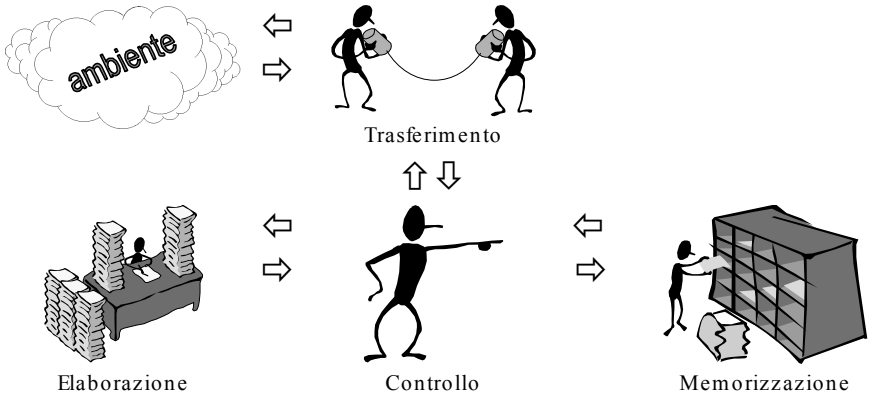
L'hardware dei sistemi di elaborazione (prima parte)

Fondamenti di Informatica A

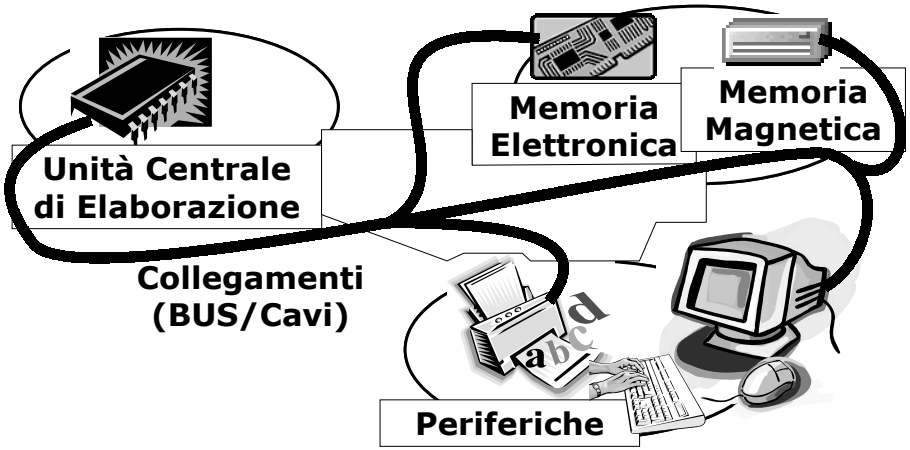
Ingegneria Gestionale
Università degli Studi di Brescia

Docente: Prof. Alfonso Gerevini

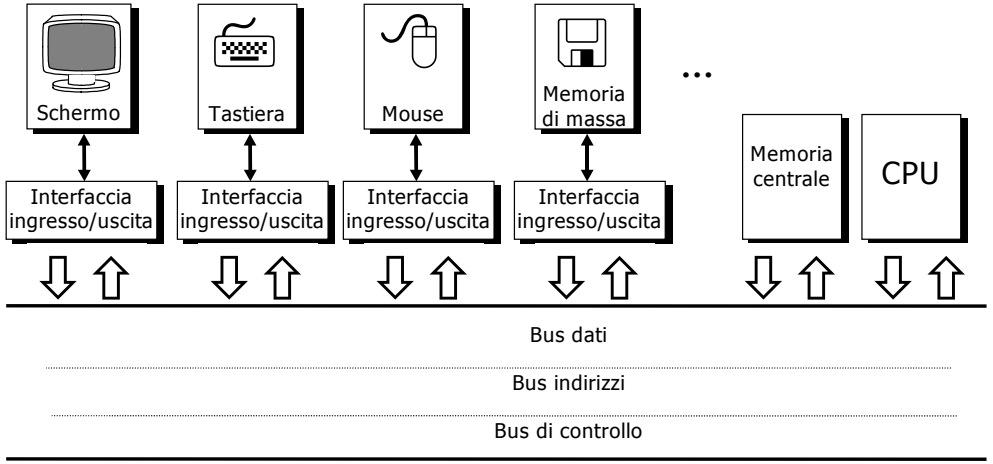
Funzionalità di un calcolatore



Il calcolatore: modello architetturale



Lo schema di riferimento



Caratteristiche del collegamento a BUS

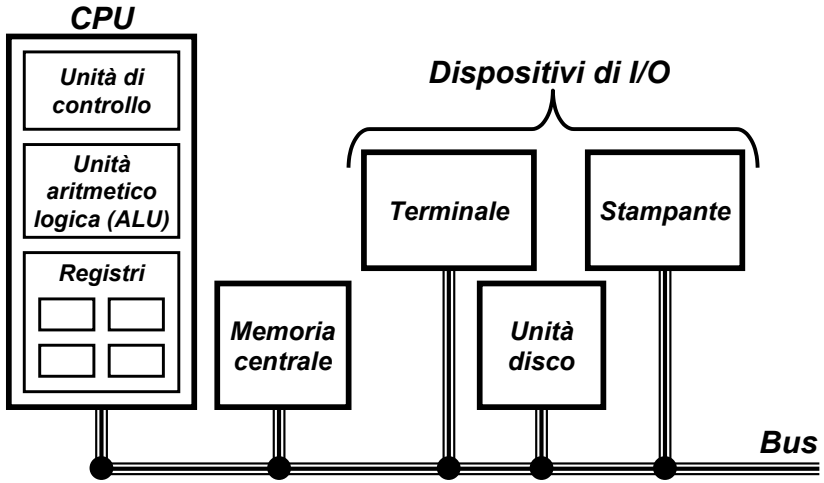
- **Semplicità**
  un'unica linea di connessione → costi ridotti di produzione
- **Estendibilità**
  aggiunta di nuovi dispositivi molto semplice
- **Standardizzabilità**
  regole per la comunicazione da parte di dispositivi diversi
- **Lentezza**
  utilizzo in mutua esclusione del bus
- **Limitatà capacità**
  al crescere del numero di dispositivi collegati
- **Sovraccarico del processore (CPU)**
  perchè funge da *master* sul controllo del bus

Architettura Von Newman

Esegue un programma sulla base dei seguenti principi

- Dati ed istruzioni memorizzati in una **memoria UNICA** (lettura e scrittura). L'unità centrale legge e scrive in memoria per **acquisire** le **istruzioni** da eseguire ed i relativi **operandi** e per **memorizzare** i risultati delle istruzioni eseguite
- Contenuti memoria indirizzati solo in base alla loro posizione (indipendentemente dal tipo di dato/istruzione)
- Le istruzioni vengono acquisite dalla memoria ed eseguite in modo **sequenziale**
- Le singole operazioni necessarie per l'esecuzione delle istruzioni sono **scandite da un orologio di sistema** (clock) che definisce l'evolvere del tempo all'interno della macchina

Organizzazione tipica di un calcolatore "bus oriented"



La memoria centrale

- La memoria può essere vista come un insieme di **celle** adiacenti
- Ogni cella è un **elemento binario**, capace cioè di assumere due stati possibili (0 e 1)
- Ogni cella contiene **un'unità di informazione** (un bit)
- Gruppi di celle formano **unità minime** di informazione **indirizzabili**: **byte**, **parole**, **doppie parole**, …
- Il concetto di **parola** dipende dal calcolatore: ad esempio in un calcolatore Intel una parola è formata da 2 byte (16 bit).

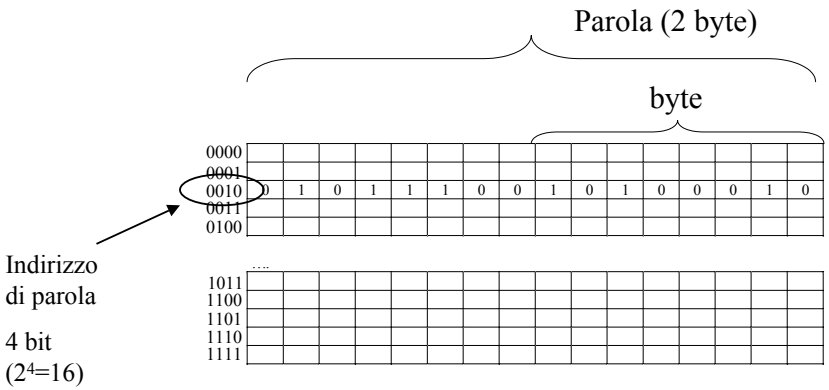
Indirizzi di memoria

- Ogni unità minima indirizzabile è identificata da un **indirizzo** (la prima ha convenzionalmente indirizzo 0)
- L'**ampiezza dell'indirizzo** è strettamente legata all'**ampiezza della memoria** ed al tipo di indirizzamento
- Esempio: con 32 bit di indirizzo si possono avere 2^{32} configurazioni di indirizzo diverse, ovvero indirizzare 4Giga (parole o byte)
- In generale: ***k* bit** di indirizzo → ***2^k* unità** indirizzabili

Unità di misura della memoria

- Imparare le potenze del 2!
- 1 byte = 8 bit = 2^3
- Kilo = K = 2^{10} = 1.024
- Mega = M = 2^{20} = 1.048.576
- Giga = G = 2^{30} = …
- Tera = T = 2^{40} = …

Esempio: memoria di 16 parole

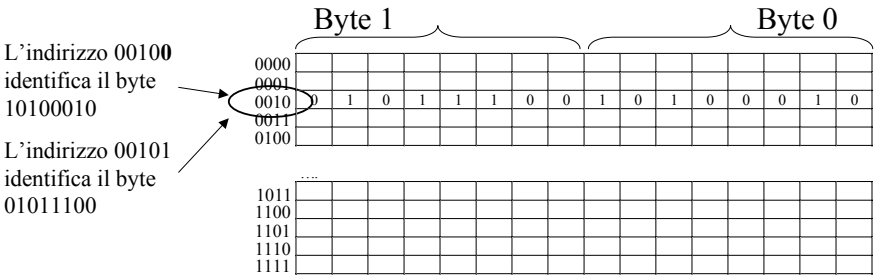


La parola il cui indirizzo è 0010 vale 0101110010100010

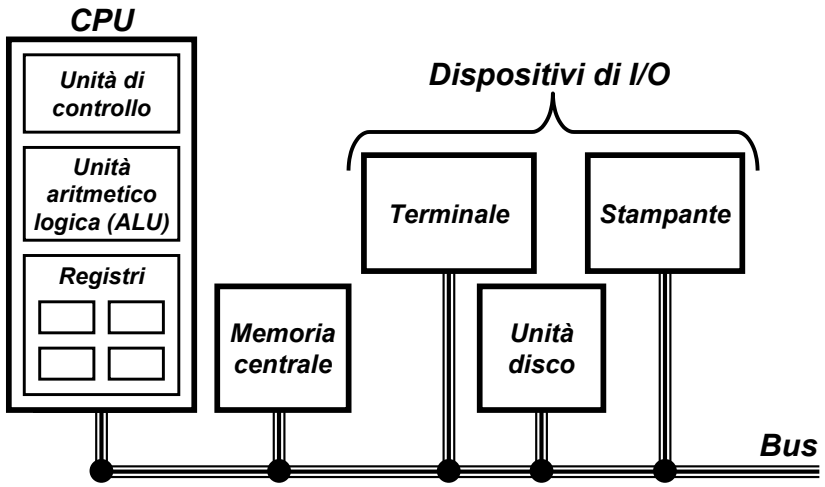
E se volessimo indirizzare ogni singolo byte?

Indirizzamento di un byte

- Per indirizzare ogni singolo byte abbiamo bisogno di un indirizzo più lungo
- Memoria di 16 parole = Memoria di 32 byte (se ogni parola è di 2 byte)
- Abbiamo bisogno di un indirizzo di 5 bit (2^5=32)



Organizzazione tipica di un calcolatore "bus oriented"



I registri della CPU

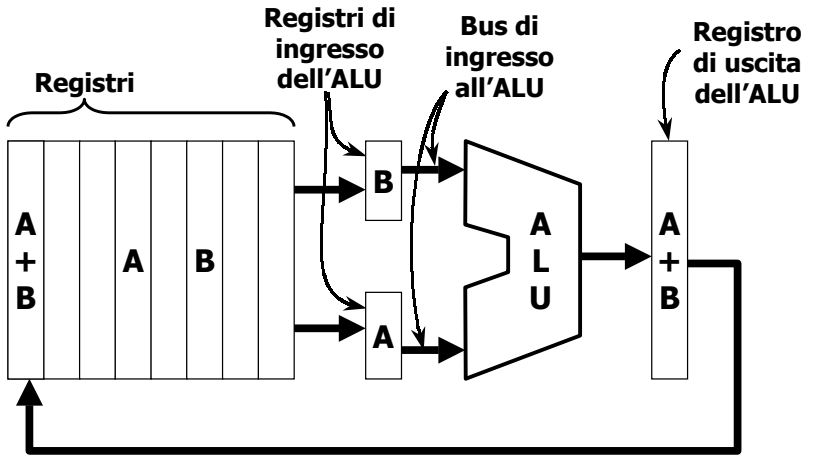
- Memoria a **breve termine** del calcolatore
- Sono **celle di memoria** utilizzate per immagazzinare le informazioni necessarie per l'esecuzione delle istruzioni:
- Ci sono **registri generici** (Registri per memorizzare gli operandi delle operazioni da eseguire) e **registri speciali**
- Il loro numero dipende dal tipo di CPU
- Gli operandi delle istruzioni aritmetico/logiche possono contenere un **indirizzo di registro**
- **Esempio**: CPU con 32 registri, occorrono 5 bit per identificare uno dei registri, nelle istruzioni ci saranno gruppi di 5 bit per gli operandi

Registri speciali

- ***PC***: *program counter*, registro contatore delle istruzioni - detto IP (*instruction pointer*) in alcune macchine (Intel)
- ***IR***: *instruction register*, registro delle istruzioni
- ***MAR***: *memory address register*, registro di indirizzamento della memoria – collegato al bus indirizzi
- ***MDR***: *memory data register*, registro dati di memoria – collegato al bus dati
- ***PSW***: *processor status word*, parola di stato del processore… detto anche FLAGS in alcune macchine (Intel)

Unità Aritmetico Logica

Esegue le operazioni necessarie per eseguire le istruzioni aritmetico-logiche

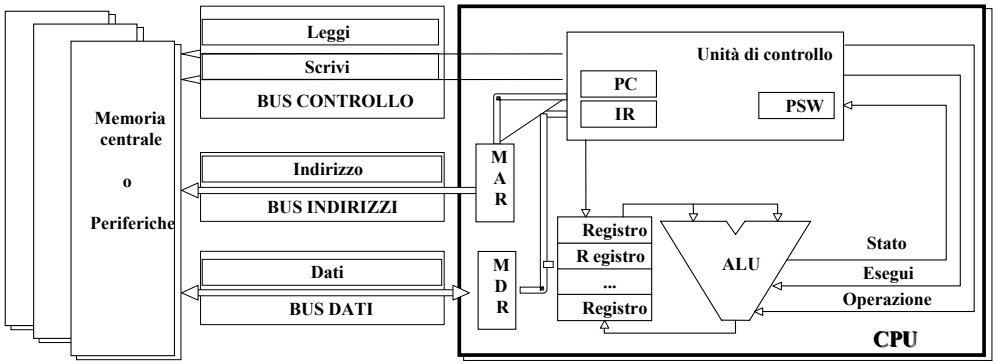


Unità di Controllo

E' l'Unità che:

- **Coordina** le varie unità del sistema nell'esecuzione dei programmi
- **Decodifica** le istruzioni in base al loro *codice operativo*
- **Manda** opportuni **comandi** (segnali di controllo) alla **ALU** (per la selezione dell'operazione), ai **registri** della CPU (per la lettura/scrittura), al **bus** per accedere alla memoria, e altri ancora…

La struttura della CPU



Ciclo macchina

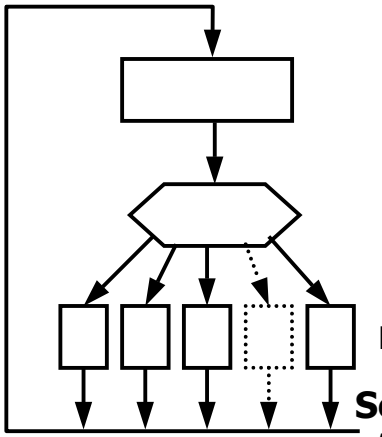
- La CPU può essere intesa come un dispositivo che **opera in modo ciclico**, ripetendo, per ogni programma i seguenti passi:
  - **Prelievo** dell'istruzione (***fetch***)
  - **Decodifica** dell'istruzione (***decode***)
  - **Esecuzione** dell'istruzione (***execute***)

Esecuzione delle istruzioni

Ciclo **Fetch–Decode–Execute** (**leggi–decodifica–esegui**):

- Prendi l'**istruzione corrente** (il contenuto di PC è messo in MAR) dalla memoria e mettila nel **registro istruzioni** (**IR**), dopo aver transitato da MDR.
- **Incrementa** il **program counter** (**PC**) in modo che contenga l'indirizzo dell'istruzione successiva.
- Determina il tipo dell'istruzione corrente (**decodifica**).
- Se l'istruzione usa una parola in memoria, determina dove si trova.
- Carica la parola, se necessario, in un registro della CPU.
- **Esegui** l'istruzione attivando le opportune componenti.
- Torna al punto 1 e inizia a eseguire l'istruzione successiva.

Ciclo macchina: Fetch–Decode–Execute



Tre tipologie di istruzioni

codice operativo	operandi

Formato generale delle istruzioni

- Istruzioni aritmetico-logiche (Elaborazione dati)
  - Somma, Sottrazione, Divisione, …
  - And, Or, Xor, …
  - Maggiore, Minore, Uguale, Minore o uguale, ...
- Controllo del flusso delle istruzioni
  - Sequenza
  - Salti condizionati (utili per operazioni di selezione/cicli)
  - Salti *in*condizionati (utili per operazioni di selezione/cicli)
- Trasferimento di informazione
  - Trasferimento dati e istruzioni tra CPU e memoria
  - Trasferimento dati e istruzioni tra CPU e dispositivi di ingresso/uscita (attraverso le relative interfacce)

Linguaggio macchina

- Il linguaggio macchina è il linguaggio per cui la CPU si comporta da **esecutore**
- Le istruzioni del linguaggio macchina sono caratterizzate da:
  - **Codice operativo** → tipo istruzione
  - **Operandi** → indirizzi dove recuperare i dati
- Ogni CPU è caratterizzata funzionalmente dal suo linguaggio macchina (**ISA – Instruction Set Architecture**)
- Esistono CPU di marca diversa con diversa struttura fisica che risultano **compatibili** (es. Intel e AMD)

Linguaggio Assembler

- Il linguaggio assembler è la ***rappresentazione simbolica*** della codifica binaria usata dal calcolatore (linguaggio macchina)
- L'assembler è più leggibile: utilizza simboli anziché bit
- I simboli del linguaggio assembler permettono di ***associare dei nomi alle configurazioni di bit più frequenti***, come codici operativi o riferimenti a registri
- L'assembler permette l'utilizzo di ***etichette*** per identificare certe parole di memoria che contengono dati o istruzioni
- ***Assemblatore***: traduce linguaggio assembler in linguaggio macchina

Esempio di linguaggio assembler: operazioni somma e sottrazione

```
add   r1, r3, r4    /* somma il contenuto di r3 col contenuto
                       di r4 e poni il risultato in r1 */
sub   r0, r1, r2    /* sottrai dal contenuto di r1 il contenuto di
                       r2 e poni il risultato in r0 */
```

Stato registri: prima

r0	4
r1	0
r2	5
r3	10
r4	2

Stato registri: dopo

r0	**7**
r1	**12**
r2	5
r3	10
r4	2

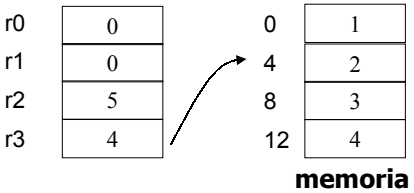
Lettura/Scrittura in memoria

- **Lettura di una parola dalla memoria**: operazione che rende disponibile all'uscita della memoria sul bus dati, e di conseguenza nel registro MDR della CPU, la parola presente all'indirizzo indicato
- **Scrittura di una parola in memoria**: l'operazione con cui il contenuto della parola di memoria indirizzata viene modificato, al fine di renderlo identico a quello presente nel registro MDR della CPU

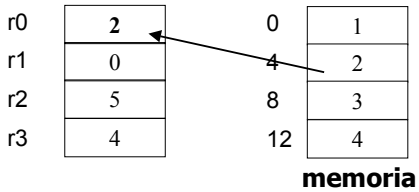
Operazione di lettura da memoria

```
load   r0, r3    /* carica in r0 il valore contenuto nella
                    locazione di memoria il cui indirizzo di
                    è in r3  */
```

Stato registri e memoria: prima



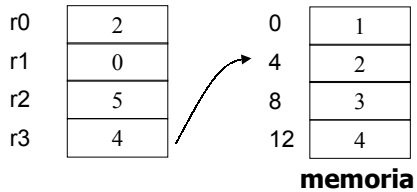
Stato registri e memoria: dopo



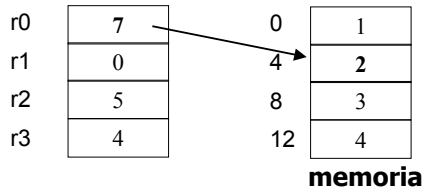
Operazione di scrittura in memoria

```
add    r0, r2, r0      /* r0 = r0 + r2 */
store  r0, r3   /* memorizza il valore presente in r0 nella
                   locazione di memoria il cui indirizzo è in r3 */
```

Stato registri e memoria: prima

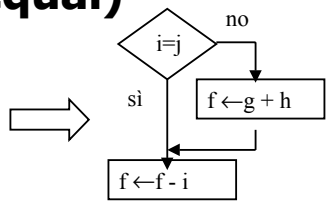


Stato registri e memoria: dopo



Esempio salto condizionato
(beq = branch if equal)

1. Se (i == j) allora vai al passo 3
2. f ← g + h;
3. f ← f – i;



Supponendo che f, g, h, i, j corrispondano ai registri r0, r1, r2, r3, r4, la traduzione potrebbe essere la seguente

```
      beq   r3, r4, L1    /* va a L1 se i è uguale a j */
      add   r0, r1, r2    /* f = g + h */
L1:   sub   r0, r0, r3    /* f = f - i */
```


Esempio salto non condizionato
(bne = branch if not equal)

Se (i != j) allora f ← g + h altrimenti f ← g - h

Supponendo che f, g, h, i, j corrispondano ai registri r0, r1, r2, r3, r4, la traduzione potrebbe essere la seguente:

```
      bne   r3, r4, allora   /* salto condizionato */
      sub   r0, r1, r2
      jump  esci             /* salto incondizionato */
allora: add r0, r1, r2
esci:
```


… oppure

Se (i == j) allora f ← g + h altrimenti f ← g - h

Supponendo che f, g, h, i, j corrispondano ai registri r0, r1, r2, r3, r4, la traduzione potrebbe essere la seguente

```
      bne   r3, r4, else
      add   r0, r1, r2
      jump  esci            /* salto incondizionato
Else: sub   r0, r1, r2
Esci:
```


Linguaggio macchina vs. Linguaggio assembler

Codice macchina di una procedura che calcola e stampa la somma dei quadrati degli interi fra 0 e 100

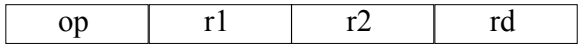
```
00100111101111011111111111100000
10101111101111110000000000010100
10101111101001000000000000100000
10101111101001010000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101111000000000000011000
00000001110011100000000000011001
00100101110010000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
00000000000000000111100000010010
00000011000011111100100000100001
00010100001000001111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
00000011111000000000000000001000
00000000000000000001000000100001
```

Codice assembler di una procedura che calcola e stampa la somma dei quadrati degli interi fra 0 e 100

```
addiu   $29, $29, -32
sw      $31, 20($29)
sw      $4, 32($29)
sw      $5, 36($29)
sw      $0, 24($29)
sw      $0, 28($29)
lw      $14, 28($29)
lw      $24, 24($29)
multu   $14, $14
addiu   $8, $14, 1
slti    $1, $8, 101
sw      $8, 28($29)
mflo    $15
addu    $25, $24, $15
bne     $1, $0, -9
sw      $25, 24($29)
lui     $4,4096
lw      $5, 24($29)
jal     1048812
addiu   $4, $4, 1072
lw      $31, 20($29)
addiu   $29, $29, 32
jr      $31
move    $2, $0
```

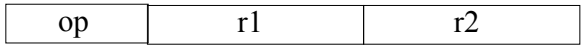
Esempi indicativi di formato delle istruzioni in linguaggio macchina

Istruzioni aritmetiche (`add, sub, ...`)



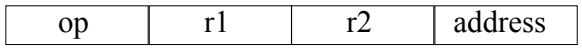
4+4+4+4 bit

Istruzioni di trasferimento (`load,store`)



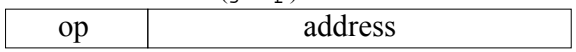
4+6+6 bit

Istruzioni salto condizionato (`beq,bne, ...`)



4+4+4+4 bit

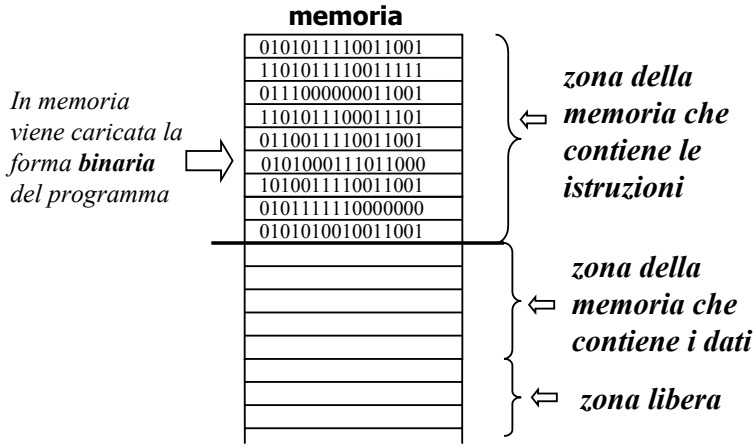
Salto incondizionato (`jump`)



4+12 bit

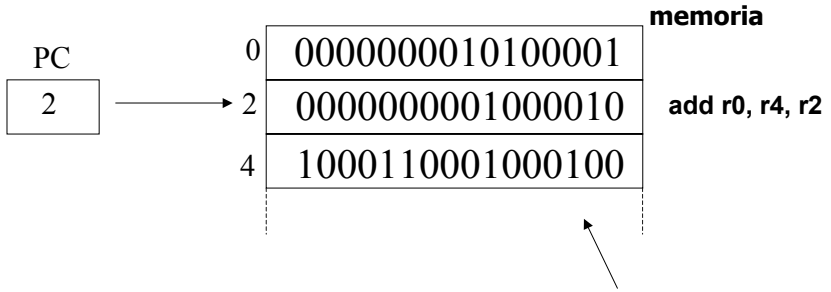
NB: in linguaggi reali possono esserci ulteriori campi

Programma in memoria



Esempio: esecuzione di una istruzione aritmetica

add r0, r4, r2 ➔ somma il contenuto del registro r_4 al contenuto del registro r_2 e memorizza il risultato in r_2



Passi per eseguire l'addizione

- Passo 1: Carica istruzione in IR

  IR ← 0000000001000010

  (si indica di solito con IR ← (PC))

  … e aggiorna PC

  PC ← PC + 2 (di 2 supponendo che le istruzioni siano lunghe 2 byte)

Dettagli sul passo 1

- L'indirizzo dell'istruzione contenuto nel PC viene trasferito in **MAR**
- Contemporaneamente viene attivato il **segnale 'leggi'** del bus di controllo
- La CPU **incrementa il valore in PC** e si pone in attesa della risposta dalla memoria
- La **memoria accede alla cella indirizzata** e ne pone il contenuto (la prossima istruzione da eseguire) sul bus dati
- Tale istruzione viene quindi trasferita in **MDR**
- L'unità di controllo emette gli opportuni comandi per **copiare il contenuto di MDR in IR**

Passi per eseguire l'addizione

- Passo 2: Decodifica istruzione in IR

  0000000001000010

  add r0 r4 r2

In pratica, l'unità di controllo

- legge il contenuto di IR
- effettua la decodifica dell'istruzione (scopre che è una **add**)
- e manda l'opportuno segnale di selezione dell'operazione da far compiere alla **ALU**

Passi per eseguire l'addizione (continua)

- Passo 3: Caricamento valori dei registri

  $R_A \leftarrow R_2$

  $R_B \leftarrow R_4$
- Passo 4: Somma (operazione con ALU)

  $R_C \leftarrow R_A + R_B$
- Passo 5: Memorizza risultato in R_2

  $R_2 \leftarrow R_C$

TOTALE = 5 passi

Esempio Istruzione di lettura da memoria

- Istruzione **load r4, r2**
- Carica nel registro R_4 il valore della parola presente all'indirizzo di memoria dato dal contenuto di R_2
- Il PC è stato aggiornato e punta all'istruzione successiva alla add

PC | 8 | → 8 | 1000000100000010 | **memoria**

Passi per eseguire una load

- Passo 1: Carica istruzione in IR e aggiorna PC
  IR ← (PC), PC ← PC + 2
- Passo 2: Decodifica istruzione in IR

$$\underbrace{1000}_{\text{load}}\ \underbrace{000100}_{\text{r4}}\ \underbrace{000010}_{\text{r2}}$$

- Passo 3: Copia contenuto di R_2 in MAR
  MAR ← R_2
- Passo 4: Accedi alla memoria e carica dato
  MDR ← (MAR)
- Passo 5: Copia contenuto di MDR in R_4
  R_4 ← MDR **5 passi in totale**

Passi per eseguire una store

Istruzione **store r4, r2**:

Memorizza il contenuto del registro R_4 nella parola il cui indirizzo è dato dal contenuto di R_2

- Passo 1: Carica istruzione in IR e aggiorna PC
  - IR ← (PC)
  - PC ← PC + 2
- Passo 2: Decodifica istruzione in IR
- Passo 3: Copia contenuto di R_2 in MAR
  - MAR ← R_2

Passi per eseguire una store (continua)

- Passo 4: Copia contenuto di R_4 in MDR
  - MDR ← R_4
- Passo 5: Accedi alla memoria e memorizza il dato
  - (MAR) ← MDR

TOTALE = ancora 5 passi

Passi per eseguire un salto

Istruzione **beq r1, r2, alfa**

Se il contenuto di **r1** è uguale al contenuto di **r2**, allora salta all'istruzione all'indirizzo **alfa**

- Passo 1: Carica istruzione in IR e aggiorna PC
  - IR ← (PC)
  - PC ← PC + 4
- Passo 2: Decodifica istruzione in IR
- Passo 3: Copia R_1 e R_2 nei registri usati dalla ALU
  - R_A ← R_1
  - R_B ← R_2

Passi per eseguire un salto (continua)

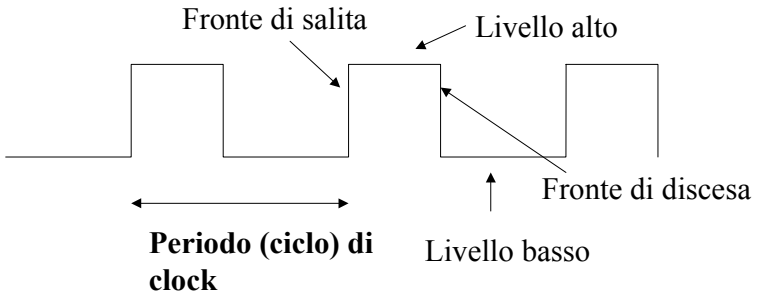
- Passo 4: Sottrai il contenuto di R_B dal contenuto di R_A
  - $R_C \leftarrow R_A - R_B$
- Passo 5: Confronta il contenuto di R_C col valore 0
- Passo 6: Se il risultato del confronto è positivo copia il valore dell'indirizzo Alfa in PC

TOTALE = 6 passi

Periodo di clock e frequenza di clock

- Le varie unità della CPU operano in *modo coordinato*
- Un **orologio (clock)** fornisce una cadenza temporale a cui tutte le attività elementari sono sincronizzate
- Un *segnale di clock* è un segnale che evolve con un **periodo** (tempo di ciclo) predeterminato e costante (intervallo di tempo fra 2 segnali di clock consecutivi)
- La **frequenza di clock** è l'inverso del periodo (numero di cicli di clock al secondo)

Un segnale di clock



Misure

- **Frequenza di clock**: normalmente misurata in **MHz**, dove 1 M = 10^6
- **1Hz** = 1 ciclo/secondo
- **Periodo di clock**: normalmente misurato in **ns** (*nanosecondi*, 10^{-9} secondi).
  Esempio: frequenza 500 MHz (500 milioni di cicli al secondo), periodo 2 ns
- I passi per compiere le operazioni viste sono in genere eseguiti in cicli di clock successivi: es. 5 passi (5 cicli di clock).
  Esempio: se un ciclo è di 2ns, occorrono 10 ns per svolgere un operazione di addizione

Prestazioni e tempo di CPU

Le prestazioni della CPU si determinano facendo riferimento al tempo di esecuzione della CPU

$$\text{Tempo di CPU relativo ad un programma} = \text{cicli di clock della CPU relativi al programma} \times \text{periodo di ciclo del clock}$$

$$\text{Tempo di CPU relativo ad un programma} = \frac{\text{cicli di clock della CPU relativi al programma}}{\text{frequenza di clock}}$$

Cicli di clock della CPU e Cicli di clock per istruzione

- **CPI** (clock per istruzione) indica il ***numero medio di cicli di clock per istruzione***, calcolato come la media del numero di cicli di clock che le diverse istruzioni richiedono
- Esempio: per una macchina che fa solo add, load, store, e beq (vedi esempi precendenti) il CPI è 5,25 (=(5+5+5+6)/4)
- Valgono quindi la seguenti relazioni:

cicli di clock della CPU = numero medio di istruzioni eseguite × CPI

(la media si intende sui dati)

5,25

Riassumendo…

tempo di CPU = **numero medio di istruzioni eseguite × CPI × durata del ciclo di clock**

o alternativamente…

$$\text{tempo di CPU} = \frac{\text{numero medio di istruzioni eseguite} \times \text{CPI}}{\text{frequenza di clock}}$$

$$T = (N \times CPI)/R$$

Evoluzione delle CPU Intel

CPU	Anno	Frequenza (MHz)	Dimensione registri / bus dati	Numero di transistor
8086	1978	4.77 — 12	8 / 16	29 000
80286	1982	8 — 16	16 / 16	134 000
80386	1986	16 — 33	32 / 32	275 000
80386 SX	1988	16 — 33	32 / 16	275 000
80486	1989	33 — 50	32 / 32	1 200 000
Pentium	1993	60 — 200	32 / 64	3 100 000
Pentium II	1997	233 — 400	32 / 64	7 500 000
Pentium III	1999	450 — 1133	32 / 64	24 000 000
Pentium 4	2000	1600 — 2000	32 / 64	42 000 000