Software - ISA

RISC vs CISC

RISC → Reduced Instruction Set Computing
- Si concentra sul <u>software</u>.
- Trasferimenti dati memoria dati tramite <u>registri</u> (non è possibile direttamente).
  memoria ⟷ registri ⟷ memoria

CISC → Complex Instruction Set Computing
- Si concentra sull' <u>hardware</u>.
- Trasferimento dati diretto memoria - memoria.
- Esecuzione più lenta di ogni istruzione.

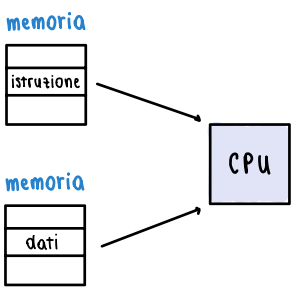
Computer

Componenti → Architettura di Von Neumann + Periferiche

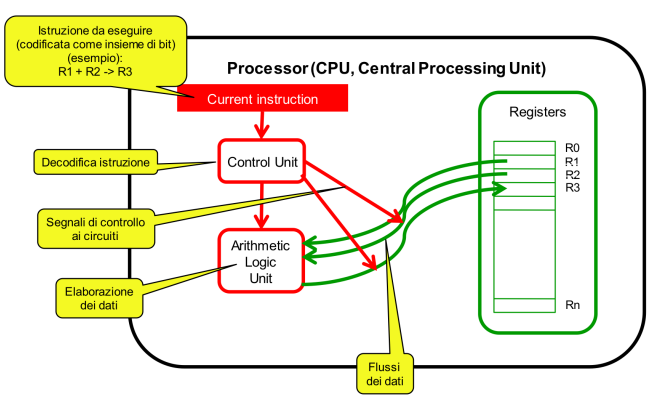
Architettura di Von Neumann:
- CPU
- Memoria
- Datapath

Periferiche:
- memorie di massa
- tastiera
- mouse
- monitor

ecc.



Definizione | Il processore (CPU → Central Processing Unit) si occupa delle istruzioni (codificate in un insieme di bit) decodificandolo e mandandolo i segnali ai circuiti e all'ALU il quale elabora i dati. L'output dell'ALU fluisce nella memoria.



memoria e registri

Definizione | La memoria è un insieme di celle che contengono i bit interpretabili in modi diversi (dati o istruzioni).

modalità di indirizzamento

Come prelevo i dati dai vari elementi di memoria

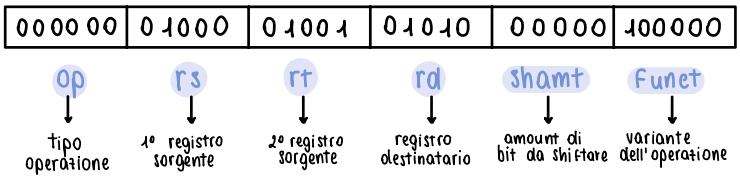
- registri di cpu → accade dentro il processore, sono pochi e veloci (coincide con il clock).
- memoria → grande, relativamente lenta. La memoria è un insieme di registri, chiamati di solito "celle" o "locazioni".
- memorie di massa

MIPS

Definizione | È un'architettura di tipo RISC, costituita da 32 registri da 32 bit con istruzioni da 32 bit.
Viene utilizzato l'assembly, un linguaggio sorgente per le istruzioni. L'assemblatore traduce da sorgente in linguaggio macchina.

formato delle istruzioni

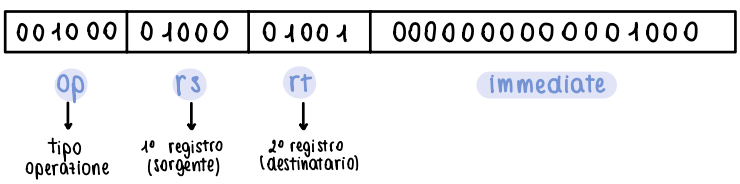
Istruzione R-Type



OSS 5 bit → 2^5 combinazioni → 32 registri

ES: add $rd, $rs, $rt
sll $rd, $rt, shamt

Istruzione I-Type

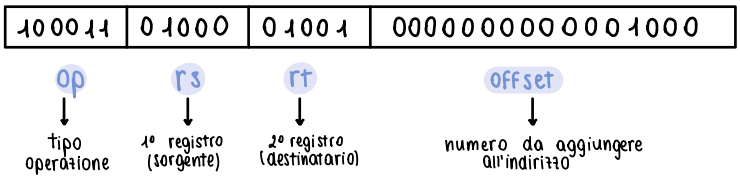


OSS A differenza dell'R-Type, posso scrivere immediatamente il valore del 2° operando. L'immediate è in CA2 e per effettuare somma si devono allineare i bit (sign extend).

ES addi $rt, $rs, imm

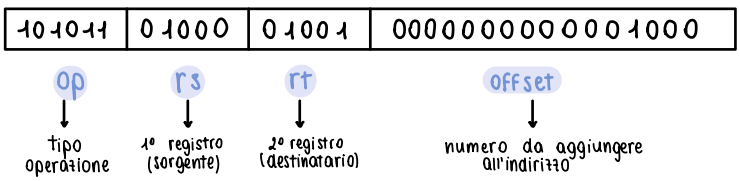
Due istruzioni I-Type permettono di caricare i contenuti di un registro in una locazione di memoria o viceversa ⟶ lw e sw

load word - memoria ⟶ registro



ES lw $rt, offset($rs)

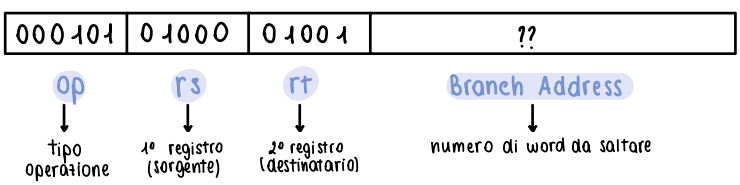
Store word - registro ⟶ memoria



ES sw $rt, offset($rs)

La seguente istruzione I-Type permette di fare salti condizionati.

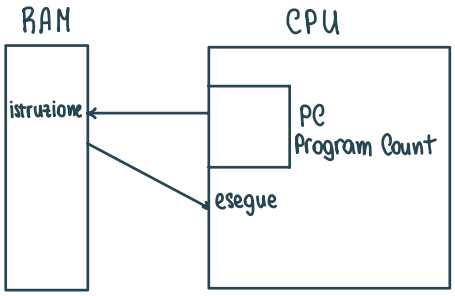
Branch - Salti condizionati



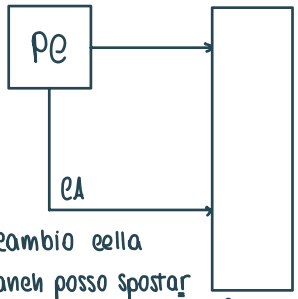
OSS Ampiezza di salto ⟶ $\pm 2^{15}$ rispetto al Program Counter
If rs ≠ rt ⟶ branch a etichetta
Confronta il contenuto dei due registri.

ES bne $rs, $rt, Exit

Attenzione!



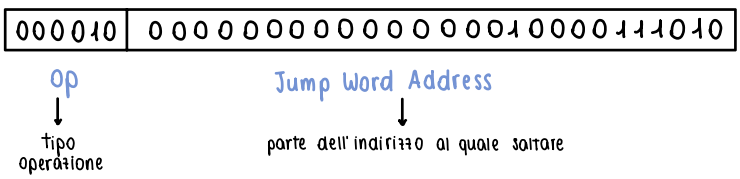
Saltare da un istruzione all'altra vuol dire cambiare Program Count.
Il Program Count ha già in se un indirizzo di cella che svolge una determinata istruzione. Per contare l'effettivo indirizzo, infatti, se si parte a contare da PC come indirizzo base, la formula di salto é PC + BranchAddr * 4.



cambio cella con branch posso spostarmi in alto o in basso rispetto al PC.

Se invece CA, ovvero l'istruzione corrente, é il registro base, si salta a
CA+4 + BranchAddr * 4
PC = CA+4 ⟶ 4 byte perché PC contiene già un'istruzione che corrisponde a 4 byte.

Istruzione J-Type



oss Indirizzo di arrivo ⟶ 4 bit più significativi del PC + Jump Word Address + Shift left di 2
Il jump ha limitatezza del salto, il jump register permette a saltare ovunque.