporre il valore dell'indirizzo fisico
- Si hanno indirizzi logici a 16 bit e fisici a 20 bit.

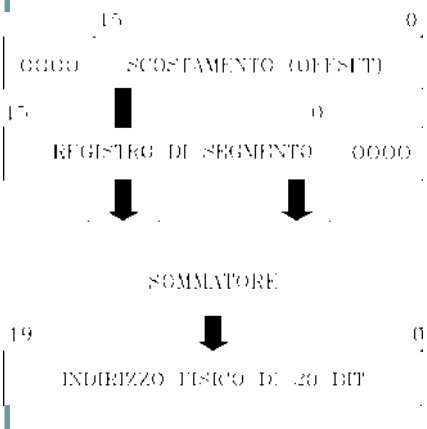
Memoria Segmentata



Memoria Segmentata

- La memoria e' di 1Mbyte, 20 bit
- Indirizzi logici sono espressi in termini di Segment:offset
- Segment e Offset sono parole di 16bit, 1 Word
- Questo permette di muovere i vari segmenti all'interno della memoria, fare rilocazione
- Anche di sovrapporre i segmenti
- L'indirizzo fisico e' ottenuto da quello logico:
  - Indirizzo fisico = segment * 10H + offset
- I 16 bit dei numeri sono memorizzati in memoria in ordine inverso.

Calcolo dell'indirizzo fisico

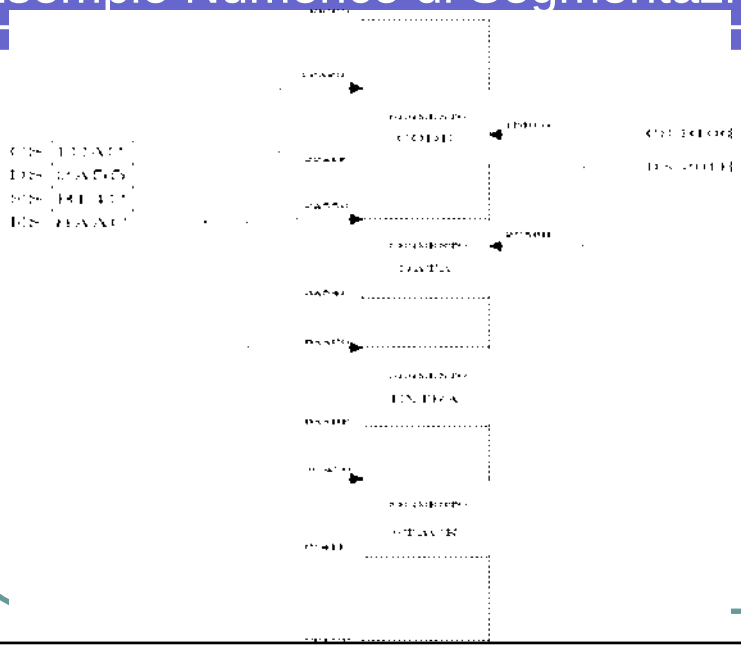


- ADD AX, VAR
  - AX ←AX+M[DS:offset(VAR)]
- ADD AX,ES:ALT
  - AX←AX+M[ES:ALT]
- Ogni programma deve dichiarare quali e quanti segmenti utilizza

Registro Accumulatore, AX

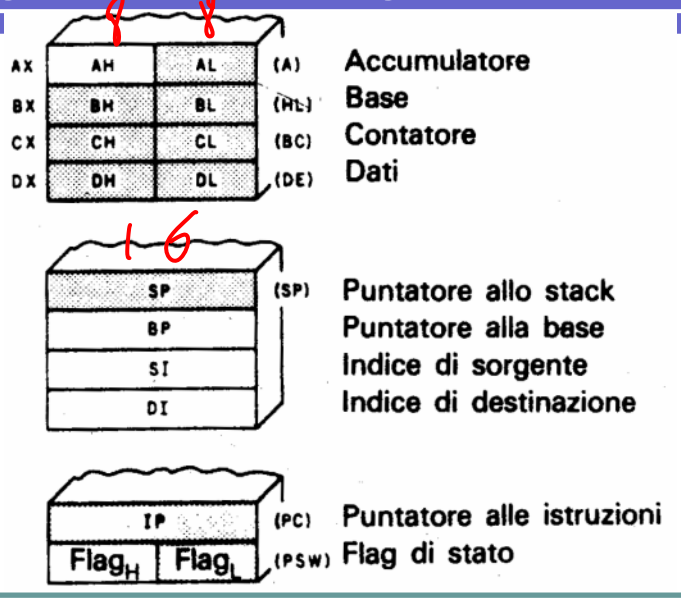
- AX viene detto anche registro accumulatore
- E' un registro privilegiato per certe istruzioni
- Per esempio ADD AX,VAR effettua una somma del valore i VAR e lo somma al valore di AX mettendo, accumulando il risultato in AX stesso:
  - AX ← AX + M[DS:VAR]

Esempio Numerico di Segmentazione



329

Registri di utilizzo generale



Paolo Nesi, Univ. Firenze, Italy, 2003-06

330

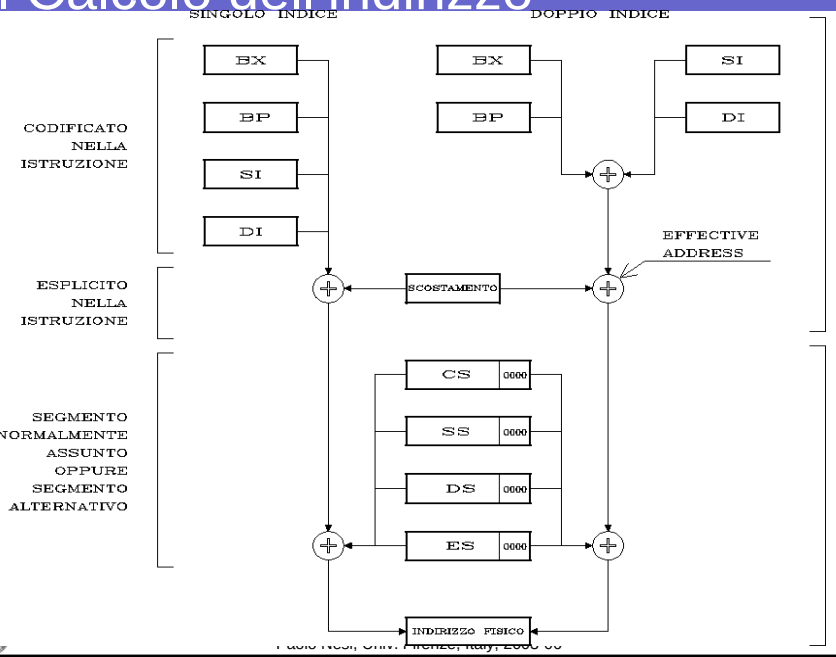
Registri

- **AH-AL, BH-BL, CH-CL, DH-DL**
  - 8 registri a 8 bit che
  - possono essere visti anche come 4 registri a 16 bit: AX, BX, CX, DX
- **SP: Stack Pointer, 16 bit**
  - offset per indirizzare alla testa dello Stack dentro il SS
- **BP: Base Pointer, 16 bit**
  - offset per indirizzare all'interno dello Stack la sua posizione di riferimento
- **SI: Source Index, 16 bit**
  - Indice che serve per alcune istruzioni per indicare la posizione relativa della sorgente dei dati
- **DI: Destination Index, 16 bit**
  - Indice che serve per alcune istruzioni per indicare la posizione relativa di destinazione dei dati

Coinvolgimento dei Registri di indiriz.

TYPE OF MEMORY REFERENCE	DEFAULT SEGMENT BASE	ALTERNATE SEGMENT BASE	OFFSET
Instruction Fetch	CS	None	IP
Stack Operation	SS	None	SP
Variable (except following)	DS	CS, ES, SS	Effective Address
String Source	DS	CS, ES, SS	SI
String Destination	ES	None	DI
BP Used As Base Register	SS	CS, DS, ES	Effective Address

Il Calcolo dell'indirizzo



Esempi

- [CS: BX + SI + Var]
- [DS: BX + VAR]
- [SS: BP + FEH]

- La CPU non ha nessuna conoscenza del tipo di dato. I dati vengono interpretati dalle istruzioni per quello che sono: cioè numeri binari.
- Se l'istruzione fa l'ipotesi di lavorare su numeri complemento a 2 e I valori dati non lo sono, ovviamente la CPU non se ne puo' accorgere.

Registri Interni di stato

- **IP: Instruction Pointer, 16 bit**
  - E' sempre un Offset per un indirizzamento logico
- **PSW: program status word**,
  - 16 bit, non tutti utilizzati, solo flag da un bit
  - Detto anche: Status Register o Flag register
  - Composto da una parte alta e bassa
  - I flag sono utilizzati dalle istruzioni
    - per tenere conto del contesto, del risultato delle istruzioni precedenti
    - Per modificare lo stato per le istruzioni successive, passare un riporto, condizionare una scelta, etc.

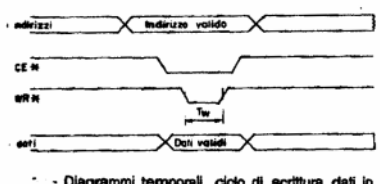
PSW, Flag Register

- **ZF (Zero Flag) -- Flag di zero (zero/nonzero).**
  - posto ad 1 quanto il risultato di una operazione aritmetica e' zero;
- **SF (Sign Flag) -- Flag di segno** (negativo/positivo).
  - posto a 1 quando il risultato di una operazione sulla ALU e' un numero negativo
- **CF (Carry Flag)** -- Flag di presenza di riporto (carry) (si/no)
  - posto ad 1 quando si ha un riporto nelle operazioni di somma o sottrazione sia a 8 che a 16 bit;
- **PF (Parity Flag)** -- Flag di presenza di parita'.
  - posto ad 1 se la parte bassa (8 bit) del risultato di una operazione dell'ALU contiene un numero pari di 1;
- **AF (Auxiliary Flag)** -- Flag ausiliario (si/no)
  - posto ad 1 in presenza di riporto nelle operazioni in codifica BCD;

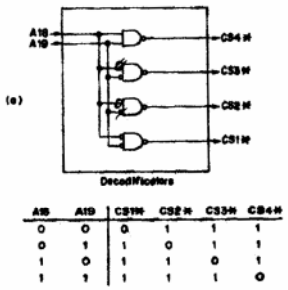
PSW, Flag Register

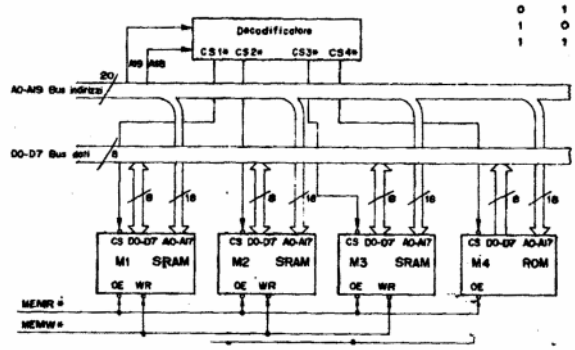
- **DF (Direction Flag)** -- Flag di direzione (decrescente/crescente),
  - assume significato solo nelle operazioni con stringhe dove identifica l'organizzazione di queste in memoria;
- **IF (Interrupt Flag) --** Flag che segnala le condizioni della CPU rispetto alle interruzioni (abilitate/disabilitate)
  - se e' ad 1 il programma in esecuzione puo' essere interrotto;
- **OF (Overflow Flag)** -- Flag di presenza di overflow (si/no)
  - il quale assume significato solo in presenza di operazioni aritmetiche
- **TF (Trap Flag)** -- Flag di presenza di trap (si/no),
  - quando e' ad 1 il programma può essere eseguito passo passo (single-step). utilizzato per il controllo dell'esecuzione del programma (debug).

Esempio di Espansione della Memoria



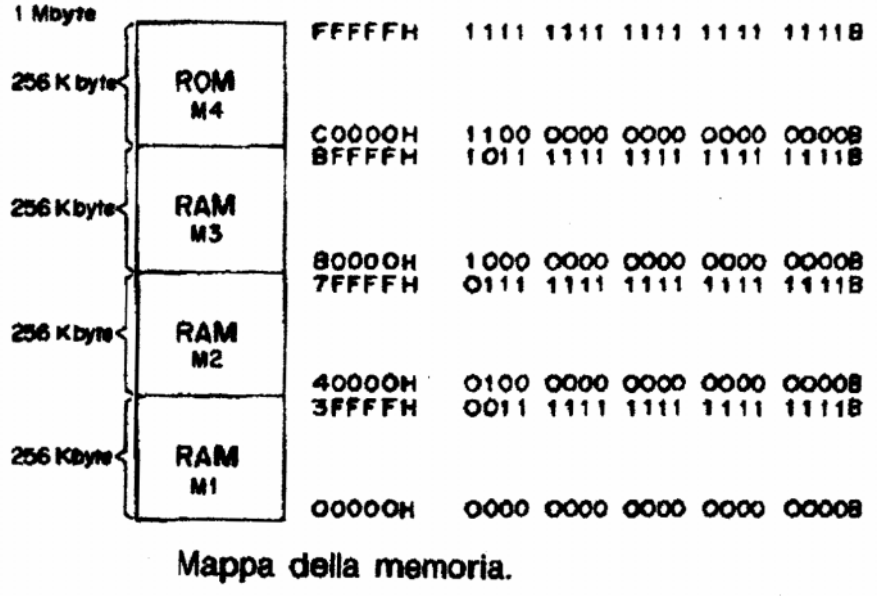
- Diagrammi temporali, ciclo di scrittura dati in moria.





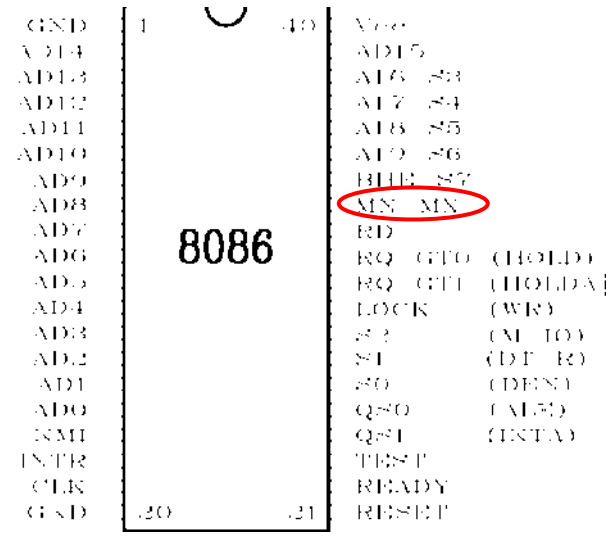
- Definizione di un banco di memoria composto da 4 quattro memorie da 256 Kbyte. Collegamento con il BUS stema.

Mappa della memoria dell'esempio

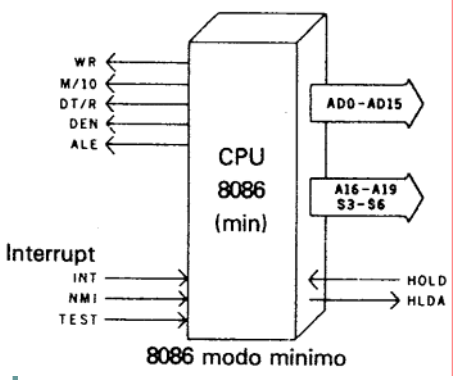


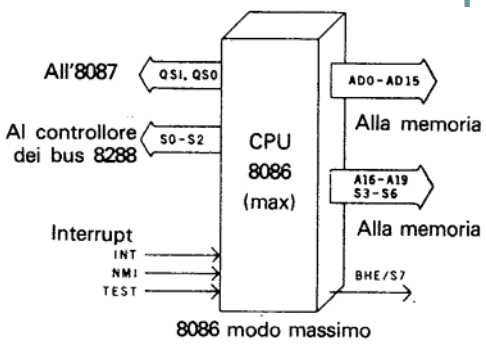
Mappa della memoria.

Piedinatura dell'8086

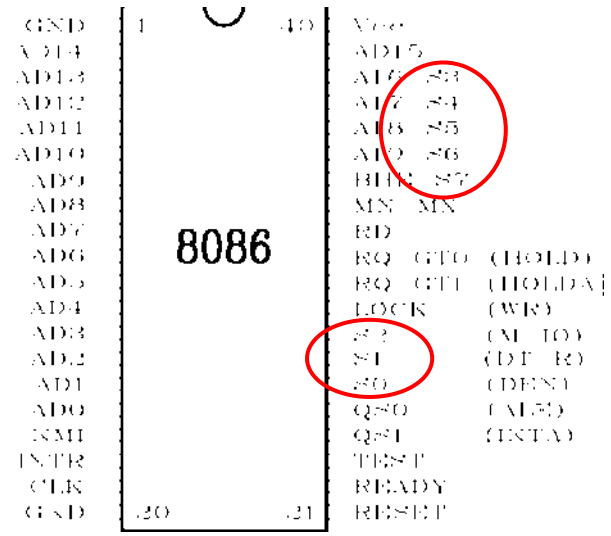


Modo Minimo e Massimo





Piedinatura dell'8086



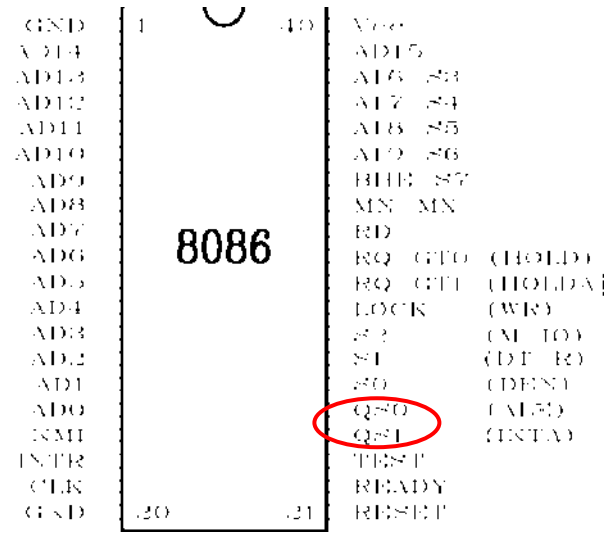
Lettura dei Bit di stato

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	CHARACTERISTICS
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

S4	S3	CHARACTERISTICS
0	0	Alternate Data
0	1	Stack
1	0	Code or None
1	1	Data

- Leggibili durante il ciclo macchina
- Letti da certe periferiche per la loro sincronizzazione
- Modo massimo
- Riconoscimento del tipo di ciclo macchina

Piedinatura dell'8086

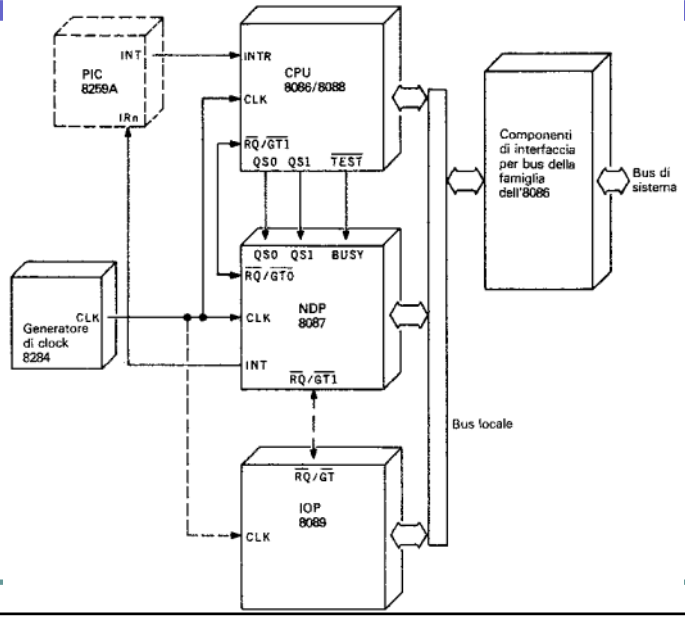


Lo stato della Coda

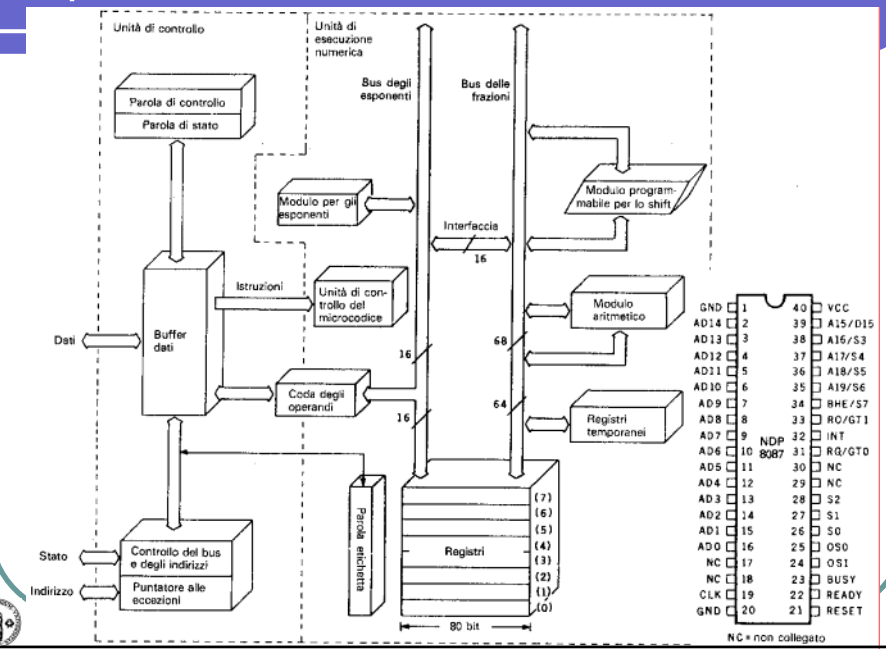
QSI	QSO	
0	0	No Operation
0	1	First byte of op code from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

- Leggibili durante il ciclo macchina
- Letti da certe periferiche per la loro sincronizzazione
- Riconoscimento del tipo di ciclo macchina

8086 Massimo & 8087



Coprocessore Matematico 8087



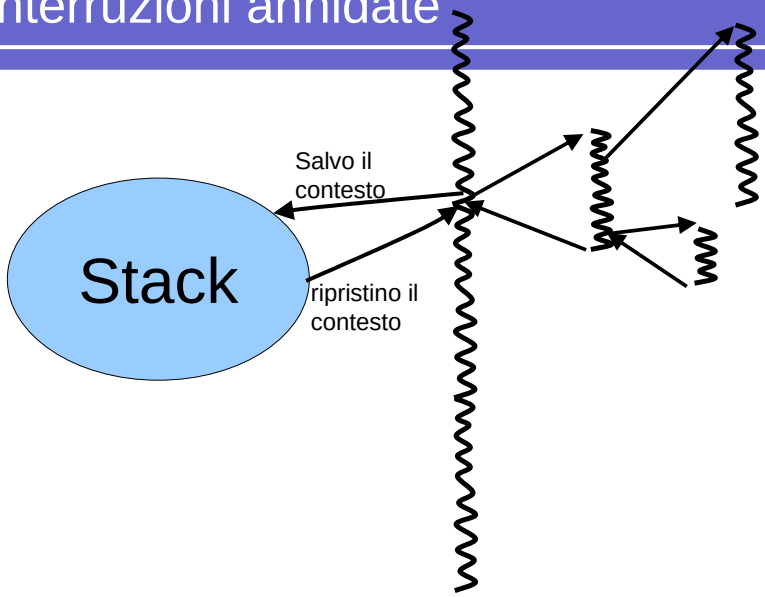
Le Interruzioni

- Le interruzioni sono situazioni eccezionali che portano la CPU ad interrompere il normale svolgimento previsto del programma.
- La CPU salva **il contesto** e passa all'esecuzione di un altro spezzone di codice
- La CPU lavora sempre: cursore, movimento del mouse, icone animate, acceso al disco, etc.
- Interrupt: tastiera, mouse, stampante, disco, timer, etc.

Le Interruzioni possono essere

- Le Interruzioni Hardware: se provocate da un periferica e quindi sono scaturite da un segnale fisico
- Software: codificate in chiaro nel codice per esempio INT 21
- trap o eccezioni: sono prodotte dall'interno della CPU da situazioni di errore (divide per zero, stack overflow, etc.) o di gestione ad elevata priorità

Interruzioni annidate



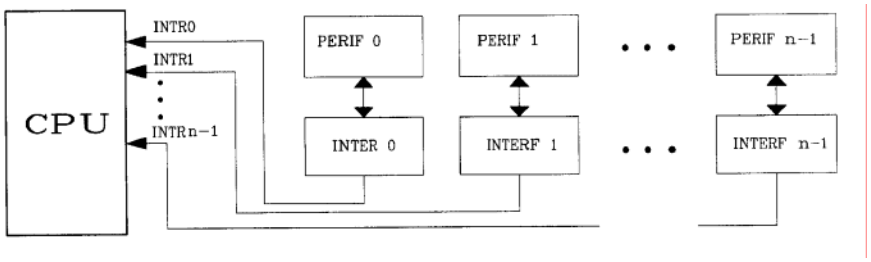
Le Interruzioni

- **La CPU deve prevedere l'arrivo di interruzioni predisponendo le routine/procedure che devono essere eseguite quando il contesto cambia.**
  - Al completamento della routine, l'esecuzione ritorna al programma che era stato interrotto
  - E' possibile interrompere anche le routine stesse con altre interruzioni. Questo può essere evitato con opportune istruzioni che abilitano e disabilitano la sensibilità alle interruzioni della CPU.
  - Il cambio di contesto scatena il salvataggio di questo nello Stack.
- **Le interruzioni possono essere disabilitate**
  - Sui pin della CPU di solito esiste anche un pin di Interruzione non disabilitabile, NMI, Non Maskerable Interrupt

Le Interruzioni

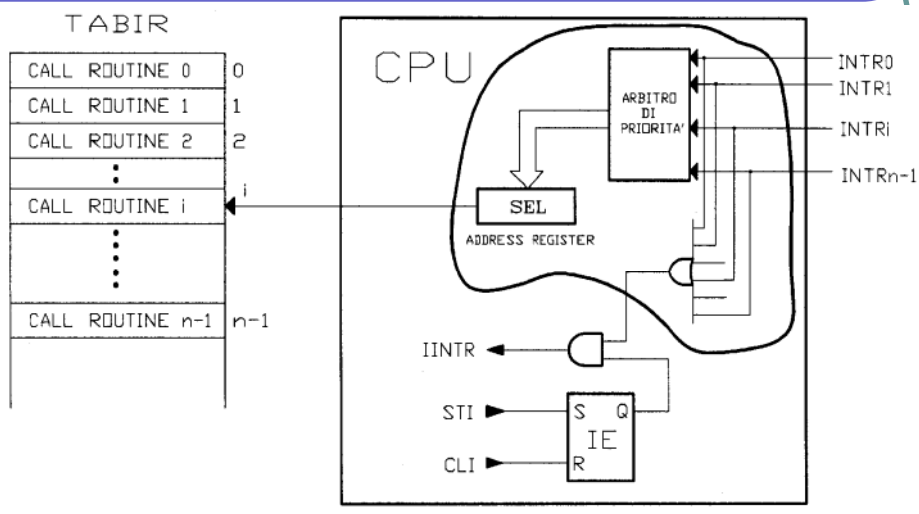
- **La CPU tipicamente può ricevere diverse interruzioni**
- Alcune interne altre da periferiche esterne, quelle esterne sono definibili dal progettista e anche in parte dall'utente
- alcune CPU hanno dei pin specifici per le interruzioni, tanti pin quante interruzioni possibili esterne (Interruzioni Indipendenti)
- Altre CPU hanno solo un pin e posseggono un meccanismo per riconoscere quale periferica delle tante ha scatenato tale interruzione. In questo caso, l'informazione che arriva alla CPU deve indicare quale procedura di servizio deve essere attivata.

Interruzioni a linee indipendenti



- Piuttosto complesso poiché la Cpu deve avere linee e pin indipendenti
- La priorità viene programmata internamente alla CPU o dalla posizione

Interruzioni a Linee indipendenti



Servizio delle Interruzioni

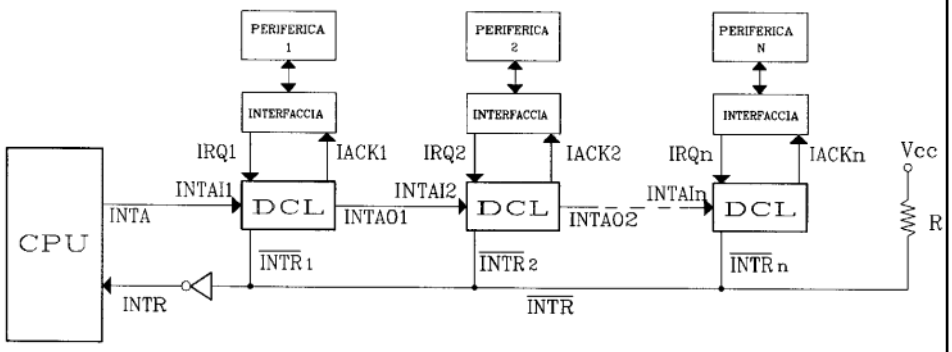
- E' necessario identificare la richiesta da servire, questa può essere diretta oppure decodificata se l'interruzione e' vettorizzata
- Nel caso in cui la CPU non riceva indicazioni dirette della periferica che ha effettuato la richiesta vi sono due metodi per arrivare ad identificarla:
  - Polling (la CPU richiede a tutte le periferiche in modo ciclico se sono state loro fino a che non identifica quella che ha provocato l'interruzione)
  - Daisy Chain (esiste una catena Hardware che risolve il problema)

Polling delle interruzioni

- Scansione di tutte le periferiche e colloquio con queste una ad una per capire quale di queste ha fatto richiesta
- La scansione (polling) puo' essere eseguita secondo un certo ordine. Tale ordine determina la priorità se la scansione si ferma alla prima periferica che afferma di avere effettuato richiesta.
- in tale caso, le altre periferiche perdono il servizio se hanno priorità piu' bassa, e devono richiederlo in seguito se vogliono essere servite.

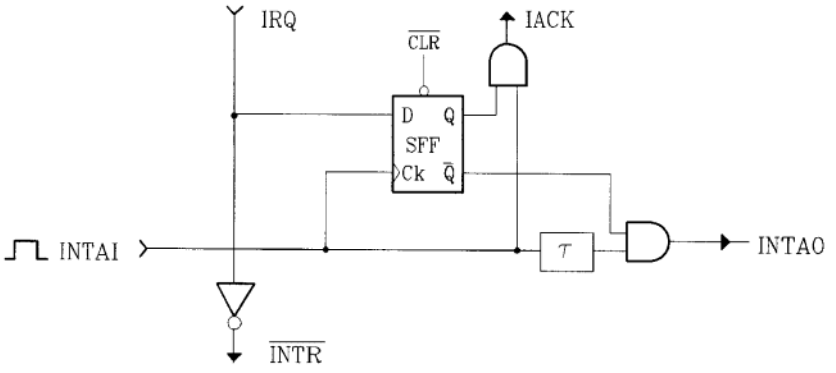
Daisy Chain Asincrona

- Si ha una catena di dispositivi la cui priorità è determinata dalla posizione



- INTR è dato dal *wired* OR di tutte le interfacce
- INTA viene propagato lungo la catena

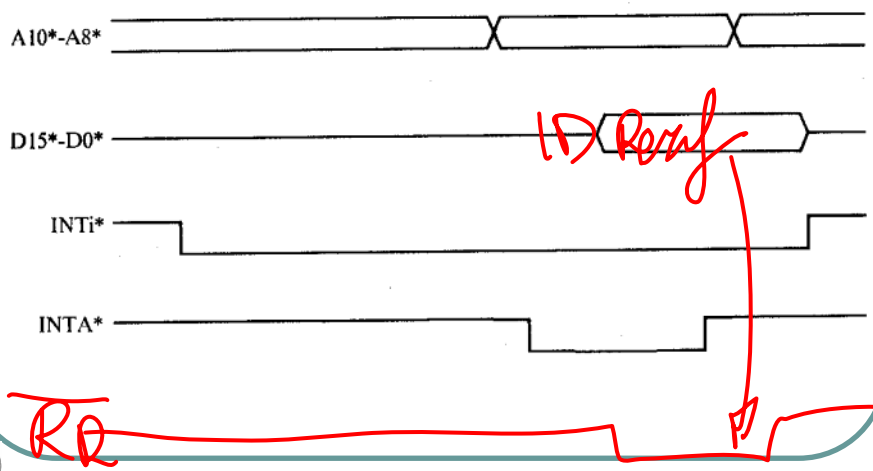
Interfaccia DCL, Daisy Chain Logic



- L'elemento di ritardo impedisce che INTA venga trasferito a valle prima che SFF abbia commutato

Un esempio di ciclo di INTA

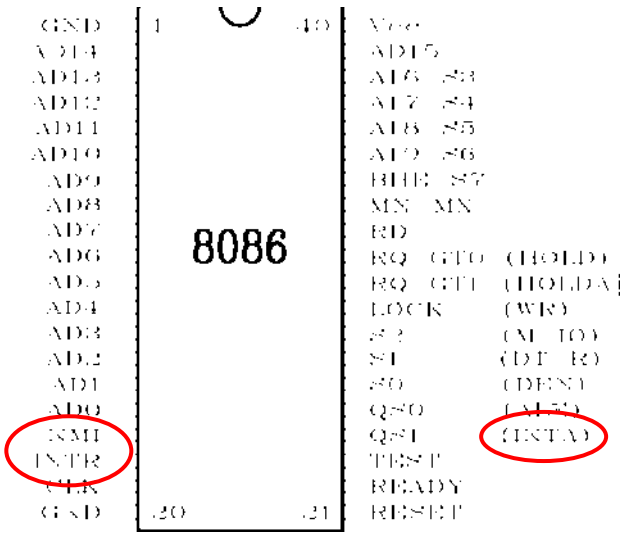
- Non e' un 8088/8086



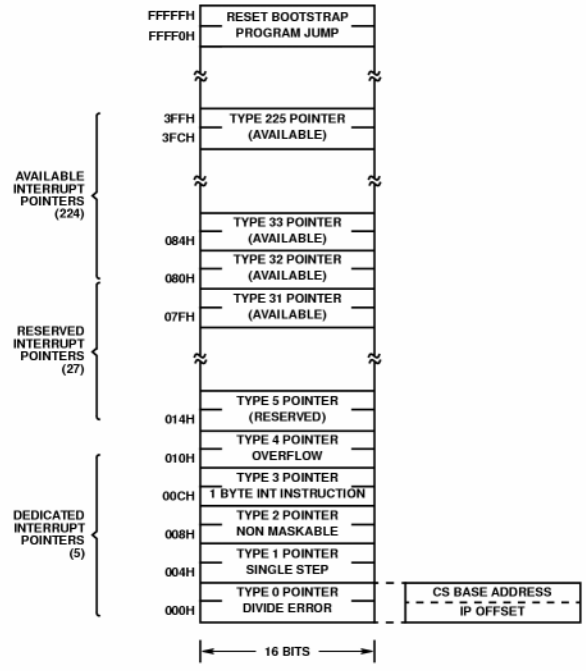
Le Interruzioni e l'8086

- Gestisce le Interruzioni in modo Vettorizzato
- Ha un pin di NMI
- Ha alcuni segnali di controllo per la gestione di un PIC, Programmable Interrupt Controller
- L'8086 può ricevere 256 diverse interruzioni e pertanto gestisce nei primi 1024 byte della memoria la Vector Table.
- Un tabella in cui sono presenti gli indirizzi della memori di CODIC delle routine che devono essere eseguite a fronte delle interruzioni.
- Tali indirizzi sono nelle forma, Offset e segmento, pertanto ogni indirizzo e' composto da due word a 16 bit

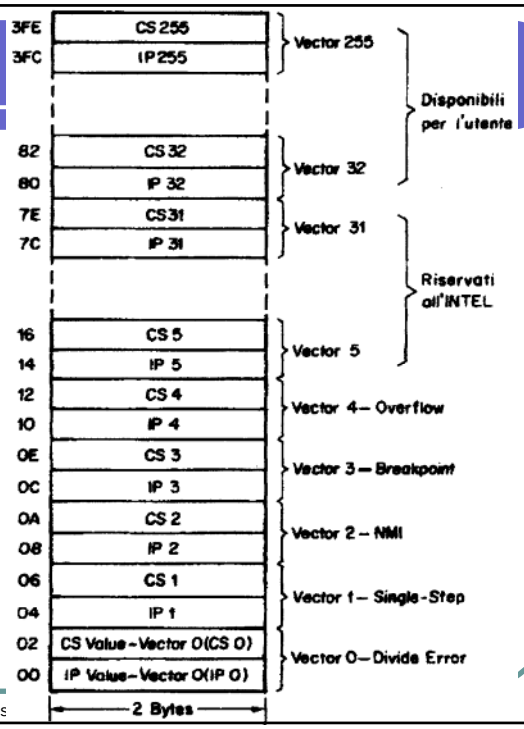
Piedinatura dell'8086



Vettore delle Interruzioni dell'8086



Vettore delle Interruzioni 8086

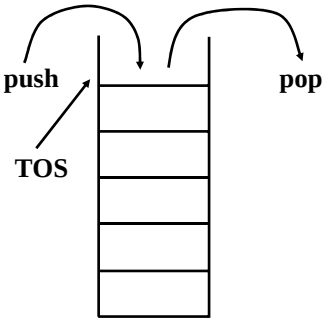


La gestione dello Stack

- La pila o *stack* e' un struttura lineare dove le operazioni di inserimento ed estrazione dei dati avvengono sempre dalla stessa parte. La pila e' un struttura che e' governata da una politica di tipo FILO (*first input last output*) il primo a entrare e' l'ultimo a uscire, oppure equivalentemente LIFO (*last input first output*), l'ultimo a entrare e' il primo ad uscire.
- Si prenda ad esempio un tubetto di pasticche di vitamina. Per togliere l'ultima (la prima pasticca che e' stata inserita) e' necessario togliere tutte le pasticche presenti.
- L'operazione di inserimento viene detta di *push* mentre l'operazione di estrazione viene detta di *pop*. Per la sua gestione e' necessario conoscere solamente il punto in cui viene fatta l'inserzione oppure l'estrazione. Tale punto e' detto *Top of the Stack*, TOS.
- Lo *stack* e' tipicamente una struttura dinamica perché le sue dimensioni possono non essere note al momento della realizzazione.

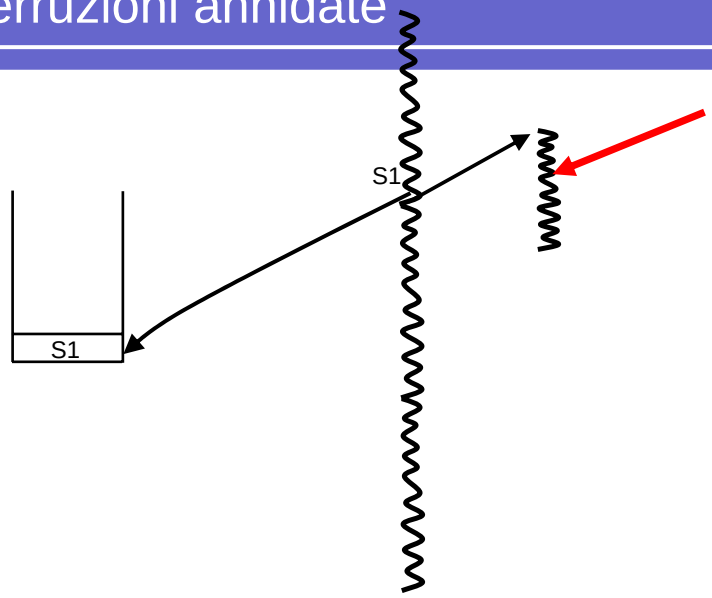
Lo Stack

- Poiché tutte le operazioni fanno riferimento ad uno stato iniziale dello *stack*, stato definito alla sua realizzazione, e' necessario tenere sotto controllo quando lo *stack* si svuota.
- Durante la gestione dello *stack* può altrimenti accadere che si cerchi di effettuare un'operazione di pop senza che vi sia un elemento da estrarre dallo *stack*.
- Pertanto le operazioni di base sono la creazione, l'inserimento (pop), l'estrazione (*push*) e la verifica di *stack* vuoto prima di fare ogni *pop*.

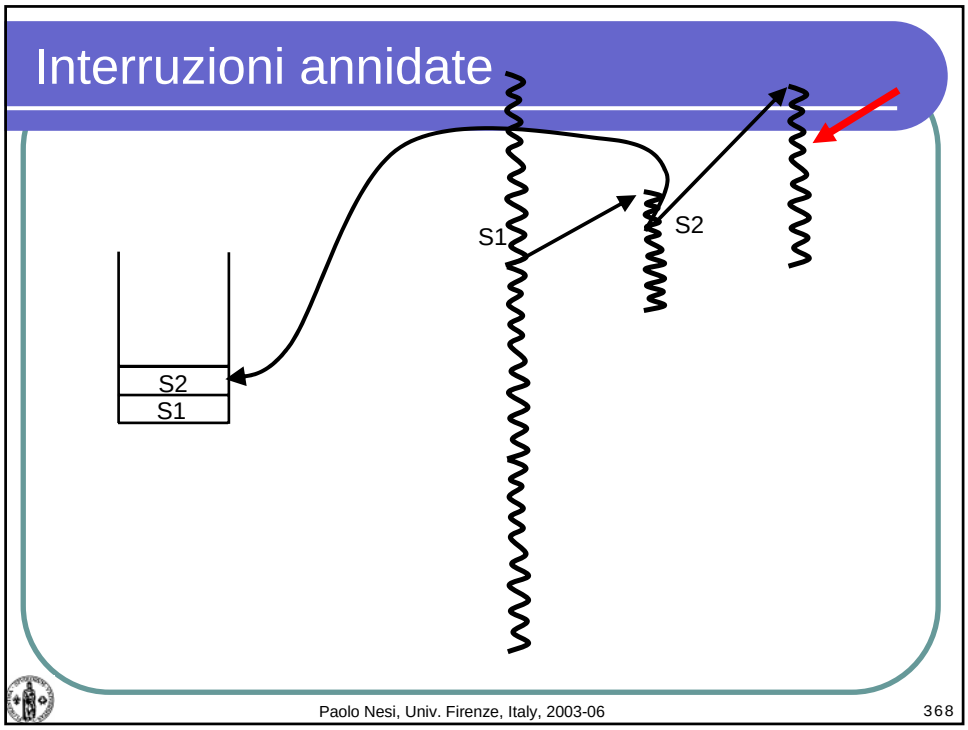


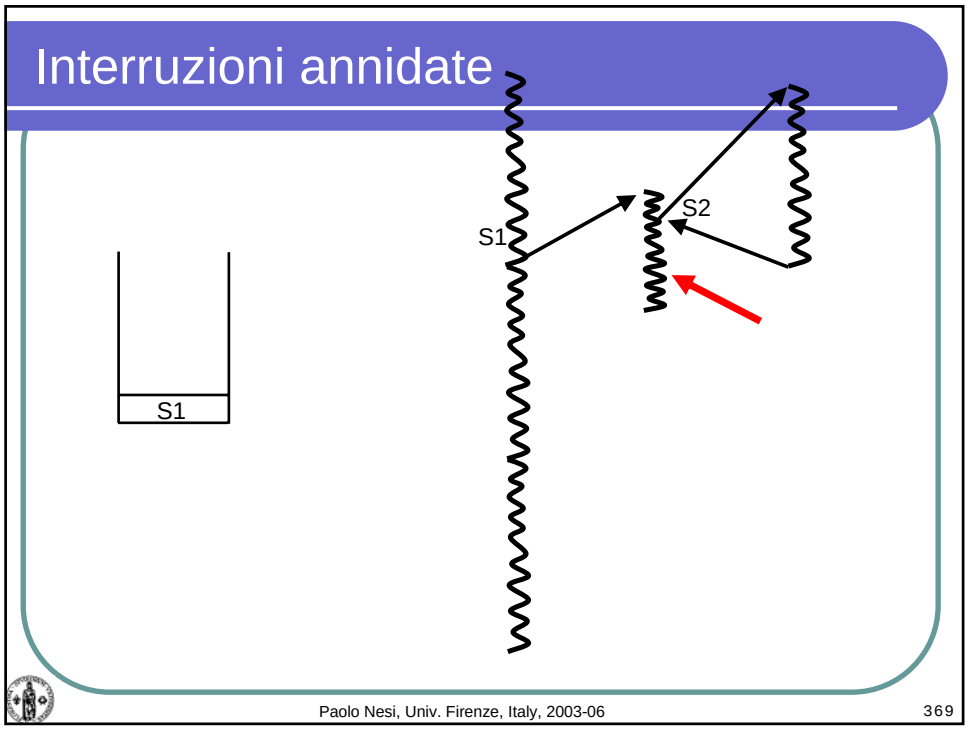
Paolo Nesi, Univ. Firenze, Italy, 2003-06 366

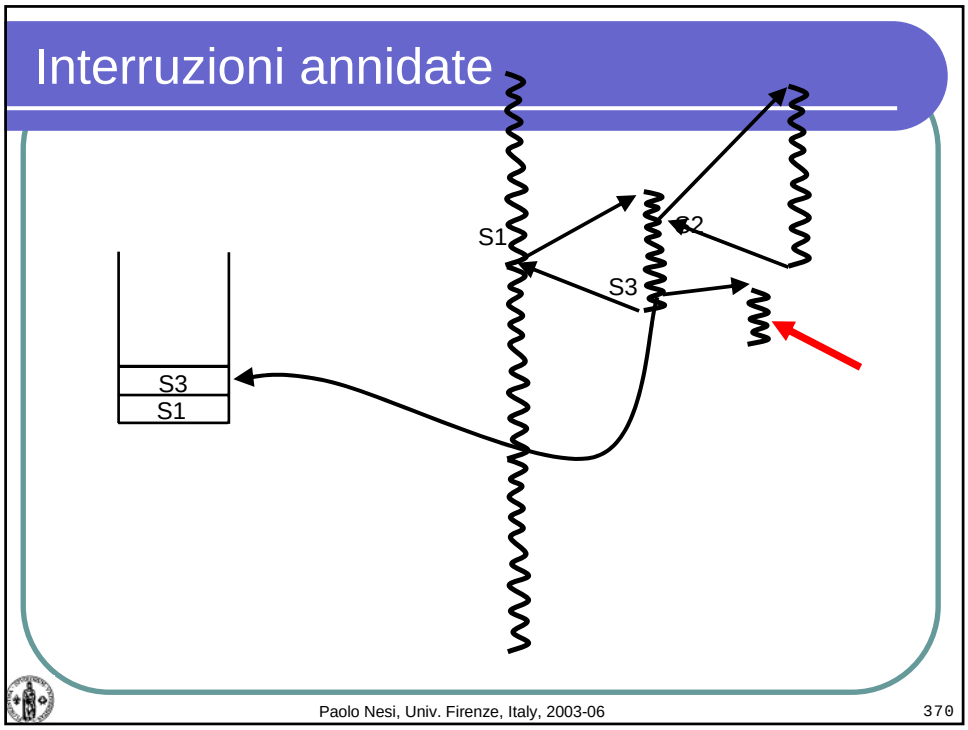
Interruzioni annidate

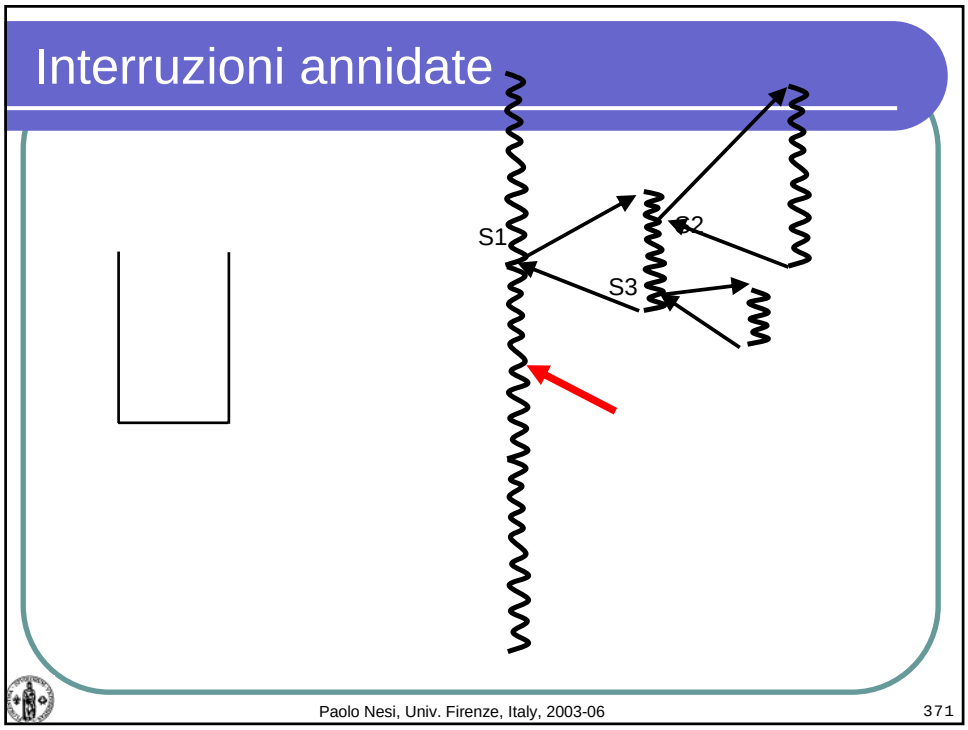


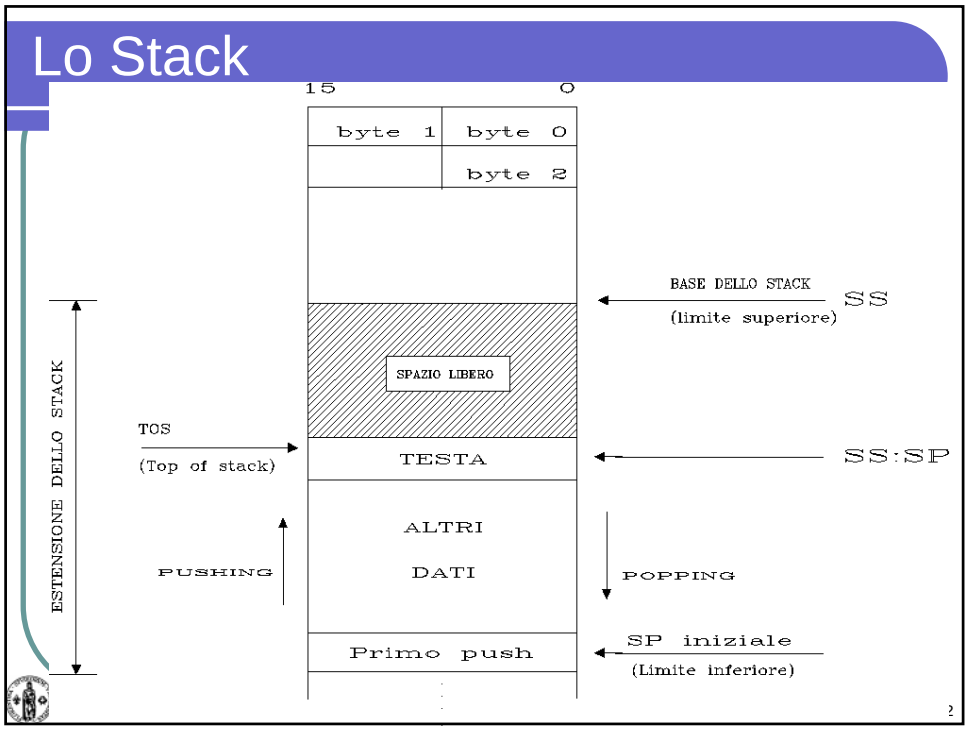
Paolo Nesi, Univ. Firenze, Italy, 2003-06 367

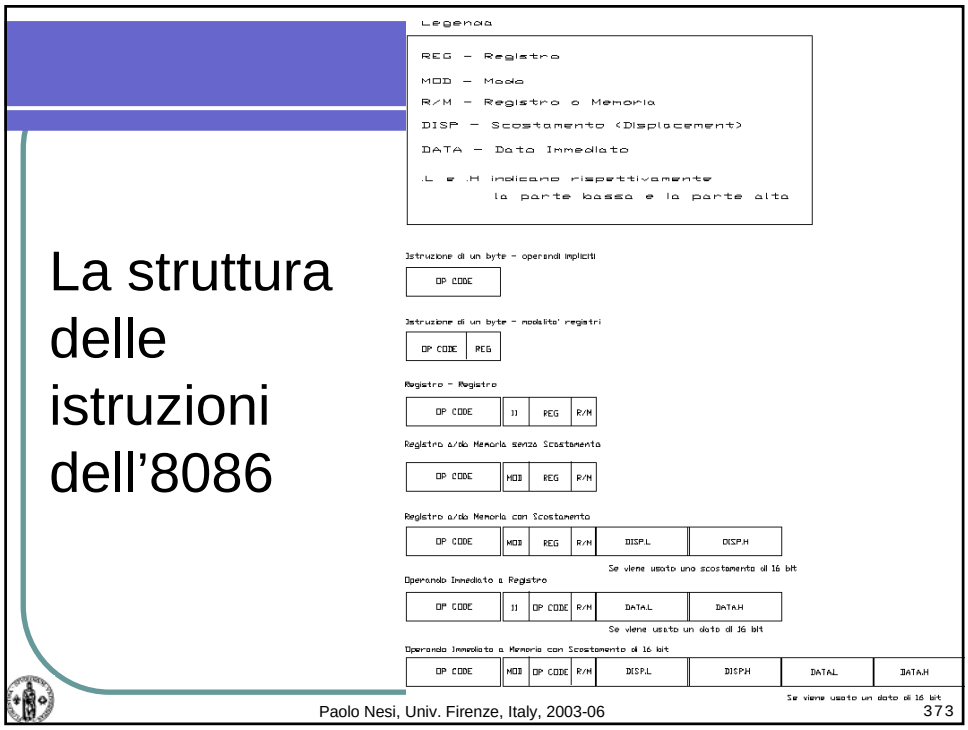

Interruzioni annidate
S1
S2
S2
S1
Paolo Nesi, Univ. Firenze, Italy, 2003-06
368


Interruzioni annidate
S1
S2
S1
Paolo Nesi, Univ. Firenze, Italy, 2003-06
369


Interruzioni annidate
S1
S2
S3
S3
S1
Paolo Nesi, Univ. Firenze, Italy, 2003-06
370


Interruzioni annidate
S1
S2
S3
Paolo Nesi, Univ. Firenze, Italy, 2003-06
371


Lo Stack
15
0
byte 1
byte 0
byte 2
BASE DELLO STACK
(limite superiore)
SS
SPAZIO LIBERO
ESTENSIONE DELLO STACK
TOS
(Top of stack)
TESTA
SS:SP
ALTRI
DATI
PUSHING
POPPING
Primo push
SP iniziale
(Limite inferiore)

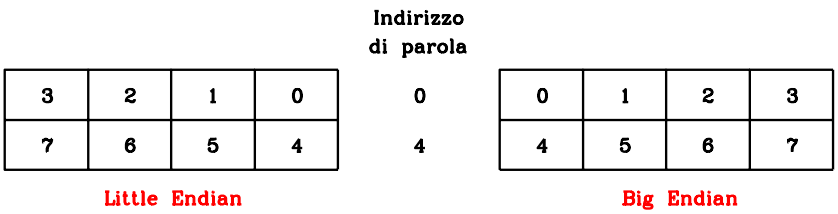

La struttura delle istruzioni dell'8086
Legenda
REG – Registro
MOD – Modo
R/M – Registro o Memoria
DISP – Scostamento (Displacement)
DATA – Data Immediato
.L e .H indicano rispettivamente la parte bassa e la parte alta
Istruzione di un byte – operandi impliciti
OP CODE
Istruzione di un byte – modalita' registri
OP CODE
REG
Registro – Registro
OP CODE
11
REG
R/M
Registro a/da Memoria senza Scostamento
OP CODE
MOD
REG
R/M
Registro a/da Memoria con Scostamento
OP CODE
MOD
REG
R/M
DISP.L
DISP.H
Se viene usato uno scostamento di 16 bit
Operando Immediato a Registro
OP CODE
11
OP CODE
R/M
DATA.L
DATA.H
Se viene usato un dato di 16 bit
Operando Immediato a Memoria con Scostamento di 16 bit
OP CODE
MOD
OP CODE
R/M
DISP.L
DISP.H
DATA.L
DATA.H
Se viene usato un dato di 16 bit
Paolo Nesi, Univ. Firenze, Italy, 2003-06
373

Si Ricorda che

- Si hanno
  - Dimensioni delle istruzioni con Numero di byte diverso, pertanto si ha un fetch dell'istruzione su piu' cicli macchina
- Opcode delle istruzioni è
  - a dimensione praticamente costante

Ordinamento

- Intel: Little Endian
- Motorola: Big Endian
- PowerPC: a scelta



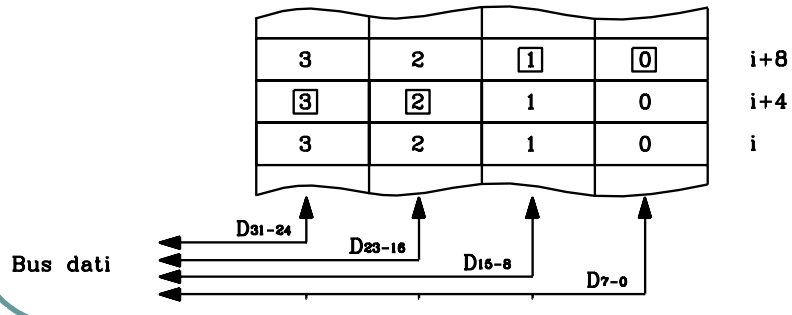
Effetti del non allineamento

- Ipotesi di CPU con 32 bit di indirizzo, 32 bit di data bus
- Dati a 32 bit (un ciclo macchina per leggere 4 byte)
- Pertanto allineamento a 32 bit, 4 byte
- Indirizzi Allineati a 4 byte, pertanto 30 bit effettivi

- Se I dati non sono allineati a indirizzi multipli di 4 byte, per effettuare la lettura della word a 32 si devono fare due cicli macchina

Allineamento in memoria

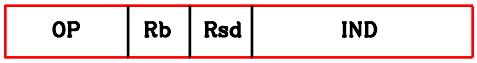
- Esempio: parole di 32 bit, formate da quattro banchi di 8 byte
- La parola tratteggiata è non allineata; ha il byte meno significativo in i+6 (Little Endian) il più significativo in i+9
- Occorrerebbero due accessi alla memoria



Indirizzamento

- L'interpretazione del campo IND può essere differente da macchina a macchina
- Indirizzo effettivo (EA): Il valore che risulta dal calcolo dell'indirizzo attraverso i componenti espliciti contenuti nell'istruzione

LD RA, VAR
MOV AX, VAR



ST VET(R28), R12
MOV VET(IS), BX

Modalità di indirizzamento (dati)

Indirizzamento diretto

LD R1, var ; EA= IND R1:= M[EA]

Indirizzamento relativo ai registri

ST var(R3),R6 ; EA= IND + R3 M[EA]:= R6

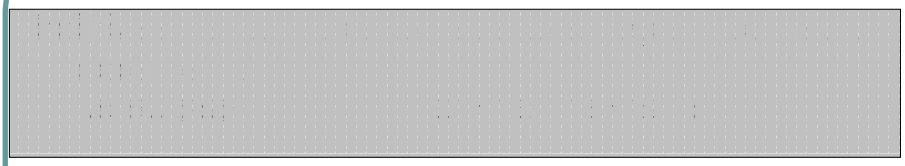
Indirizzamento indiretto rispetto ai registri

LD R1, (R2) ; EA= R2
Mov AH, [R2]

Indirizzamento relativo ai registri indiciato e scalato

LD R1, var (R2) (Rx) ; EA= IND + R2 + RX*d
d è la dimensione dell'elemento

Modalità di indirizzamento (dati)



Indirizzamento immediato

```
LD  R1, [2346]      ; R1:= M[2346]
LD  R1, 2346        ; R1:= 2346   // questa e' una copia
```

Indirizzamento tra registri

```
LD  R16,R8          ; R16:= R8
```

Indirizzamento porte di I/O

```
IN  R5,Porta        ; R5:= porta (di ingresso)
```

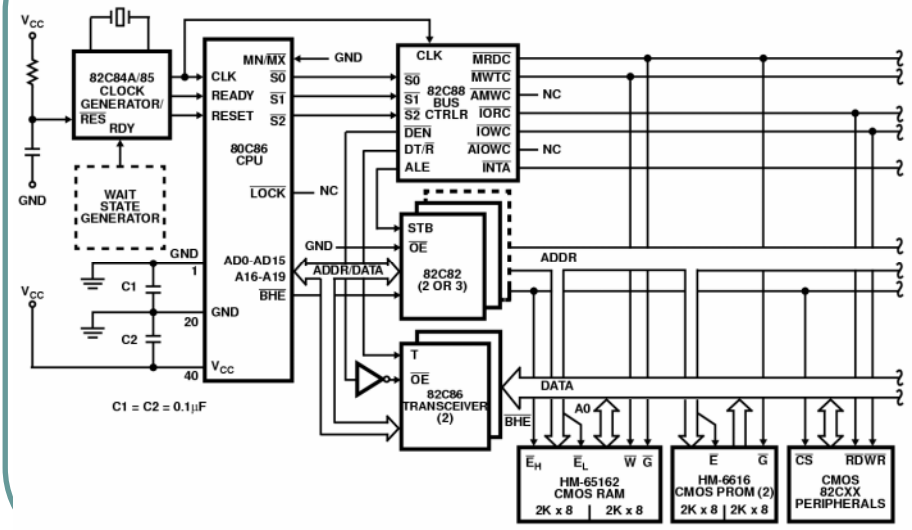
Modalità di indirizzamento (controllo)

- Salto, salto condizionato, chiamata e ritorno da sottoprogrammi
  - Diretto
  - Relativo al PC o ad altro registro
- Esempi

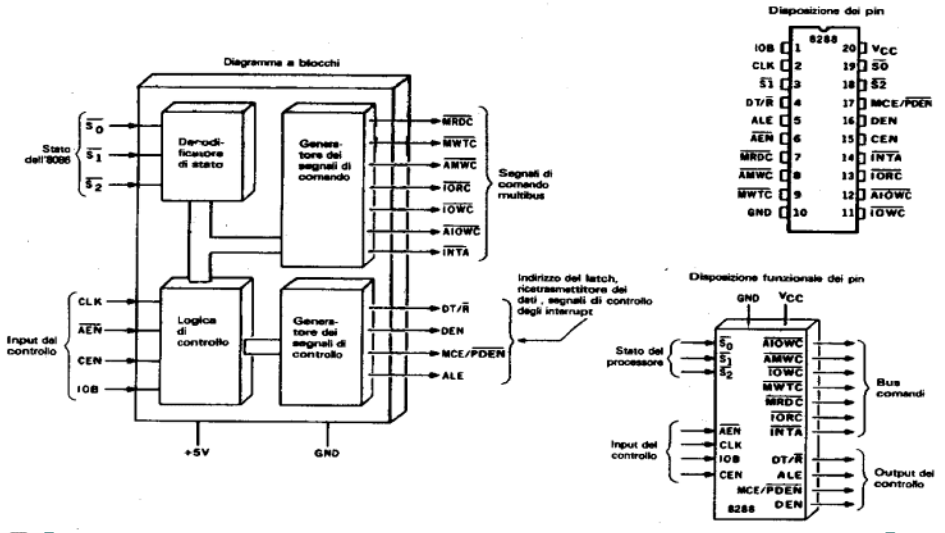
```
JMP     DEST        ; Diretto o relativo a PC
JZ      wait        ; Di solito relativo a PC
call    sub         ; Di solito diretto
BR      R30         ; EA destinazione = R30
INT 21              ; interruzione
```

In alcuni casi la CPU salva lo stato automaticamente in altri no

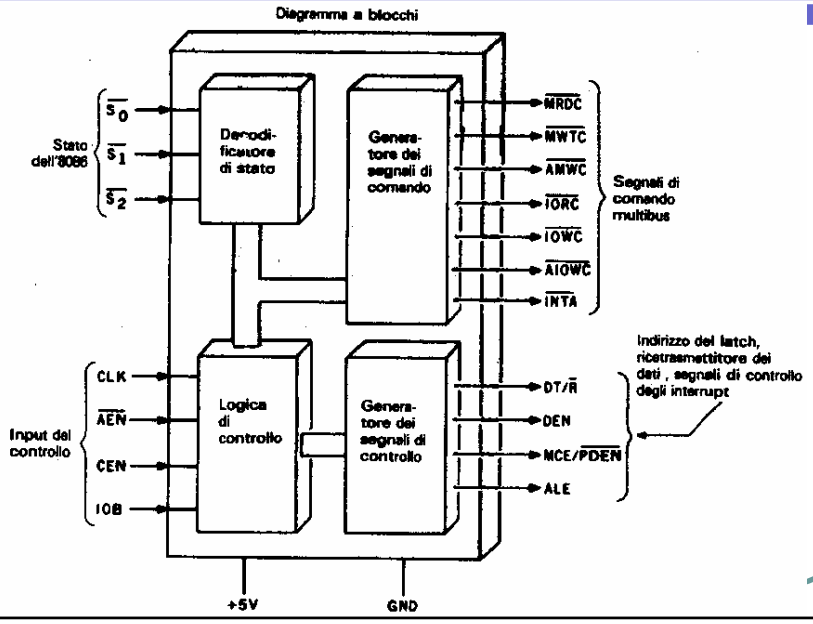
Modo Massimo dell'8086



Il Controllore di BUS, 8288



Il Controllore di BUS, 8288



384

Calcolatori Elettronici

CDL in Ingegneria Elettronica

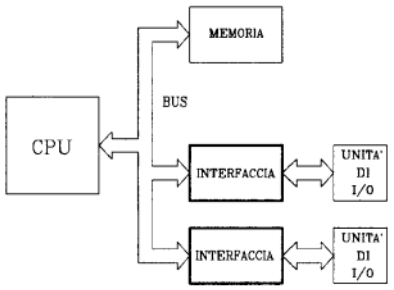
Facoltà di Ingegneria,
Università degli Studi di Firenze

Nuovo Ordinamento

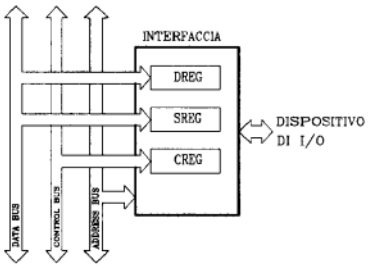
Parte 7, Il Sistema di I/O

Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
nesi@dsi.unifi.it
Agosto, 2003

Schema Generale di I/O



a) Schema di riferimento



b) Elementi fondamentali di una interfaccia

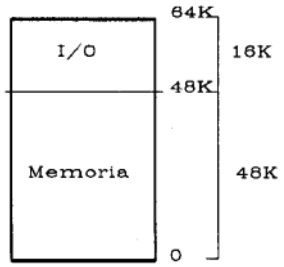
DREG (Data REGister) - **CREG** (Command REGister) - **SREG** (Status REGister)

Ci sono due modalità per associare un indirizzo a ciascun registro

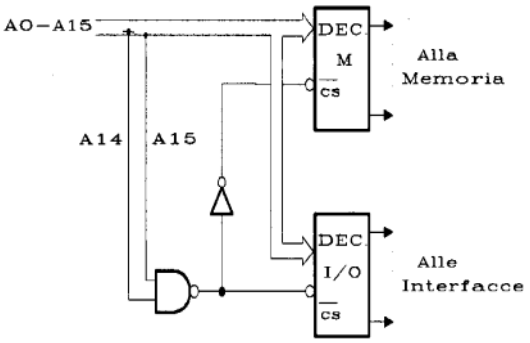
- Ingresso/uscita mappato in memoria
- Ingresso/uscita isolato

I/O mappato in memoria

- Non sono necessarie istruzioni dedicate per l'I/O



a) Mappatura



b) Decodifica indirizzi

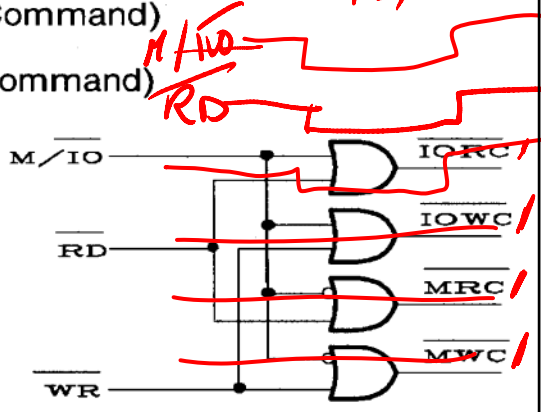
Esempio

- Mov [FEh],AH ; scrivo in IO
- Mov [03h], AL ; scrivo in Memoria

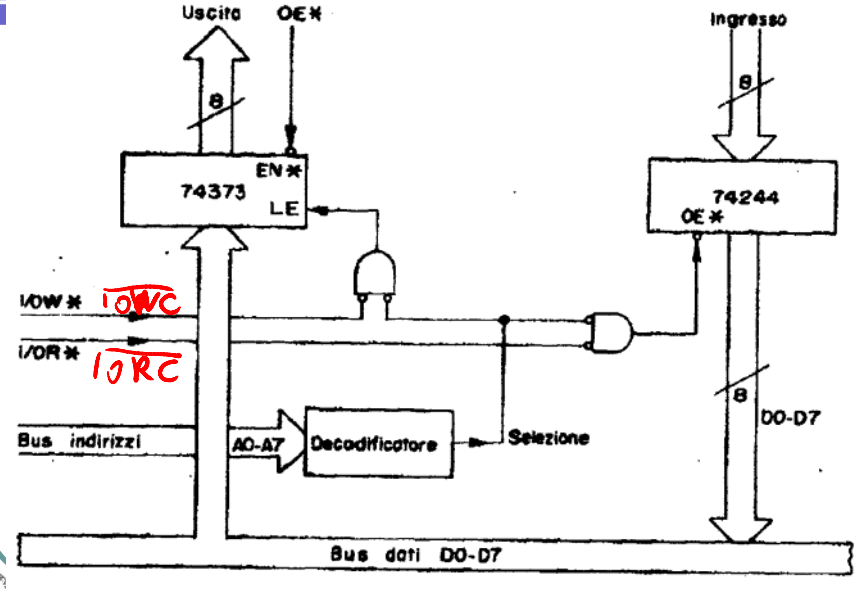
I/O Isolato, Indipendente

- Ci sono istruzioni dedicate per l'I/O
- Due specifiche linee di comando
  - IOWC (I/O Write Command)
  - IORC (I/O Read Command)

In MODO Massimo tali segnali vengo prodotti dal controllore di BUS



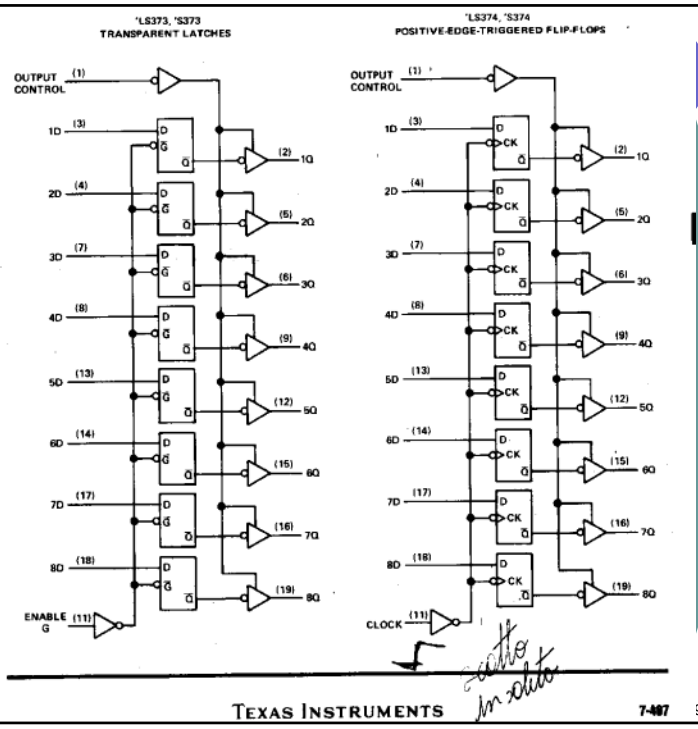
Interfaccia I/O semplice e BUS



Note sull'esempio precedente

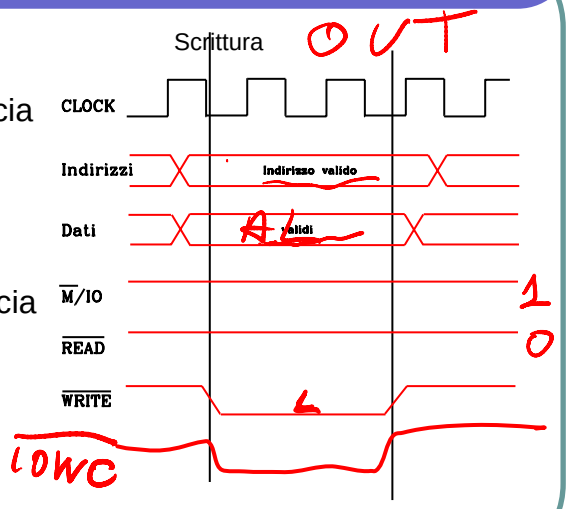
- Il 244 e' stato mostrato in precedenza ed e' un semplice Tri-State Octal Buffer
- Sia la lettura che la scrittura sono allo stesso indirizzo identificato dalla decodifica che vedremo in seguito
- Vi sono istruzioni diverse per la lettura e la scrittura su I/O, queste producono segnali di Ingresso e Uscita rispettivamente

373
e
374



Esempio

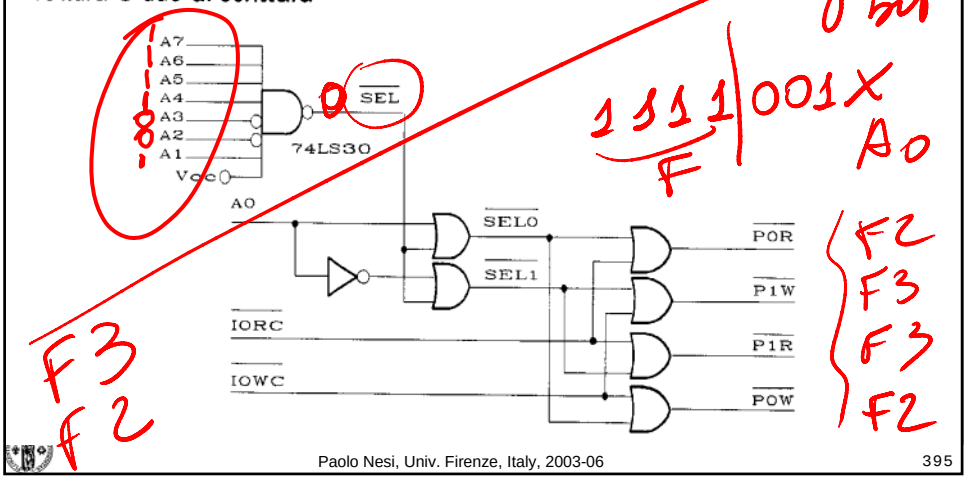
- IN ah,32h
  - leggo dall'interfaccia
- OUT 32h, AL
  - scrivo sull'interfaccia



Paolo Nesi, Univ. Firenze, Italy, 2003-06
394

Decodifica degli indirizzi

- Sull'interfaccia deve essere presente una parte di decodifica degli indirizzi
- Esempio di decodifica di comandi e indirizzi per un'interfaccia con due porte di lettura e due di scrittura



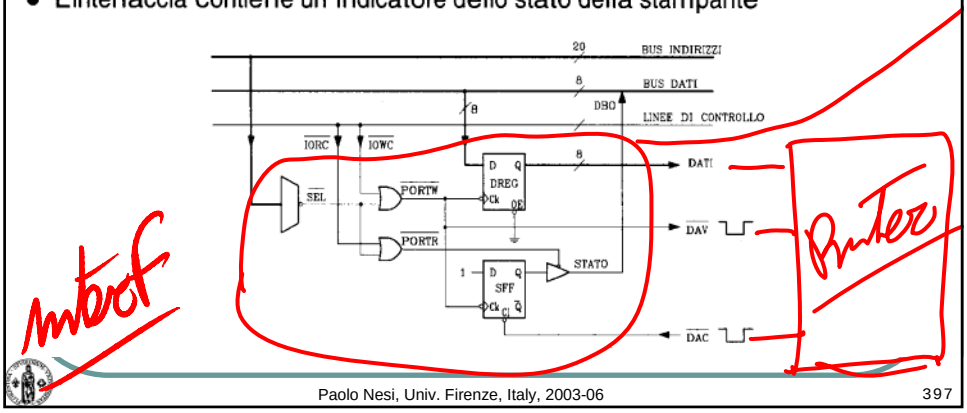
Modalità di Esecuzione di op. I/O

- Per la sincronizzazione tra programma in esecuzione e dispositivi di I/O è necessario tener conto della diversa velocità di CPU e dispositivi esterni
- Ci sono tre principali approcci:
  - Controllo di programma
  - Controllo di interruzione
  - Accesso diretto alla memoria (DMA) o con processori di I/O
- Protocollo di Comunicazione fra CPU e Periferica

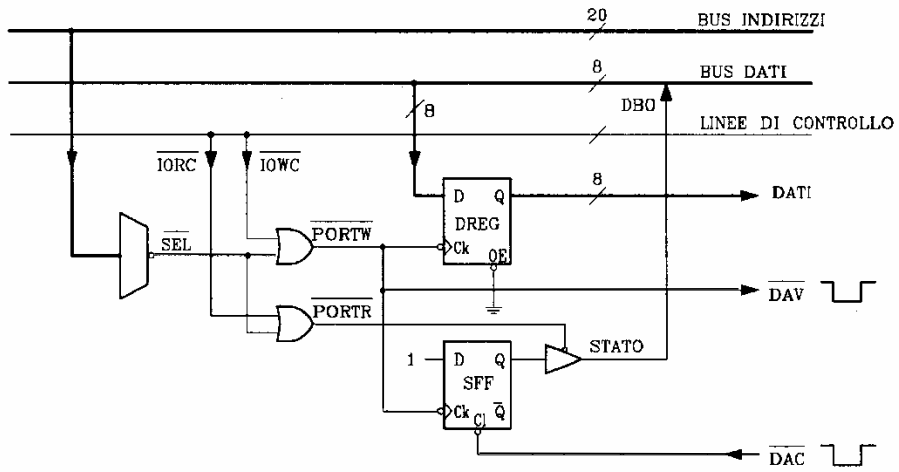
Gestione a Controllo di Programma, interfaccia

Esempio: trasferimento di un blocco di dati verso una stampante

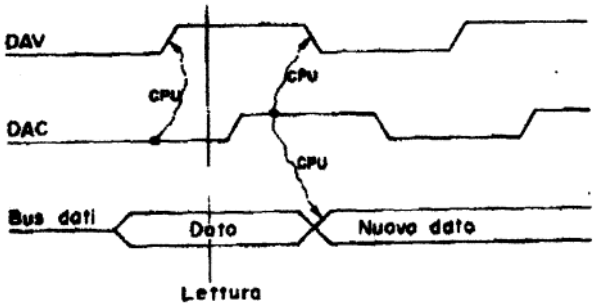
- $\overline{DAV}$ indica la disponibilità di un nuovo carattere all'ingresso della stampante
- $\overline{DAC}$ indica che la stampante ha terminato la stampa del carattere
- L'interfaccia contiene un indicatore dello stato della stampante



Gestione a Controllo di Programma, interfaccia



Evoluzione temporale dei segnali DAV e DAC



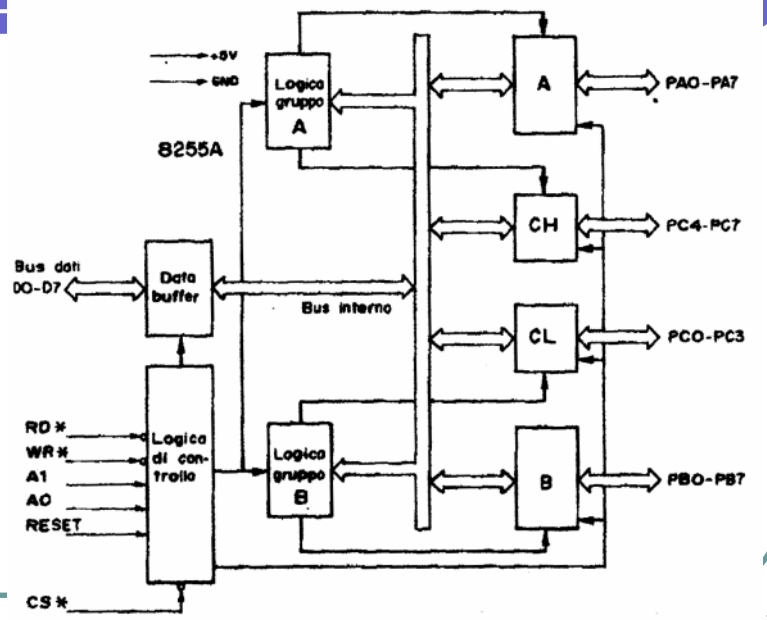
- Meccanismo di Handshaking fra CPU ed una periferica in lettura
- In questa figura DAC e DAV sono attivi alti
- Con DAV alto la CPU comunica che il dato e' presente, questo viene effettuato solo se il DAC e' basso
- Dopo la lettura la periferica mette DAC alto, questo viene letto dalla CPU che mette giù il DAV e e un nuovo dato valido
- ….

Gestione a Controllo di Programma

L'istruzione OUT PORT, AL copia il contenuto di AL in DREG dell'interfaccia, porta SFF a 1 e trasmette il $\overline{DAV}$ alla stampante

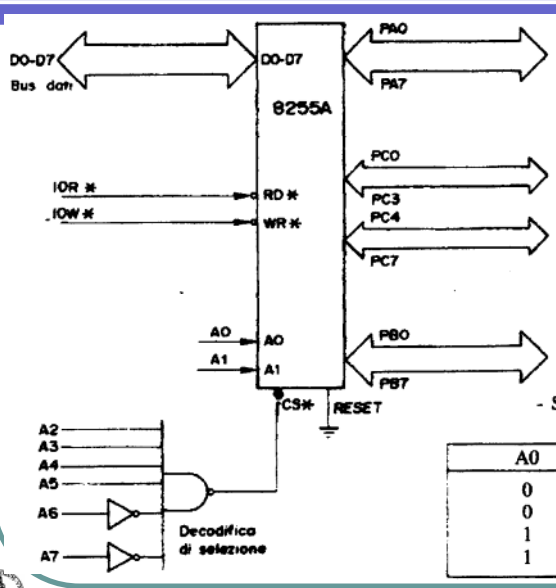
```
STAMPA:  MOV   AL, SI
         OUT   PORT, AL
ATTESA:  IN    AL, PORT
         AND   AL, 1
         JNZ   ATTESA
         INC   SI
         LOOP  STAMPA
         RET
```

I/O con l'8255A



401

Collegamento dell'8255



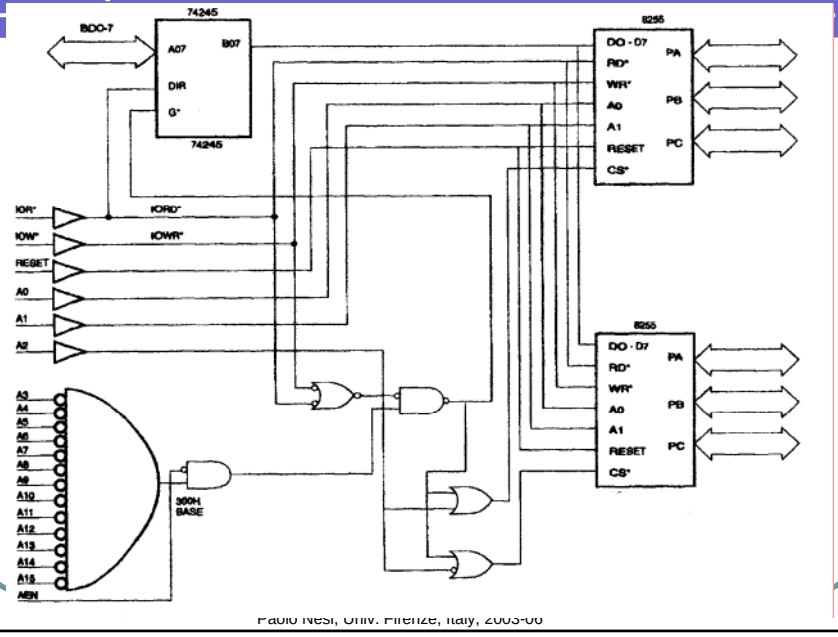
- SELEZIONE DELLE PORTE A, B, C E DEL REGISTRO R DELL'8255A

A0	A1	Porta
0	0	A
0	1	B
1	0	C
1	1	R

Paolo Nesi, Univ. Firenze, Italy, 2003-06

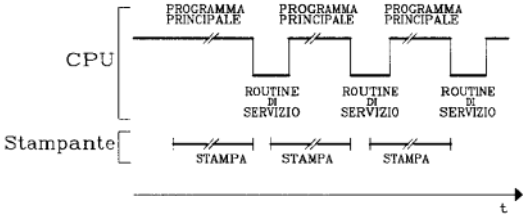
402

Esempio di Interfaccia con 8255

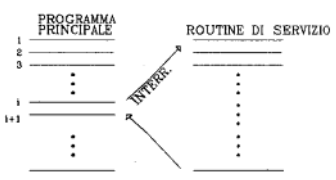


Gestione delle Interruzioni

- Le interruzioni consentono di proseguire l'esecuzione di un programma mentre viene effettuta l'operazione di I/O dalla periferica
- Un segnale INTR generato dalla periferica indica quando questa ha terminato l'operazione richiesta
- A seguito della ricezione di una richiesta di interruzione, viene eseguita una routine di servizio (*interrupt handler*)

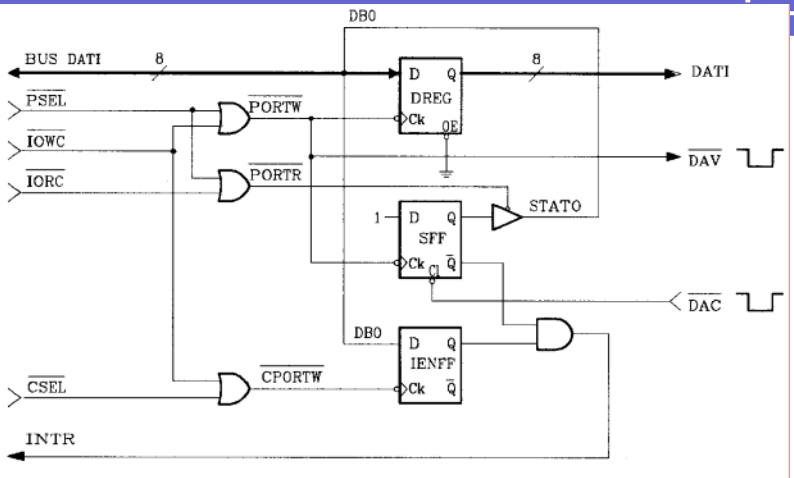


a) Temporizzazione delle attivita'



b) Esecuzione routine di servizio

Interfaccia di Uscita con Interrupt



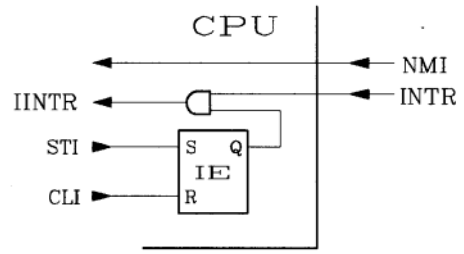
- L'interfaccia deve generare la richiesta di interruzione
- PORTW disasserisce INTR, $\overline{DAC}$ lo asserisce

Mascheramento delle Interruzioni

- È possibile mascherare (disabilitare) la richiesta di interruzione tramite il flip-flop IENFF

```
MOV     AL, 1/0      ; di Set/Reset
OUT     CPORT, AL    ; CPORT: porta controllo
```

- È posssibile mascherare l'interruzione internamente alla CPU



- Ci sono anche interruzioni non mascherabili NMI dell'8086

Routine/Procedura di Servizio

La routine di servizio dell'interruzione deve compiere alcune operazioni fondamentali

- Salvare i registri della CPU (in particolare PSW: Program Status Word)
- Servire la periferica
- Disasserire la richiesta di interruzione
- Rirpristinare le informazioni salvate al primo punto
- Ritornare al programma originale nel punto in cui era stato interrotto
  - Ripristinando il valore del PC originale
  - Riabilitando il flip-flop IE della CPU

Interruzioni da parte di più periferiche

Se si hanno più periferiche in grado di generare le interruzioni si hanno i seguenti problemi

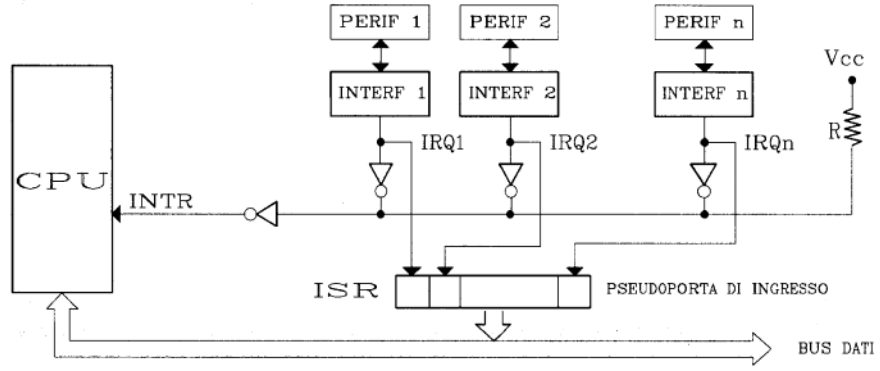
- Individuazione della periferica
- Scelta della routine di servizio
- Priorità fra periferiche (interruzioni)
- Interrompibilità della routine di servizio

E possibile:

- Gestire le richieste da programma
- Impiegare interruzioni vettorizzate
- Impiegare uno schema *daisy chain*

Gestione a Polling, Interfaccia

Le istruzioni possono essere discriminate leggendo una porta (ISR) i cui bit rappresentano lo stato delle singole IRQ

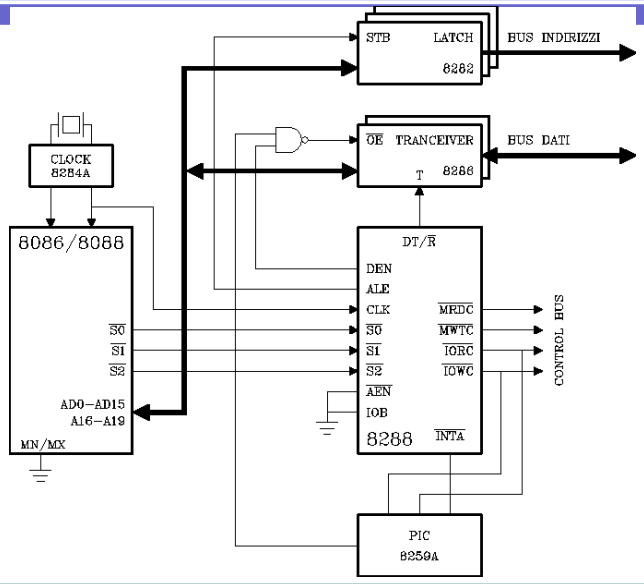


Si parla di priorità software

Polling, programma

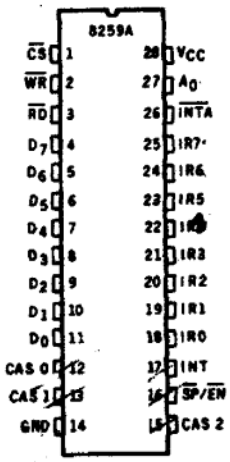
```
IN   AL,ISPORT ; lettura di ISR
MOV  COPIA,AL
AND  1 ; maschera per verificare bit 0
JNZ  ROUT0 ; salto alla routine per int 0
MOV  COPIA,AL
AND  2 ; maschera per verificare bit 1
JNZ  ROUT1 ; salto alla routine per int 1
MOV  COPIA,AL
AND  4 ; maschera per verificare bit 2
JNZ  ROUT2 ; salto alla routine per int 2
 .
 .
MOV  COPIA,AL
AND  80H ; maschera per verificare bit 7
JNZ  ROUT7 ; salto alla routine per int 7
```

8086 con un PIC, Interrupt Controller



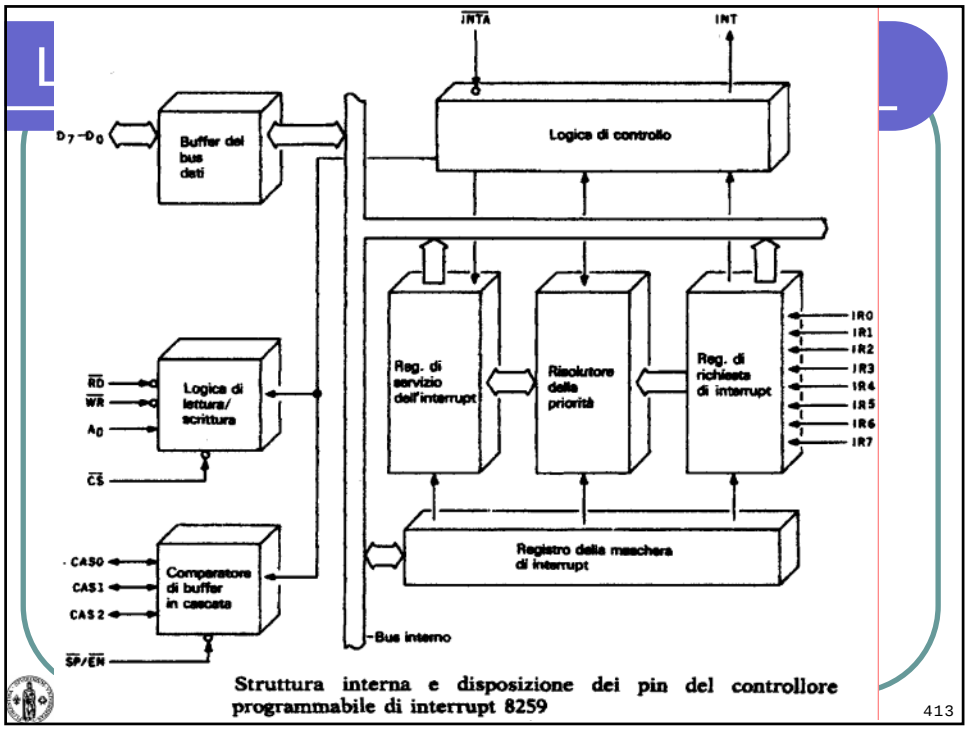
L'8059, Programmable Interrupt Controller

Disposizione dei pin



Nomi dei pin

$D_7 - D_0$	Bus dati (bidirezionale)
$\overline{RD}$	Lettura input
$\overline{WR}$	Scrittura input
A_0	Indirizzo di selezione del comando
$\overline{CS}$	Selezione del chip
CAS2 CAS0	Linee in cascata
$\overline{SP}/\overline{EN}$	Buffer di abilitazione del progr. secondario
INT	Output dell'interrupt
$\overline{INTA}$	Input di riconoscimento dell'interrupt
IR0 – IR7	Input delle richieste di interrupt



Struttura interna e disposizione dei pin del controllore programmabile di interrupt 8259

413

Calcolatori Elettronici

CDL in Ingegneria Elettronica

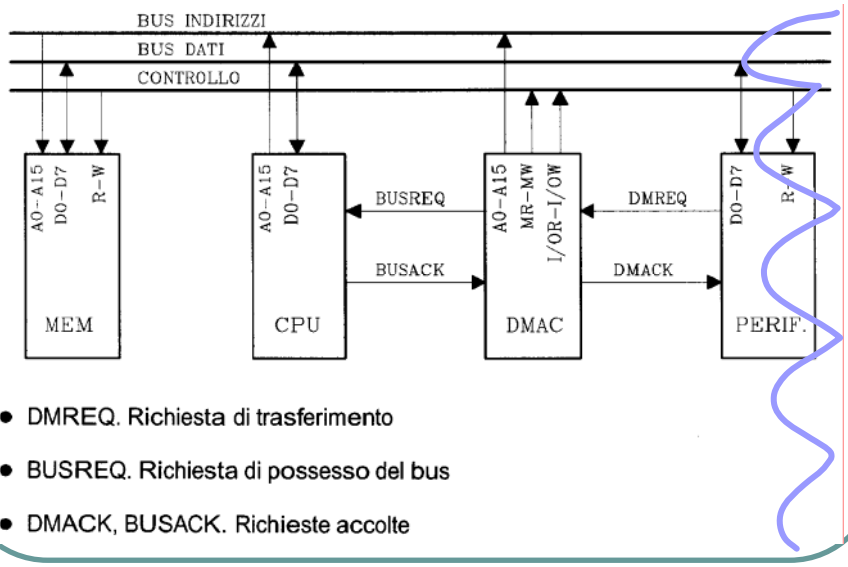
Facoltà di Ingegneria,
Università degli Studi di Firenze

Nuovo Ordinamento

Parte 8, Altri Aspetti

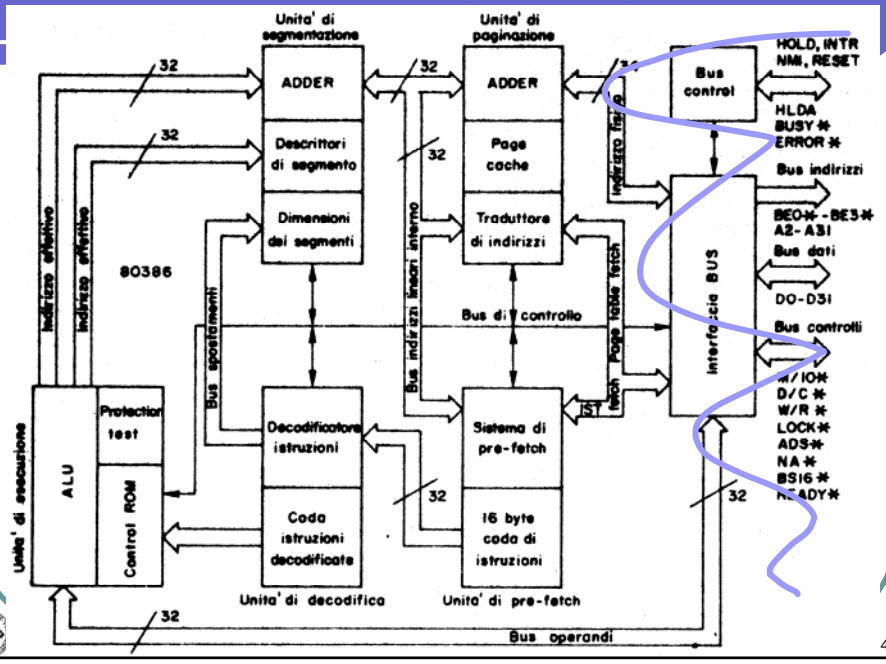
Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
nesi@dsi.unifi.it
Agosto, 2003

Il DMA, Direct Memory Access

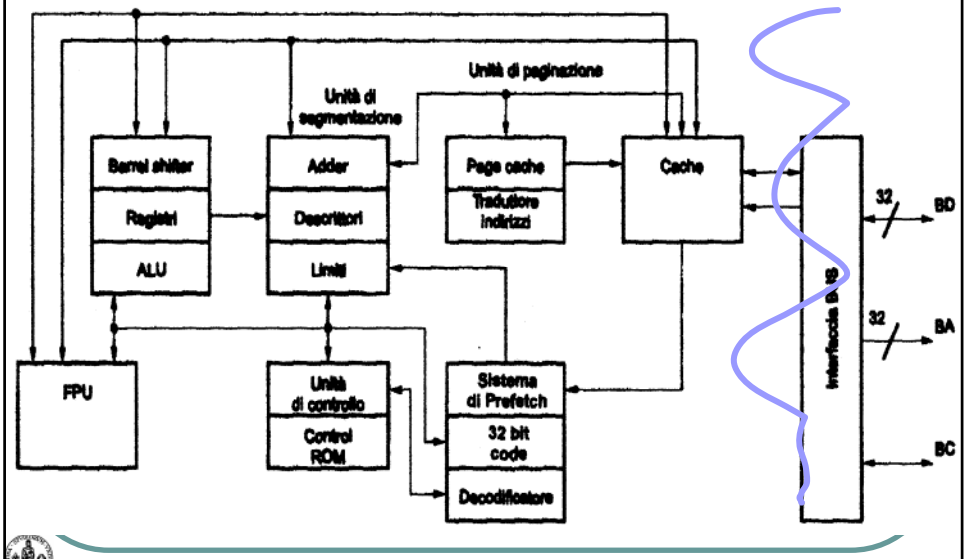


- DMREQ. Richiesta di trasferimento
- BUSREQ. Richiesta di possesso del bus
- DMACK, BUSACK. Richieste accolte

i-386



i-486



Calcolatori Elettronici

CDL in Ingegneria Elettronica

Facoltà di Ingegneria,
Università degli Studi di Firenze

Nuovo Ordinamento

Parte 8bis, La Programmazione

Prof. Paolo Nesi
Dr. Ing. Ivan Bruno
http://www.dsi.unifi.it/~ivanb
ivanb@dsi.unifi.it
2006

Introduzione alla programmazione

- Il calcolatore elettronico è uno strumento in grado di eseguire insiemi di *azioni* ("*mosse*") *elementari*
- le azioni vengono eseguite su oggetti (***dati***) per produrre altri oggetti (***risultati***)
- l'esecuzione di azioni viene richiesta all'elaboratore attraverso *frasi* scritte in qualche *linguaggio* (*istruzioni*)

Programmazione

- È l'attività con cui si predispone l'elaboratore a **eseguire un *particolare insieme di azioni* su *particolari dati***, allo scopo di *risolvere un problema.*

Problemi da risolvere

- I problemi che siamo interessati a risolvere con l'elaboratore sono di natura molto varia:
  - *Dati due numeri trovare il maggiore*
  - *Dato un elenco di nomi e relativi numeri di telefono trovare il numero di telefono di una determinata persona*
  - *Dati a e b, risolvere l'equazione ax+b=0*
  - *Stabilire se una parola viene alfabeticamente prima di un'altra*
  - *Somma di due numeri interi*
  - *Scrivere tutti gli n per cui l'equazione: $X^n+Y^n = Z^n$ ha soluzioni intere (problema di Fermat)*
  - *Ordinare una lista di elementi*
  - *Calcolare il massimo comun divisore fra due numeri dati.*
  - *Calcolare il massimo in un insieme.*
  - *….*

Risoluzione dei problemi

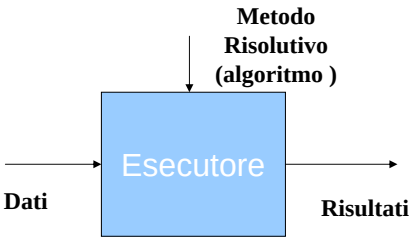
- La descrizione del problema non fornisce (in generale) un metodo per risolverlo.
  - Affinché un problema sia risolvibile è però necessario che la sua definizione sia chiara e completa
- Non tutti i problemi sono risolvibili attraverso l'uso del calcolatore. Esistono classi di problemi per le quali la soluzione automatica non è proponibile. Ad esempio:
  - se il problema presenta infinite soluzioni
  - per alcuni dei problemi **non è stato trovato** un metodo risolutivo
  - per alcuni problemi è stato dimostrato che **non esiste** un metodo risolutivo automatizzabile

Risoluzione dei problemi

- *La risoluzione di un problema è il processo che, dato un problema e individuato un opportuno metodo risolutivo, trasforma i dati iniziali nei corrispondenti risultati finali.*
- Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso del calcolatore, tale processo deve poter essere definito come *sequenza di azioni elementari.*

Algoritmo

- Un algoritmo è una sequenza **finita** di mosse che risolve ***in un tempo finito*** una *classe* di problemi.
- L'esecuzione delle azioni *nell'ordine specificato dall'algoritmo* consente di ottenere, a partire dai dati di ingresso, i risultati che risolvono il problema.



ESECUTORE
una ***macchina astratta***
capace di ***eseguire le azioni***
specificate dall'algoritmo.

Algoritmi: proprietà

- **Eseguibilità**: ogni azione dev'essere *eseguibile* dall'esecutore *in un tempo finito*.
- **Non ambiguità**: ogni azione deve essere *univocamente interpretabile* dall'esecutore.
- **Finitezza**: il numero totale di azioni da eseguire, per ogni insieme di dati di ingresso, deve essere finito.

Algoritmi: proprietà (2)

- **Quindi, l'algoritmo deve:**
  - essere *applicabile a qualsiasi insieme di dati di ingresso* appartenenti al **dominio di definizione** dell'algoritmo
  - essere costituito da operazioni appartenenti ad un determinato **insieme di operazioni fondamentali**
  - essere costituito da **regole non ambigue**, cioè interpretabili in modo **univoco** qualunque sia l'esecutore (persona o "macchina") che le legge.

Algoritmi e programmi

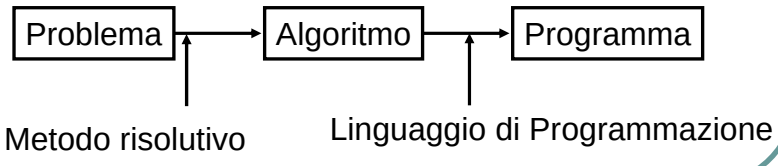
- Ogni elaboratore è una macchina in grado di eseguire azioni elementari su oggetti detti **DATI**.
- L'esecuzione delle azioni è richiesta all'elaboratore tramite comandi elementari chiamati **ISTRUZIONI** espresse attraverso un opportuno formalismo: il **LINGUAGGIO di PROGRAMMAZIONE**.
- La formulazione testuale di un algoritmo in un linguaggio comprensibile a un elaboratore è detta **programma**.

Programma

- **Un programma è un testo** scritto in accordo alla **sintassi** e alla **semantica** di un linguaggio di programmazione.
- Un ***programma*** è la **formulazione testuale,** in un certo linguaggio di programmazione, di un **algoritmo** che risolve un dato *problem*a.

Algoritmo & programma

- Passi per la risoluzione di un problema:
  - individuazione di un procedimento risolutivo
  - scomposizione del procedimento in un insieme ordinato di azioni ⇒ **ALGORITMO**
  - rappresentazione dei dati e dell'algoritmo attraverso un formalismo comprensibile dal calcolatore
    ⇒ **LINGUAGGIO DI PROGRAMMAZIONE**



Metodo risolutivo

Linguaggio di Programmazione

Algoritmi equivalenti

- Due algoritmi si dicono ***equivalenti*** quando:
  - hanno lo stesso **dominio di ingresso**;
  - hanno lo stesso **dominio di uscita**;
  - in corrispondenza degli **stessi valori del dominio di ingresso *producono gli stessi valori* nel dominio di uscita.**
- Due algoritmi ***equivalenti***
  - forniscono lo **stesso risultato**
  - ma possono avere ***diversa efficienza***
  - e possono essere **profondamente diversi!**

Algoritmi: esempi

- **Soluzione dell'equazione ax+b=0**
  - leggi i valori di a e b
  - calcola -b
  - dividi quello che hai ottenuto per a e chiama x il risultato
  - stampa x
- **Calcolo del massimo di un insieme**
  - Scegli un elemento come massimo provvisorio *max*
  - Per ogni elemento *i* dell'insieme: se *i>max* eleggi *i* come nuovo massimo provvisorio *max*
  - Il risultato è *max*

Algoritmi: esempi

- **Stabilire se una parola P viene alfabeticamente prima di una parola Q**
- **leggi** P,Q
- **ripeti quanto segue:**
  - **se** prima lettera di P < prima lettera di Q
  - **allora** scrivi vero
  - **altrimenti se** prima lettera P > Q
    - **allora** scrivi falso
    - **altrimenti** (le lettere sono =) togli da P e Q la prima lettera
- **fino** a quando hai trovato le prime lettere diverse.

Linguaggio & Programma

- Dato un algoritmo, un **programma** **è la sua descrizione *in un particolare linguaggio*** di programmazione
- **Un linguaggio di programmazione** **è una notazione formale** che può essere usata per descrivere algoritmi.

Linguaggi di Programmazione

I linguaggi sono classificati in:

- Low level
  ⇒ Linguaggio macchina & Linguaggio Assembly
- High level
  ⇒ C/C++, Pascal, Delphi, Java, Cobol….

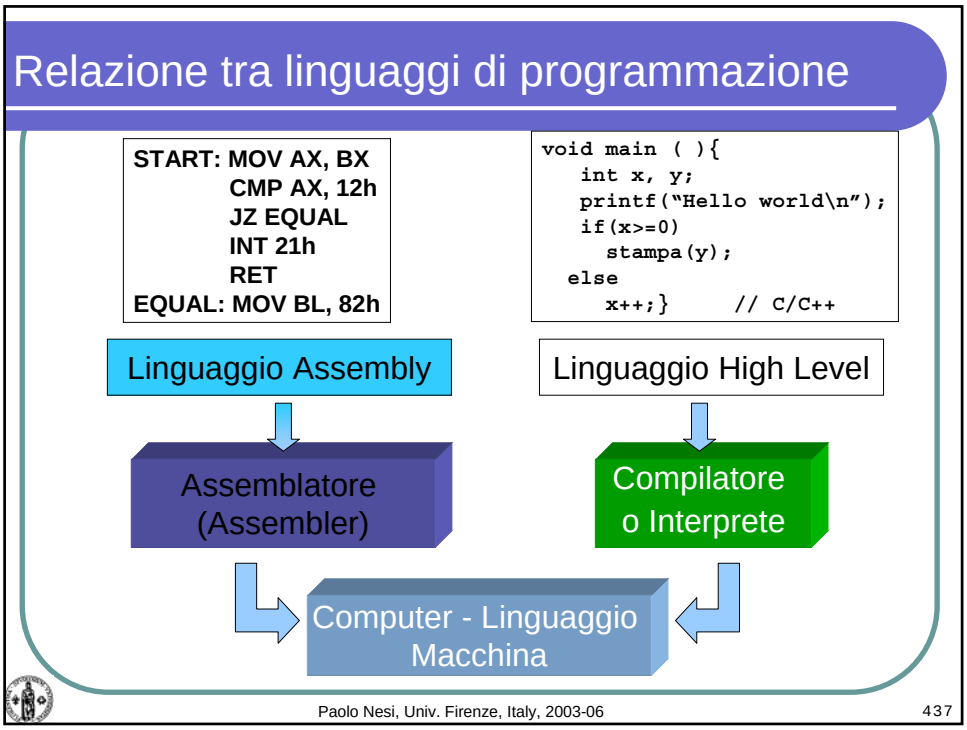
Linguaggi di Programmazione

- I linguaggi *low level* sono strettamente legati all'architettura interna del computer
- Ad ogni singola azione corrisponde una linea di programma (rapporto 1 a 1)
- I linguaggi *high level* sono più "vicini" al nostro modo di pensare
- Devono essere opportunamente tradotti per essere compresi da un computer

Assembly

- Il linguaggio naturale di un computer è il *linguaggio macchina*
- Le istruzioni del linguaggio macchina sono rappresentate da una sequenza binaria di '1' e '0'
- Il linguaggio *Assembly* è il linguaggio più vicino alla macchina ma deve essere tradotto in *linguaggio macchina* per essere eseguito (*Assembler*)
- Processori diversi hanno differenti linguaggi macchina e perciò necessitano di un proprio linguaggio assembly.

Relazione tra linguaggi di programmazione



Sintassi e semantica

Aspetti caratteristici del linguaggio:

- Esistenza di un insieme di **parole chiave**
- **Sintassi**: **l'insieme di regole formali per la scrittura di programmi** in un linguaggio, che dettano le ***modalità per costruire frasi corrette*** nel linguaggio stesso.
- **Semantica**: **l'insieme dei significati** da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio.

Compilatore vs Interprete

- I **compilatori** traducono automaticamente un programma dal linguaggio *L* a quello macchina (per un determinato elaboratore).
  - Durante la traduzione verificano la correttezza di ciascuna istruzione.
  - La traduzione termina quando non ci sono più errori sintattici.
  - Al termine della traduzione il programma è pronto per essere eseguito.

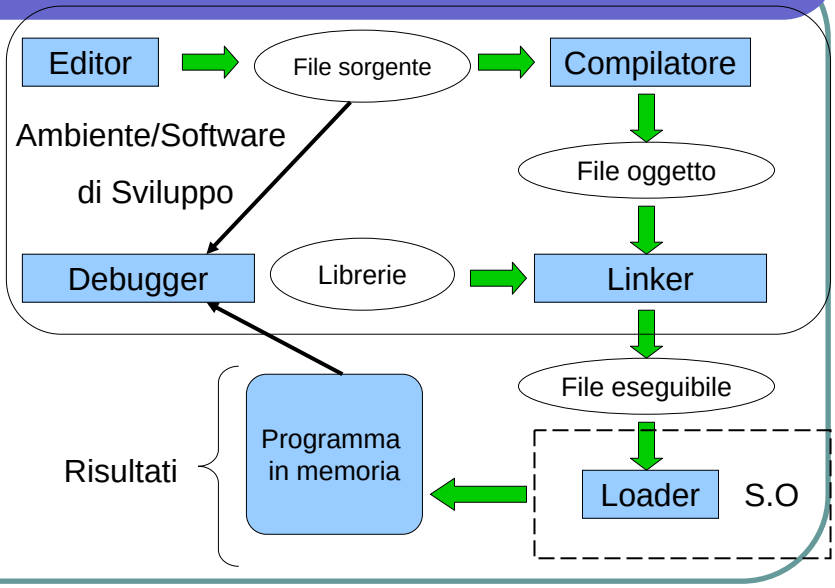
Compilatore vs Interprete

- Gli **interpreti** sono programmi capaci di eseguire direttamente un programma in linguaggio *L* istruzione per istruzione.
  - Verificano la correttezza sintattica di ogni istruzione durante l'esecuzione.
  - In presenza di istruzioni ripetute (cicli) queste sono verificate e tradotte come se fossero da eseguire per la prima volta.
- Prestazioni:
  - I programmi compilati sono in generale più efficienti e più veloci di quelli interpretati.

Le fasi della compilazione:

- **Compilatore:** opera la **traduzione di un programma *sorgente*** (scritto in un linguaggio ad alto livello) **in un *programma oggetto.***
- **Linker:** (*collegatore*) nel caso in cui la costruzione del programma oggetto richieda l'unione di ***più moduli o librerie*** (compilati separatamente), il linker provvede a **collegarli** formando un unico *programma eseguibile.*
- **Debugger:** consente di **eseguire passo-passo** un programma, **controllando via via quel che succede**, al fine di ***scoprire ed eliminare errori*** non rilevati in fase di compilazione.

Ambiente di sviluppo



Come sviluppare un programma

- Qualunque sia il linguaggio di programmazione scelto occorre:
  - Scrivere il ***testo del programma*** e memorizzarlo su supporti di memoria permanenti (fase di editing);
- Se il linguaggio è compilato:
  - Compilare il programma, ossia utilizzare il compilatore che effettua una traduzione automatica del programma scritto in un linguaggio qualunque in un programma equivalente scritto in ***linguaggio macchina***;
  - Eseguire il programma tradotto.
- Se il linguaggio è interpretato:
  - Usare l'interprete per eseguire il programma.