

Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica



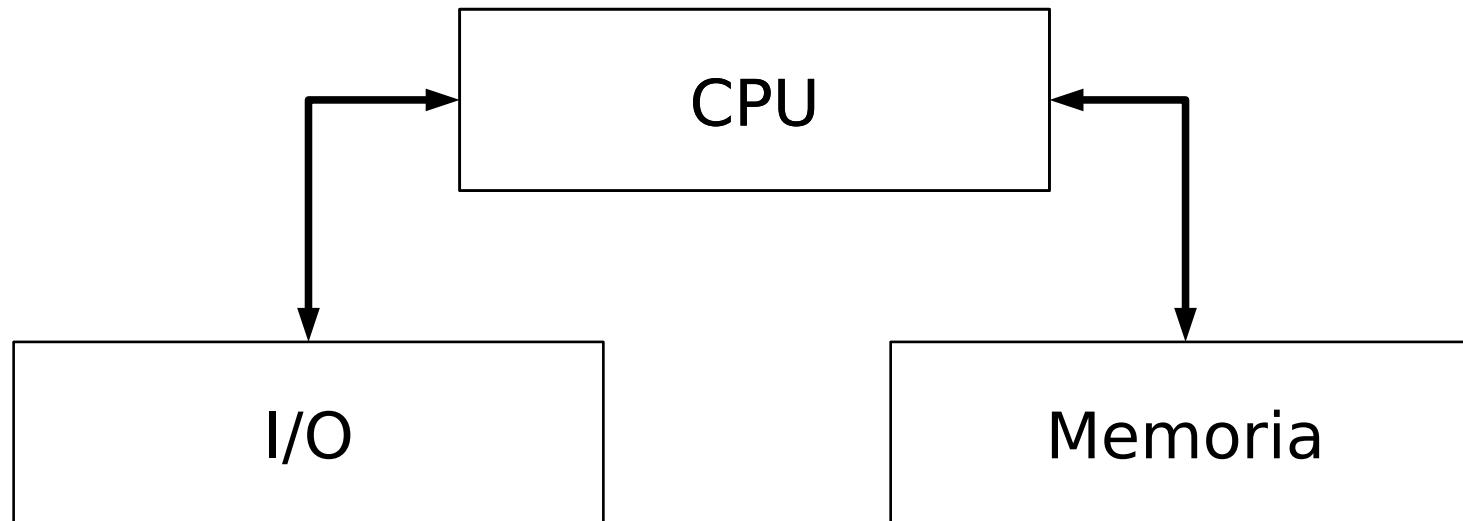
Corso di Calcolatori Elettronici I

Introduzione all'architettura dei calcolatori



Calcolatore: sottosistemi

- Processore o CPU (Central Processing Unit)
- Memoria centrale
- Sottosistema di input/output (I/O)



Calcolatore: organizzazione a bus

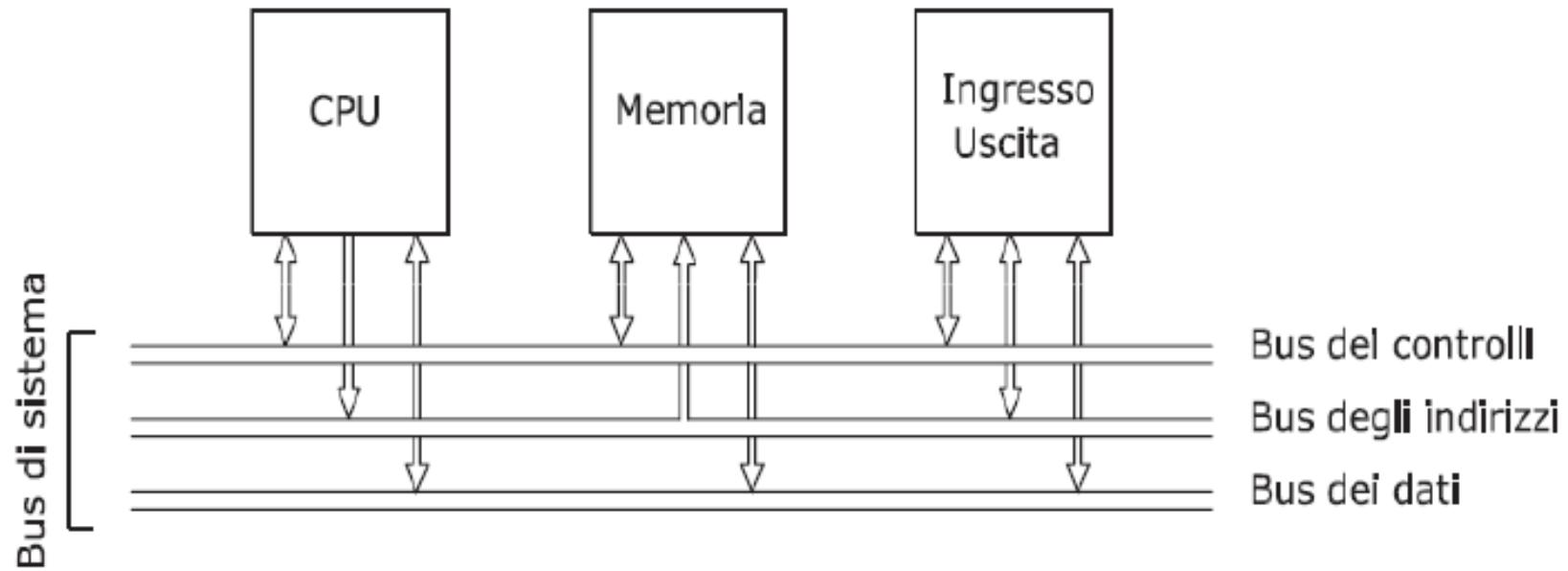
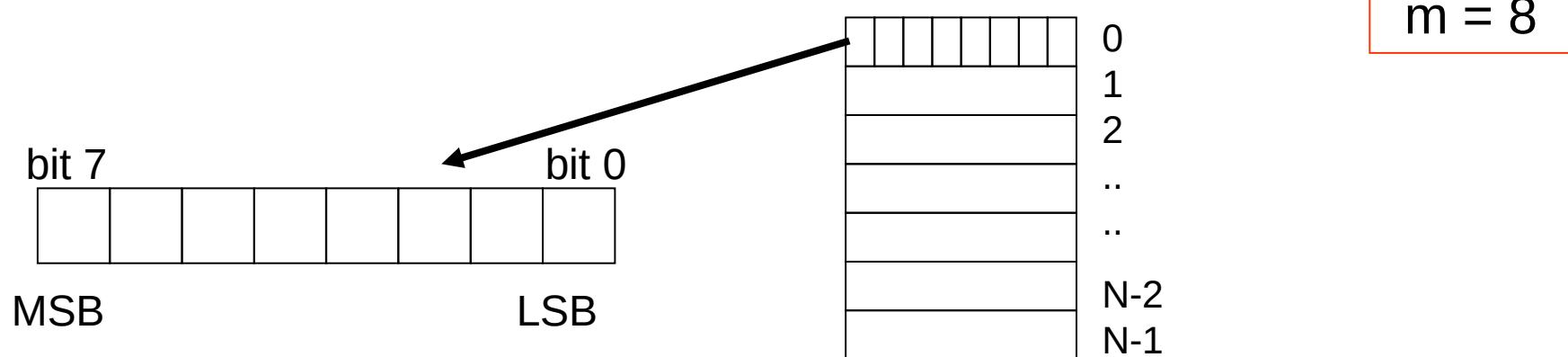


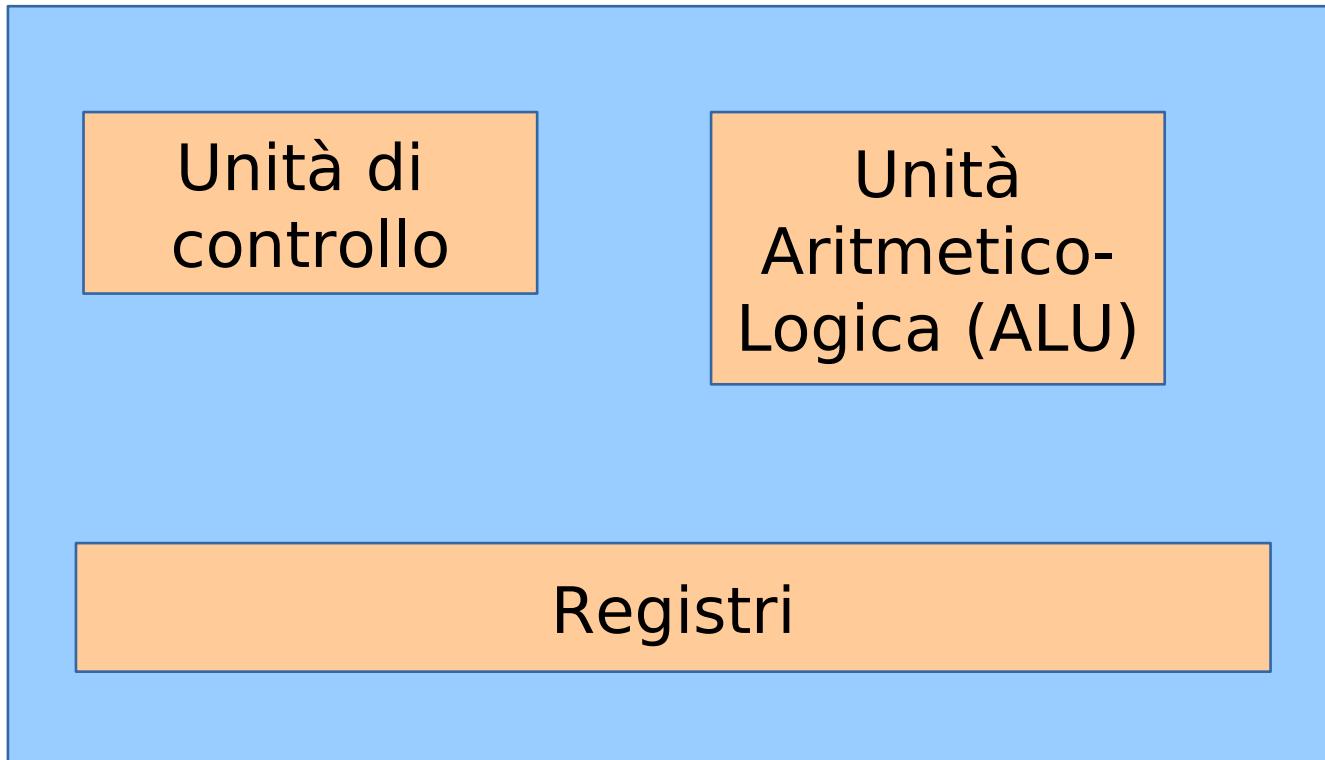
Figura 5.2 - Organizzazione a bus. Un bus è semplicemente un percorso che collega tutte le parti che si affacciano su di esso. La figura mostra che il bus di sistema si compone di tre sotto-bus.

La memoria centrale

- La memoria centrale di un computer è organizzata come un array di stringhe di bit di lunghezza m, dette locazioni
- Gli m bit di una locazione sono accessibili dal processore (in lettura/scrittura) mediante un'unica operazione
- Ogni locazione è individuata da un indirizzo, cioè un intero compreso tra 0 e N-1, con $N = 2^k$
 - $[0, N-1] = \text{SPAZIO DI INDIRIZZAMENTO}$
- La memoria centrale è ad accesso casuale (RAM) cioè il tempo di accesso non dipende dalla posizione del dato



Il processore o CPU



CPU: Struttura interna

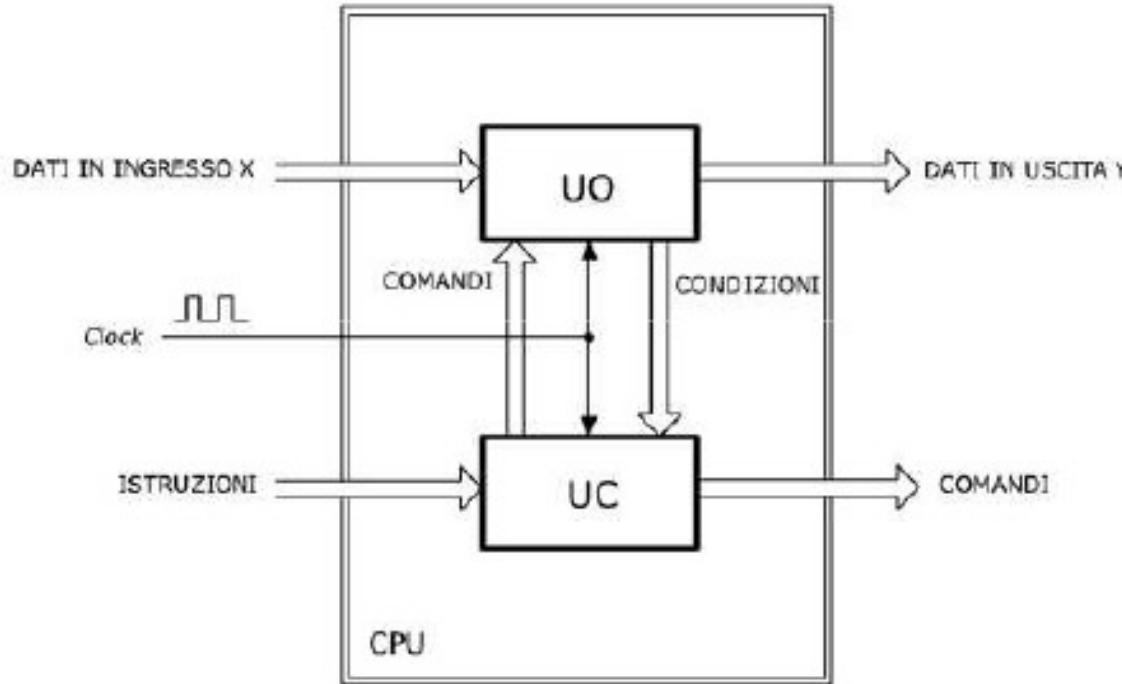


Figura 5.8 - Struttura generale della CPU. Lo schema mette in evidenza due parti: l'unità operativa (UO) e l'unità di controllo (UC). Lo schema mostra anche i flussi informativi tra UC e UO e tra l'intera CPU e l'esterno [LP86].

Il funzionamento della CPU è scandito dal clock

CPU: Struttura interna

- Componenti fondamentali del processore:
 - Unità di controllo
 - registro Program Counter (PC) o Prossima Istruzione
 - Instruction Register o registro di decodifica (IR o D)
 - registri di Macchina
 - Unità aritmetico-logica (ALU)
 - Sezione di Collegamento con la memoria
 - registro degli indirizzi di memoria o Memory Address Register MAR
 - registro di transito dei dati dalla memoria DTR o Memory Buffer MB
 - Sezione di Collegamento con Ingresso-Uscita
- Il linguaggio macchina di un processore è costituito dalla codifica in binario delle istruzioni eseguibili dal processore

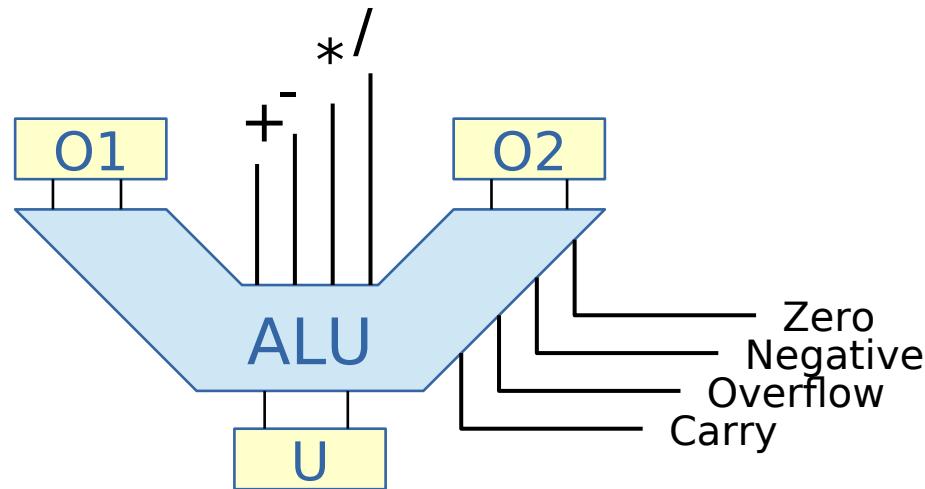
Registri della CPU

- Registri interni
 - Necessari al funzionamento del processore
 - Non direttamente visibili al programmatore
 - non appartengono al *modello di programmazione*
 - Es. MAR, MDR, IR, ...
- Registri di macchina
 - Visibili al programmatore
 - appartengono al *modello di programmazione*
 - Registri generali (R0, R1, Rn-1)
 - Registri speciali (PC, SR, ...)

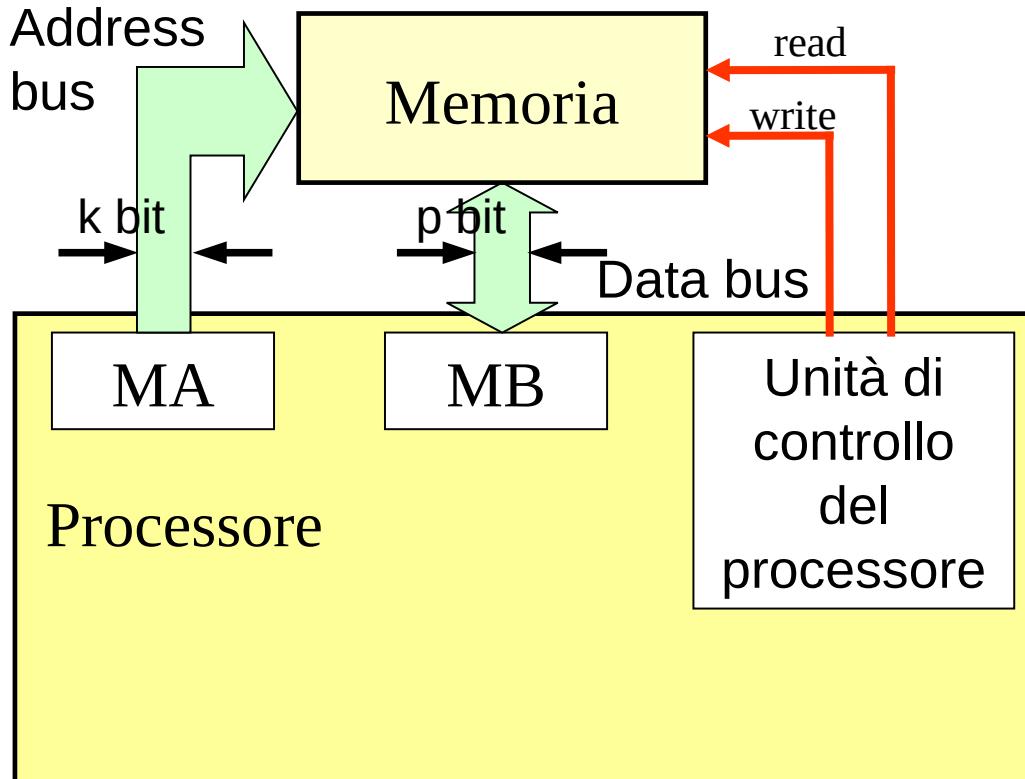
Unità aritmetico-logica (ALU)



- L'Unità di controllo fornisce alla ALU gli operandi, insieme ad un comando che indica l'operazione da effettuare
- Gli operandi sono copiati nei registri di ingresso della ALU (O1, O2)
- La ALU esegue l'operazione e pone il risultato nel registro risultato (U); inoltre, altera il valore dei flag del registro di stato (SR) in funzione del risultato



Interazione processore-memoria



Processori a carattere e processori a parola

- I processori “a carattere” accedono alla memoria con parallelismo 1 byte (8 bit)
 - prime CPU ad accumulatore
- I processori “a parola” hanno la capacità di indirizzare ed accedere la memoria per unità (parole o word) di 16 bit, 32 bit o 64 bit
- In questi sistemi (tranne pochissime eccezioni nel passato) l’unità indirizzabile di memoria (locazione) è ancora il byte
 - Si parla di sistemi a memoria byte-addressable: ogni byte ha il suo indirizzo
- Terminologia Motorola 68000
 - 1 byte, word = 2 byte, longword = 4 byte

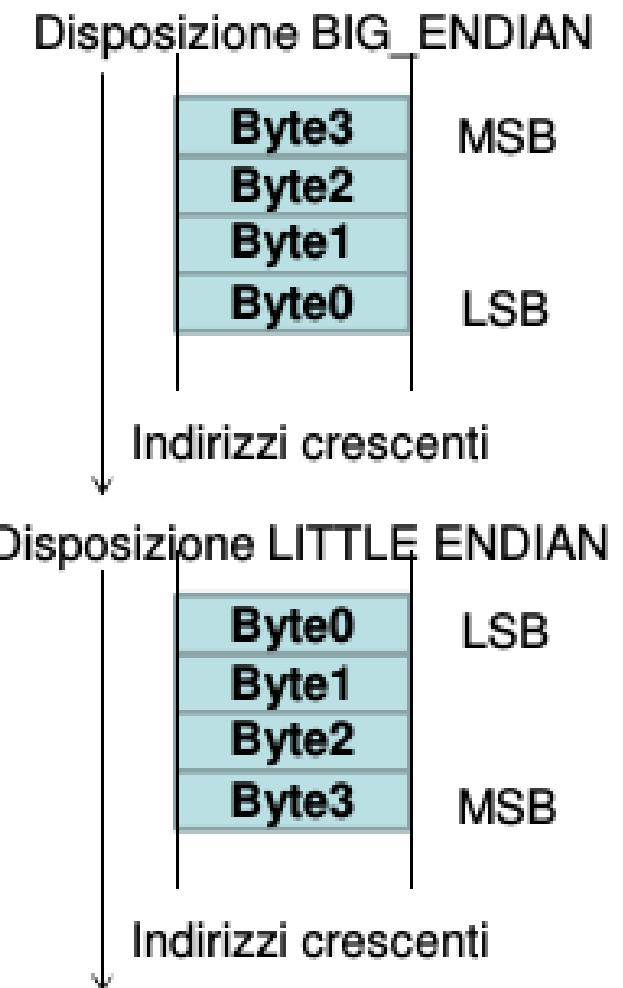
Big-endian e little-endian

- Immaginiamo di avere un processore a parola, con parole di 32 bit (4 byte) e voler scrivere in memoria il valore intero (esadecimale) \$12FA34ED all'indirizzo 812
- Le figure sottostanti illustrano il contenuto della memoria nei due casi big-endian e little-endian



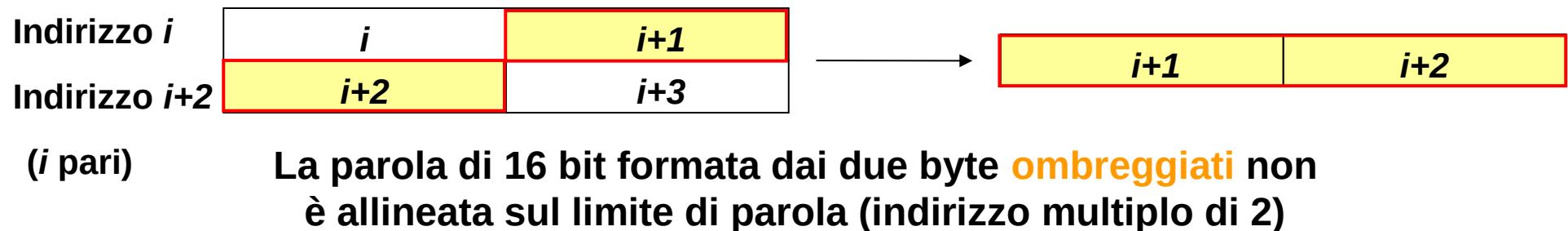
Big-endian e little-endian

- I processori possono disporre in memoria i byte che formano una parola da 16, 32 o 64 bit in 2 modi
 - Big-endian:
 - i byte sono disposti in memoria in modo che il byte più significativo (MSB) occupi la locazione di memoria di indirizzo minore, e poi via via gli altri, fino a quello meno significativo (LSB) che è collocato nella locazione di indirizzo maggiore
 - Little-endian: disposizione opposta
- Il processore Motorola 68000 usa la convenzione Big Endian



Memoria: parole allineate e non

- Per un processore a parola di 16 bit, una parola che inizia ad un indirizzo pari si dice “allineata sul limite di parola”
- Tipicamente, un tale processore è in grado di accedere ai due byte che costituiscono una parola allineata mediante una sola operazione di lettura
- Il processore 8086 consente l'utilizzo di parole non allineate, cioè parole che iniziano ad un indirizzo dispari, ma in tal caso sono necessari 2 distinti accessi in memoria
- Il processore 68000 NON consente l'accesso a parole non allineate



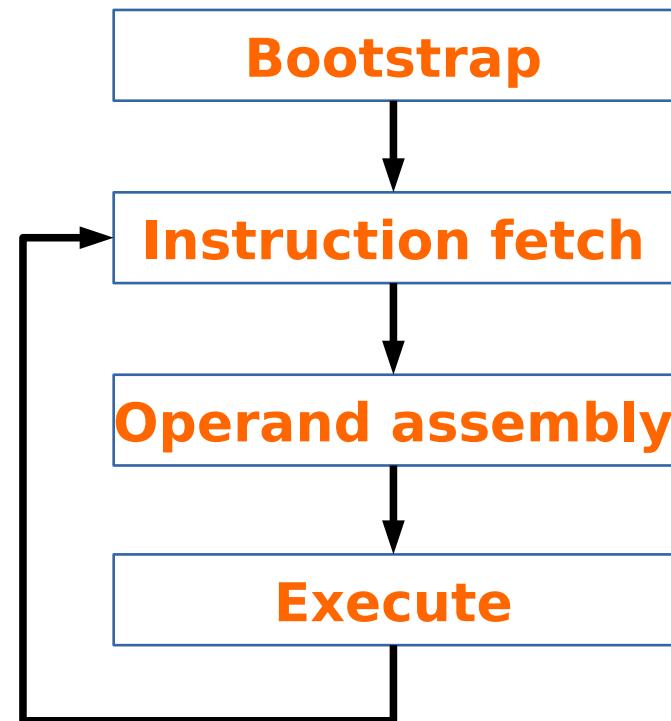
Algoritmo del processore

- **Prelievo dell'istruzione** (Fetch)
 - La CPU preleva dalla memoria l'istruzione il cui indirizzo è in PC
 - L'istruzione viene copiata nel registro IR
- **Decodifica / prelievo degli operandi** (Operand Assembly)
 - L'unità di controllo esamina il contenuto di IR e ricava il tipo di operazione ed i relativi operandi
 - Eventuali operandi contenuti in memoria vengono prelevati
- **Esecuzione dell'istruzione** (Execute)
 - L'unità di controllo richiede all'ALU di effettuare l'operazione specificata nell'istruzione ed invia il risultato ad un registro o alla memoria

Algoritmo del processore

- L'unità di controllo opera in un ciclo infinito:
 1. Prelievo
 2. Preparazione degli operandi
 3. Esecuzione

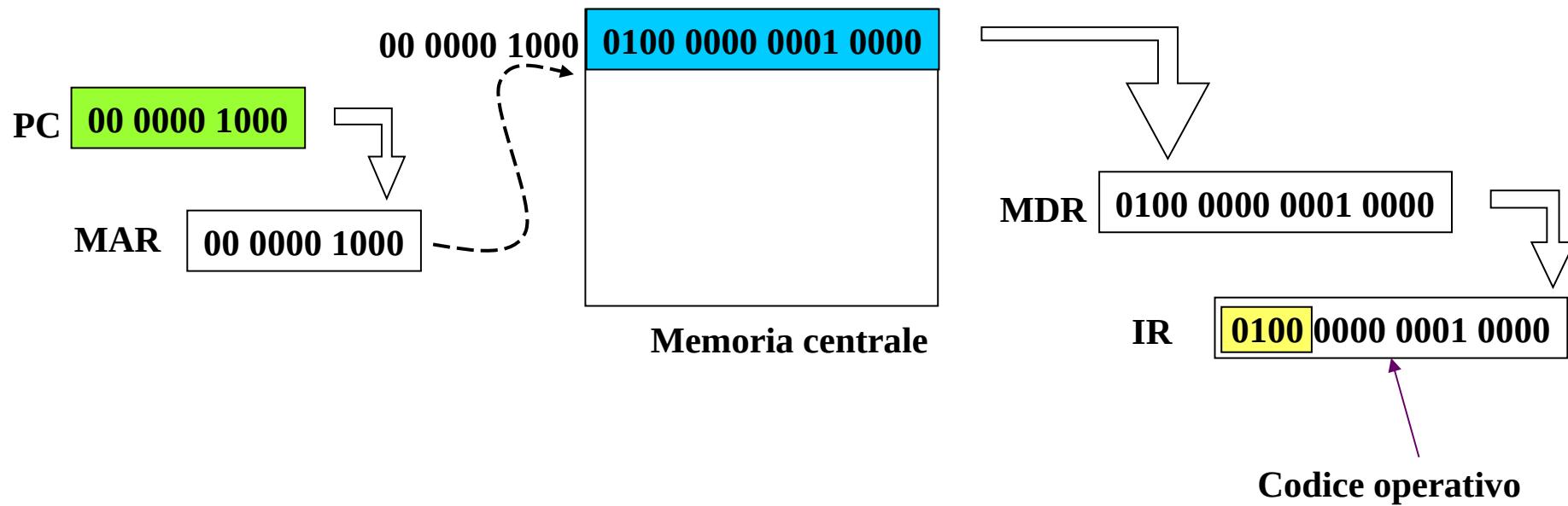
Nella fase di bootstrap il ciclo viene inizializzato;
viene assegnato un valore iniziale opportuno a PC in modo da avviare l'esecuzione di un programma iniziale in ROM



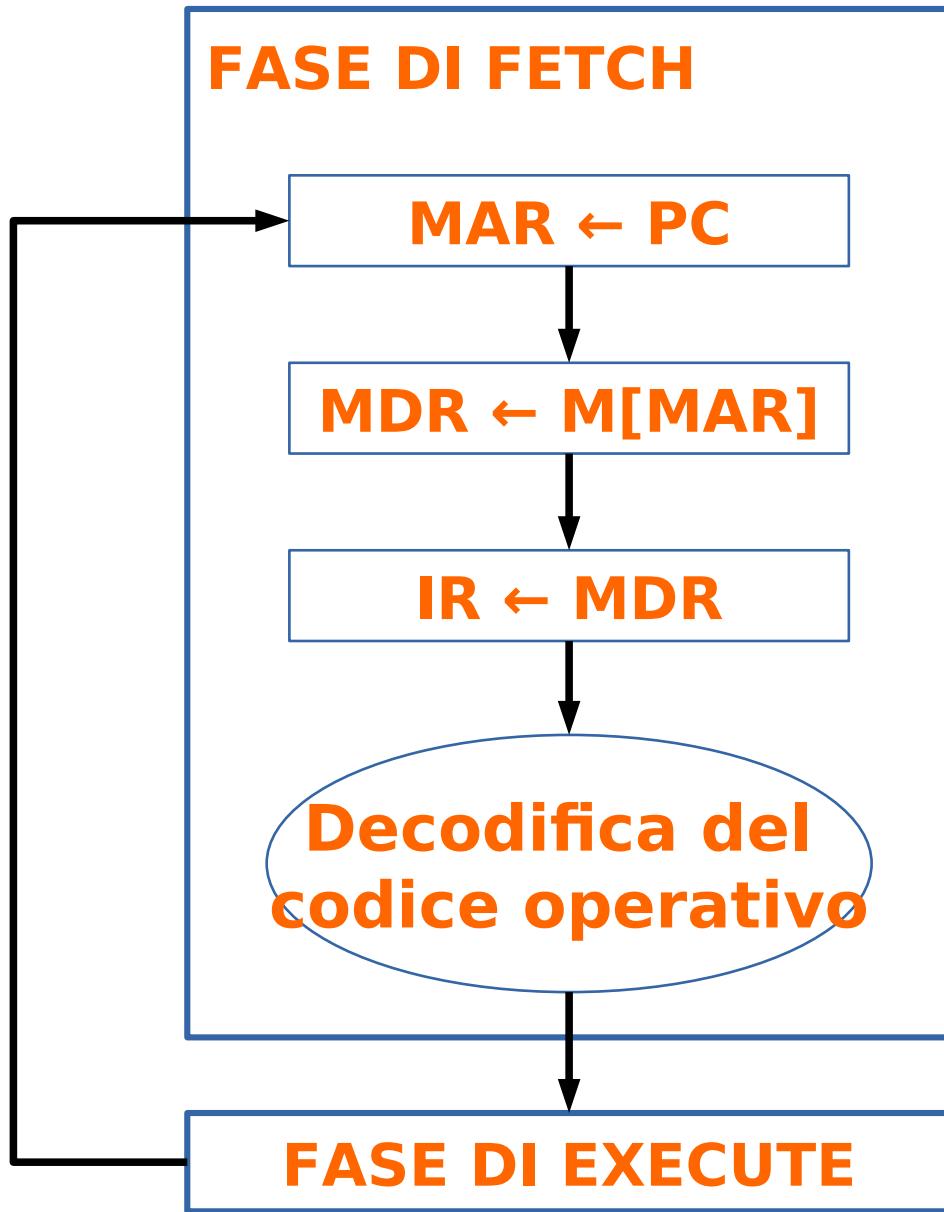
Fase di fetch



- $IR = M[PC]$; $PC = PC + k$

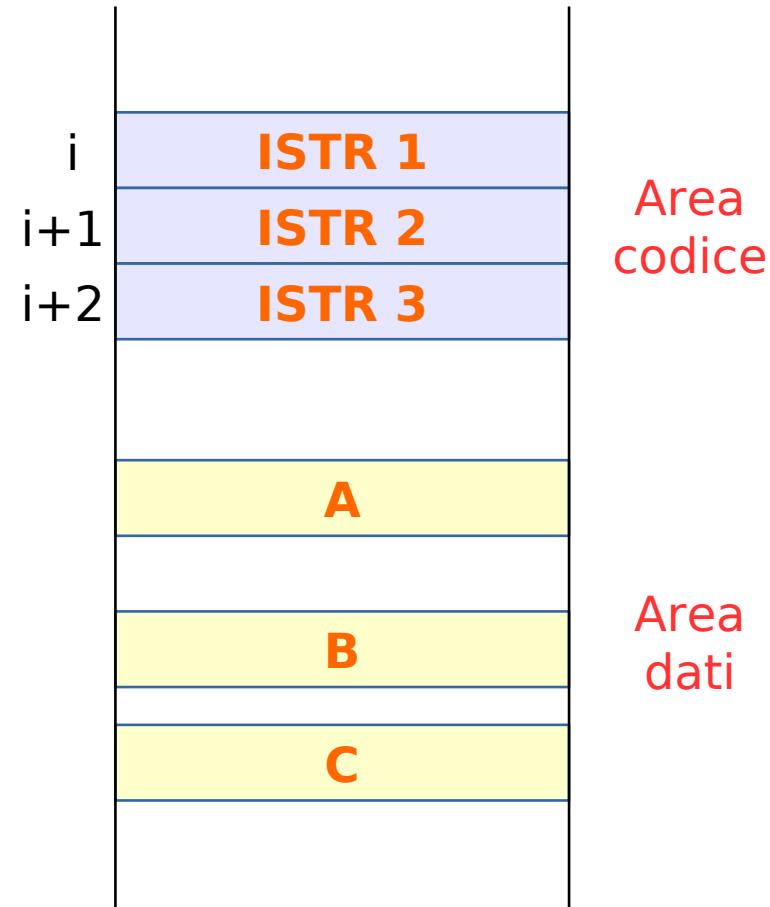


Fase di fetch: sottopassi



Esecuzione sequenziale delle istruzioni

- Alla fine della fase fetch:
 - $PC=PC+k$
 - k = lunghezza istruzioni in byte
- serve a far sì che PC punti all'istruzione posta subito dopo
- Esecuzione delle istruzioni in sequenza così come sono memorizzate
- Per cicli e figure di controllo (if-then, if-then-else, switch) occorrono istruzioni di salto





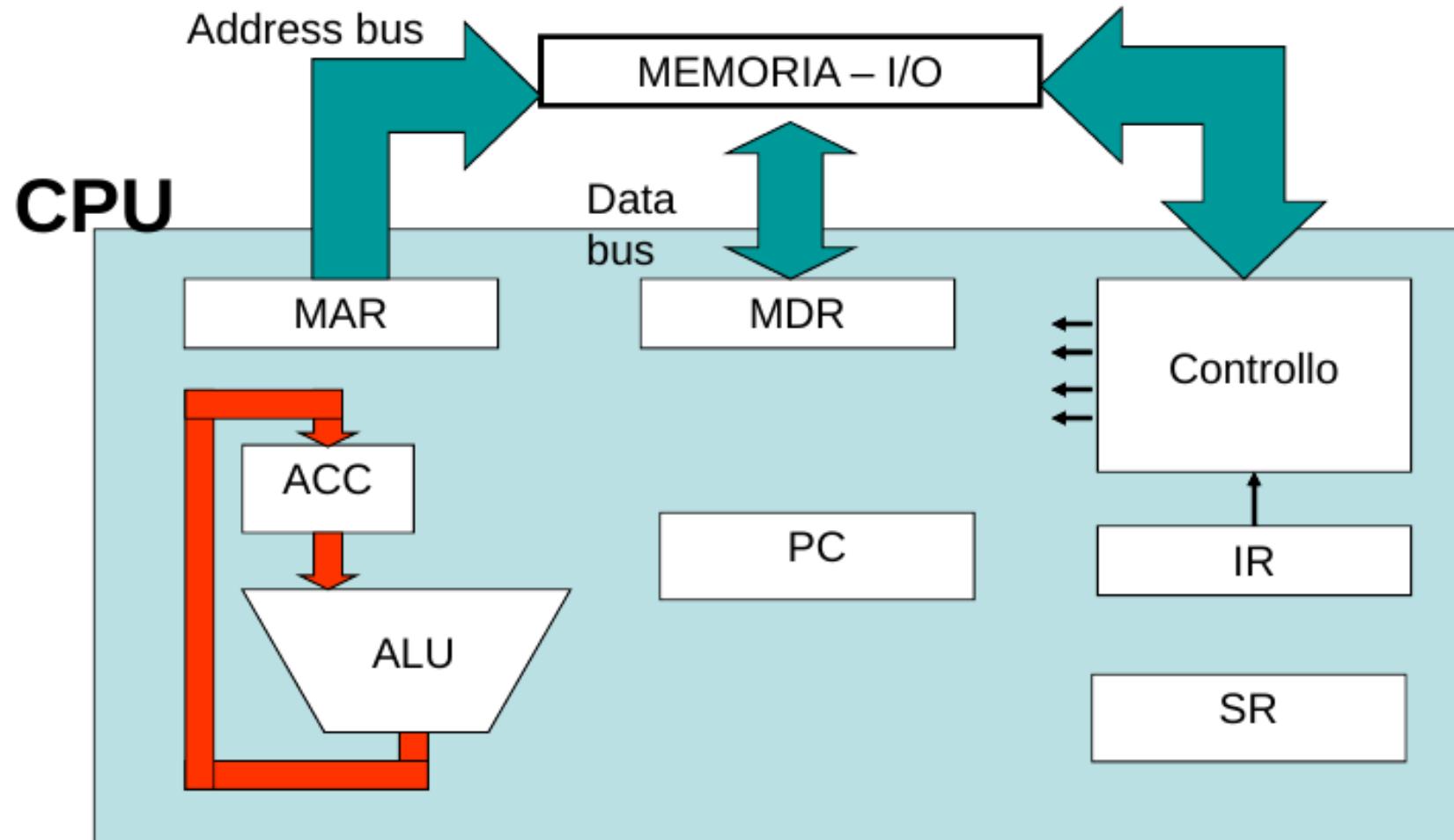
DIE
TI.
UNI
NA

Alcuni modelli architetturali

Modello architetturale di un processore: modello ad accumulatore



DIE
TI.
UNI
NA



Processori ad accumulatore



- In un processore ad accumulatore tutte le istruzioni aritmetiche, logiche e di confronto hanno un operando in memoria ed un altro (riferito implicitamente) contenuto in un registro interno del processore detto accumulatore
- Esempio: per realizzare $[x]+[y] \rightarrow z$ con una macchina ad accumulatore (es. Motorola 6809) occorre eseguire una sequenza di istruzioni del tipo

LDA x $[x] \rightarrow \text{accumulatore}$ (Istruzione LOAD)

ADDA y $[y]+[\text{accumulatore}] \rightarrow \text{accumulatore}$

STA z $[\text{accumulatore}] \rightarrow z$ (Istruzione STORE)

- Dimensione e velocità di esecuzione dei programmi penalizzate dal fatto che tutte le istruzioni devono indirizzare un dato in memoria

Esempio: $y=a*b+c*d$

LDA a $[a] \rightarrow \text{accumulatore}$ (Istruzione LOAD)

MULU b $[b]*[\text{accumulatore}] \rightarrow \text{accumulatore}$

STA t $[\text{accumulatore}] \rightarrow t$ (Istruzione STORE)

LDA c $[c] \rightarrow \text{accumulatore}$ (Istruzione LOAD)

MULU d $[d]*[\text{accumulatore}] \rightarrow \text{accumulatore}$

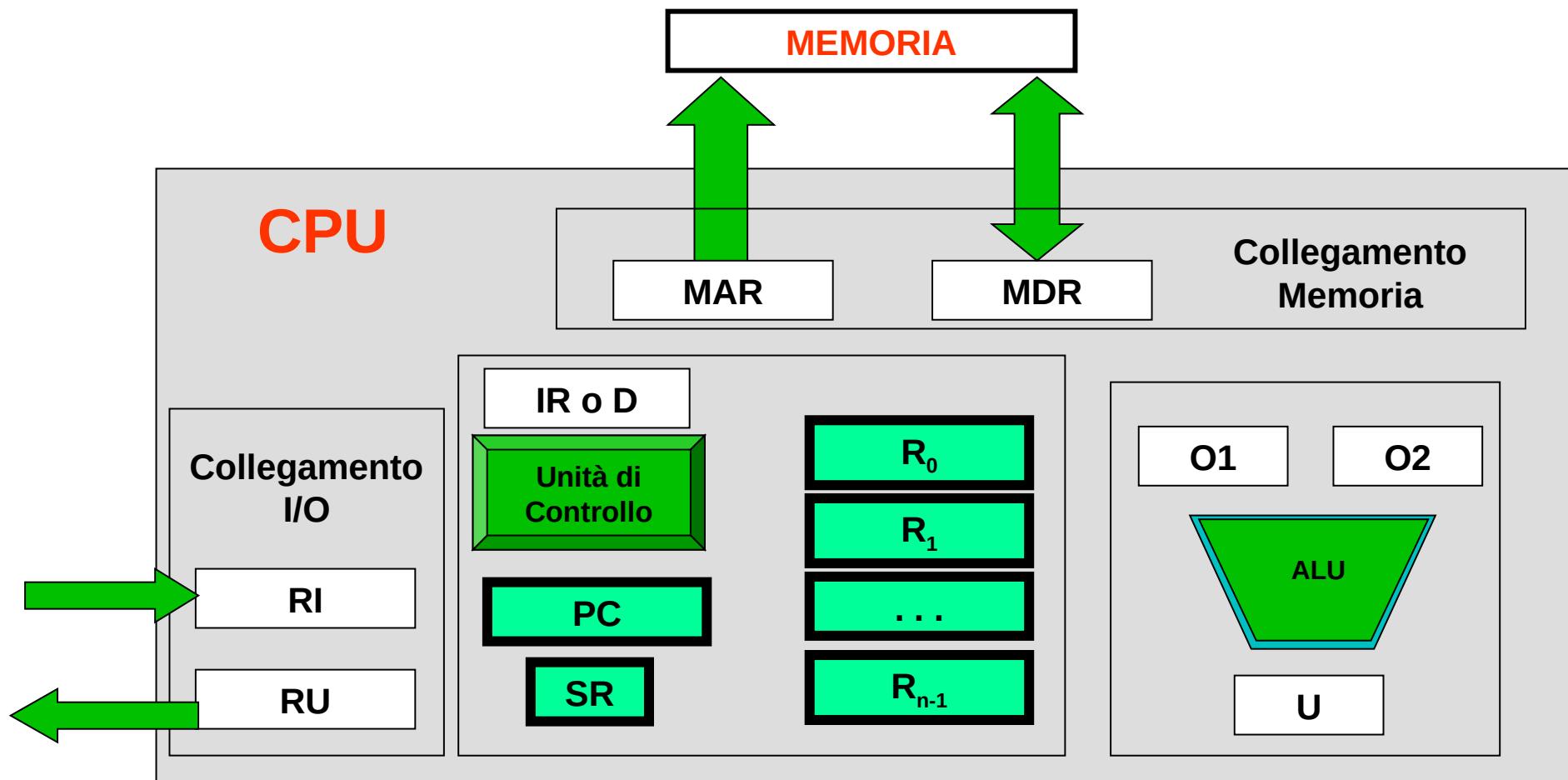
ADDA t $[t]+[\text{accumulatore}] \rightarrow \text{accumulatore}$

STA y $[\text{accumulatore}] \rightarrow y$ (Istruzione STORE)

Architetture a stack

- In un'architettura a stack si impiega una struttura di memoria organizzata a stack. Lo stack può essere realizzato all'interno della CPU e/o utilizzando memorie esterne.
- Gli operandi devono essere memorizzati sullo stack ed il risultato di una qualsiasi operazione (eseguita sullo stack) sostituisce successivamente gli operandi
- Di solito il valore in cima allo stack viene “duplicato” all'interno della CPU

Modello architetturale di un processore: modello a registri generali



Processore a registri generali



- Il processore dispone di un set di registri R0, R1,, RN-1 utilizzabili indifferentemente dal programmatore
- Le istruzioni che operano su registri sono più veloci di quelle che operano su locazioni di memoria
- Il programmatore può utilizzare i registri del processore per memorizzare i dati di uso più frequente
 - concetto di gerarchia di memorie
- Istruzioni con operandi registri:
 $[R0] + [R1] \rightarrow R1$
- Istruzioni con operandi memoria-registri:
 $[R0] + M[1000] \rightarrow R0$ *memory-to-register*
 $M[1000] + [R1] \rightarrow M[1000]$ *register-to-memory*

CPU: struttura interna ad 1 bus



DIE
TI.
UNI
NA

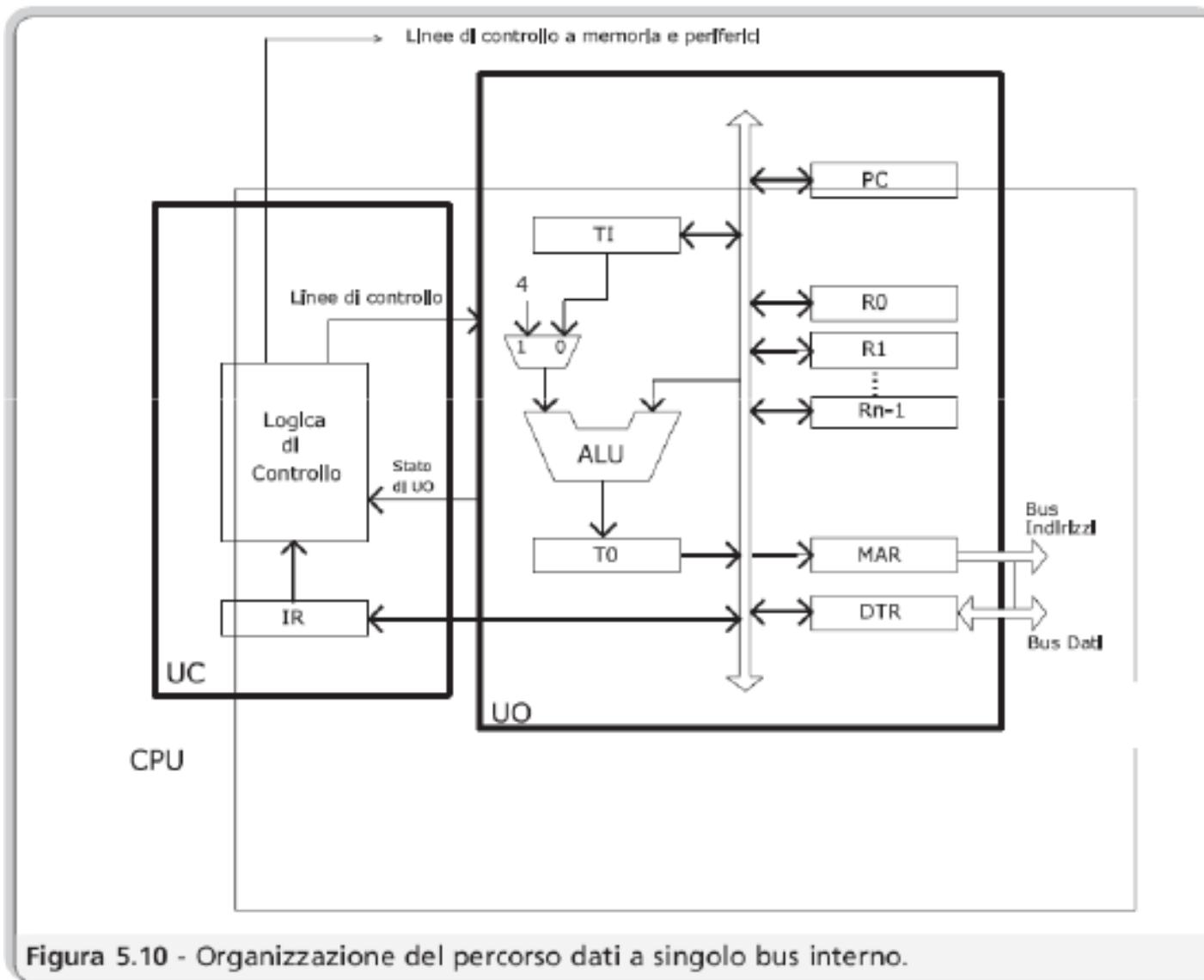


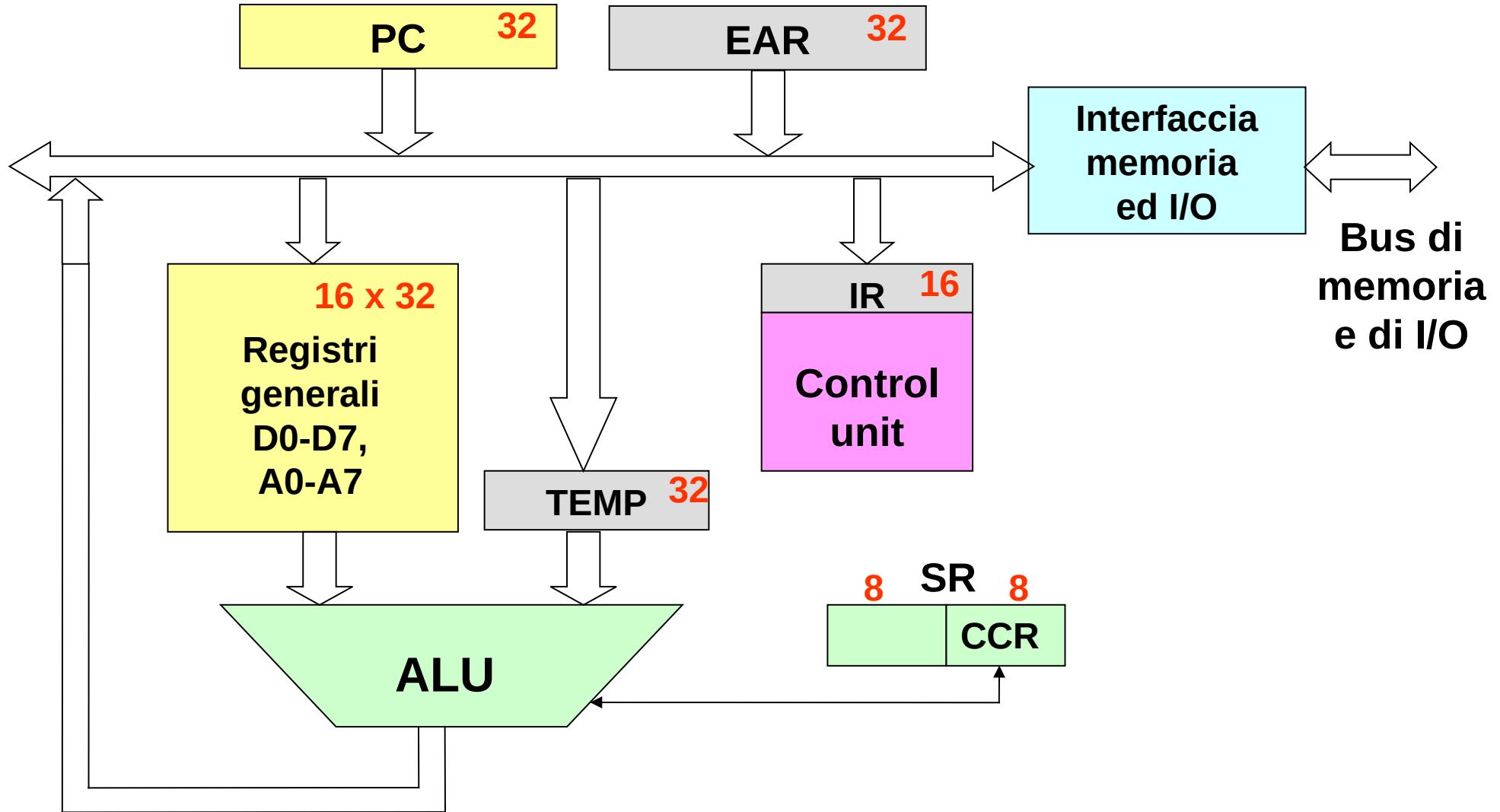
Figura 5.10 - Organizzazione del percorso dati a singolo bus interno.

Architetture pipelined

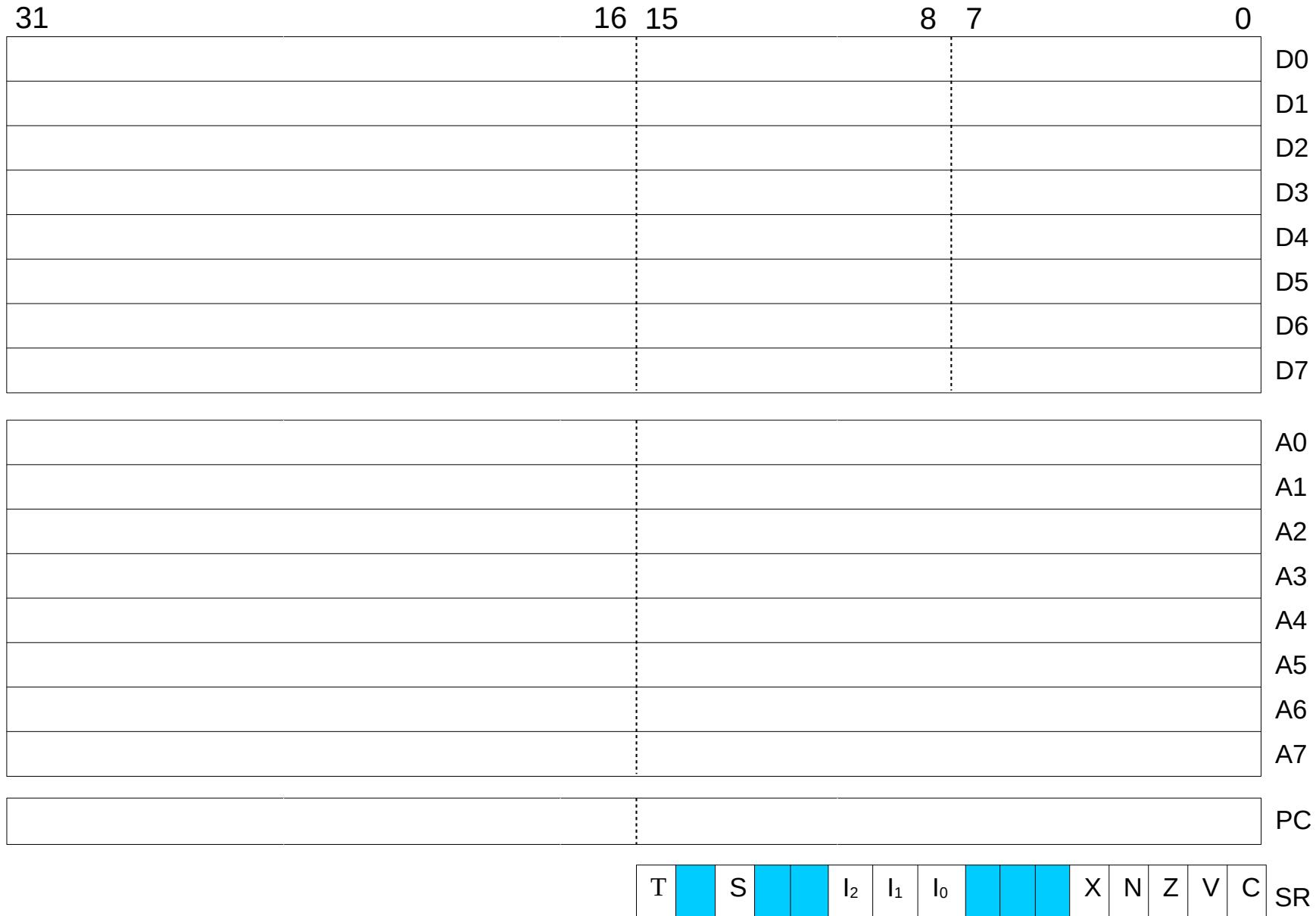
- Tutte le unità che eseguono le istruzioni sono tutte attive contemporaneamente (fetch, decodifica, esecuzione e scrittura)
- E' analoga ad una catena di montaggio!!!

Write			I1	I2	I3	I4	I5
Execute			I1	I2	I3	I4	I5
Decode		I1	I2	I3	I4	I5	
Fetch	I1	I2	I3	I4	I5		

Architettura del processore MC68000



Modello di programmazione del MC68000



Caratteristiche del processore MC68000

- Dati:
 - All'esterno:
parola di 16 bit (16 pin per i dati)
 - All'interno:
registri di 32 bit
- Indirizzi:
 - All'esterno:
24 bit (spazio di indirizzamento fisico $2^{24} = 16M$)
 - 512 pagine (2^9) da 32K (2^{15})
 - All'interno:
32 bit

Caratteristiche del processore MC68000

- Parallelismo della memoria:
 - Parole di 16 bit, ognuna costituita da due byte con indirizzi distinti (memoria byte addressable)
- Convenzioni della memoria:
 - Una parola deve essere allineata ad un indirizzo pari (even boundary)
 - Convenzione big-endian



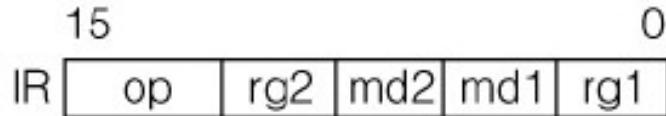
DIE
TI.
UNI
NA

Codifica delle istruzioni

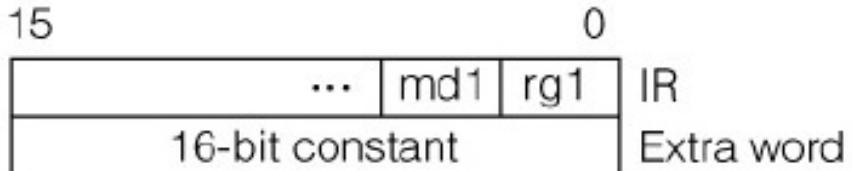
Codifica istruzioni MC68000



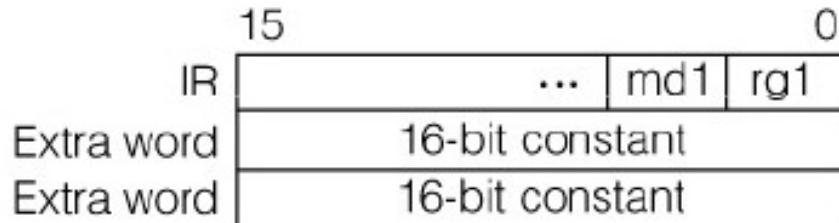
DIE
TI.
UNI
NA



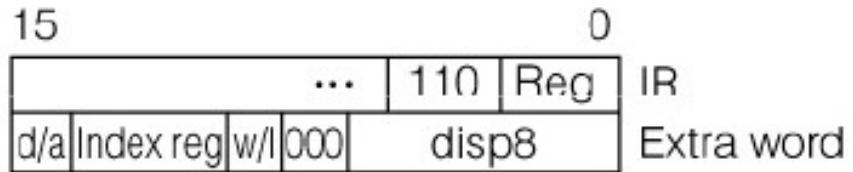
(a) A 1-word move instruction



(b) A 2-word instruction



(c) A 3-word instruction



(d) Instruction with indexed address

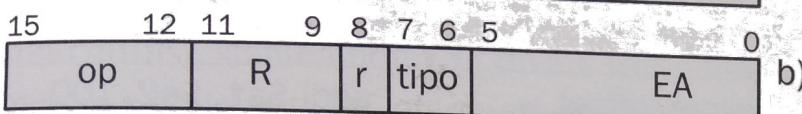
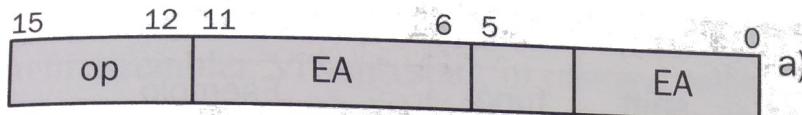
**Codifica a lunghezza variabile multipla di 2 byte:
opcode word + extra word(s)**

Copyright © 2004 Pearson Prentice Hall, Inc.

Codifica istruzioni MC68000



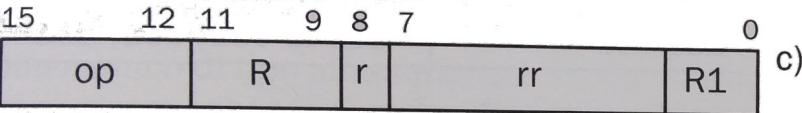
DIE
TI.
UNI
NA



op = codice operativo
R = indirizzo di registro D o A
tipo = tipo di operando (B, W, L)
spiazzamento = indirizzo di salto relativo
EA = Effective address
r = banco di registri A o D
cc = codice di condizione

b1)

R e r riuniti in un solo campo cc

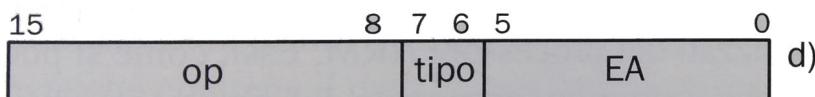


c1)

R e r riuniti in un solo campo cc

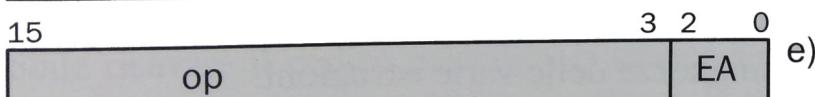
c2)

rr e R1 uniti in uno spiazzamento

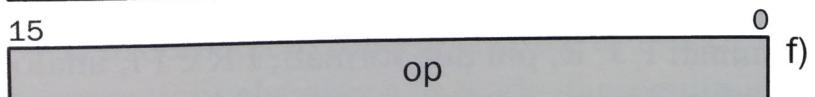


d1)

op e tipo uniti in tipo



f)



Codifica delle istruzioni di un processore in stile RISC

ESEMPIO: una CPU con istruzioni a lunghezza fissa di 32 bit

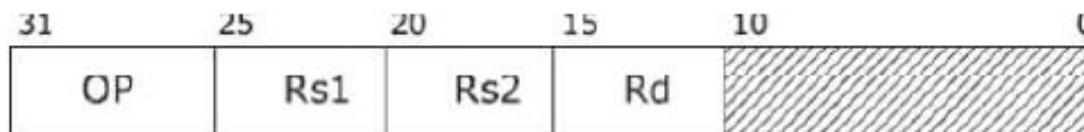


Figura 5.6 - Due possibili formati per la codifica delle istruzioni di macchina su parole di 32 bit. Il campo OP contiene il codice di operazione ed è sempre presente; gli altri campi dipendono dal codice di operazione stesso. Il primo formato prevede due operandi: un registro della CPU e un indirizzo della memoria. Il secondo formato prevede tre operandi, tutti registri della CPU; il campo tratteggiato è inutilizzato. La numerazione dei bit all'interno delle parole adotta la convenzione di assegnare il numero 0 al meno significativo.

LD R1, Var ; R1 \leftarrow M[Var]

ADD R1, R2, R3 ; R1 \leftarrow R2 + R3