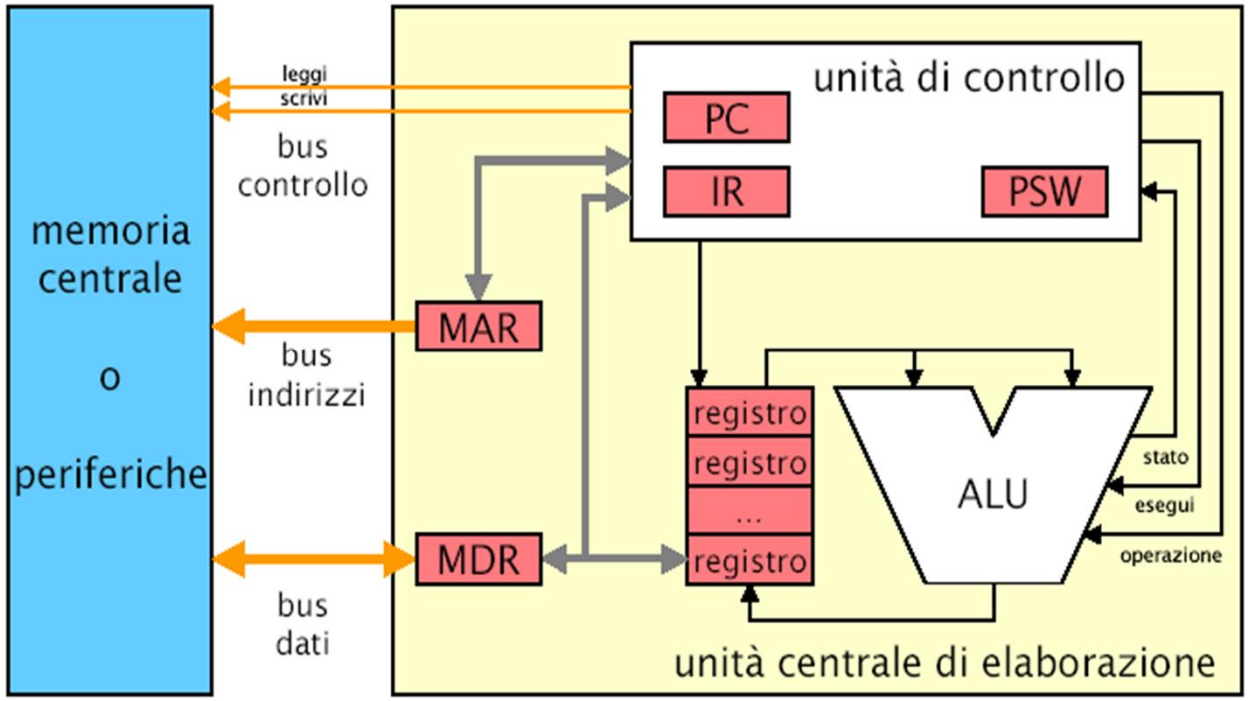
Architettura dei sistemi di elaborazione:
La CPU: Architettura (parte1)

La CPU – Architettura



- L'organizzazione interna di una CPU è caratterizzata dal data path, che è costituito da una serie di componenti, tra i quali l'ALU, i registri, e molti bus di comunicazione.
- I registri memorizzano i dati che vengono poi utilizzati e rielaborati dall'ALU che, a sua volta, riscrive i risultati delle elaborazioni nei registri stessi attraverso i bus di accesso ai registri.

La CPU – Registri

•MBR (Data Register) - Memorizza i dati provenienti e diretti alla memoria.

•MAR (Address Register) - Memorizza gli indirizzi da comunicare alla memoria.

•PC (Program Counter o Contatore di Programma) – Memorizza l'indirizzo della istruzione da eseguire.

•IR (Instruction Register) - Memorizza il codice dell'istruzione da eseguire.

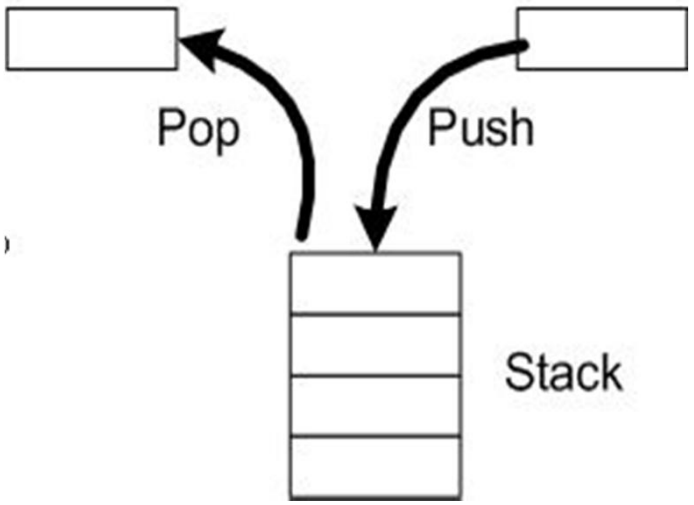
•ACC (Accumulatore) – Registro che mmagazzina dati in ingresso e in uscita dalla ALU.

La CPU – Lo stack

Lo stack (pila) è una struttura dati gestita in memoria secondo una politica LIFO (last in – first out).

La possibilità di gestire uno stack in memoria è comune a molte architetture. In tal caso il repertorio delle istruzioni prevede le operazioni

•PUSH – inserimento di un dato nello stack
•POP – prelievo di un dato dallo stack



La CPU – Lo stack

Per la gestione dello stack servono due registri:

•BP (Base Pointer) – punta entro lo stack come se si trattasse di una normale area di memoria

Es. LOAD R,[BP+8] carica in R il contenuto della cella dello stack il cui indirizzo è dato dalla somma del contenuto di BP con 8

•SP (Stack Pointer) – punta sempre alla testa dello stack e viene aggiornato ad ogni esecuzione di una operazione PUSH o POP; non può essere impiegato come indicato sopra per BP

La CPU – Lo stack

Lo stack viene utilizzato per:

•salvare temporaneamente il contenuto dei registri prima dell'esecuzione di un sottoprogramma (procedura)

•depositare i parametri passati a un sottoprogramma

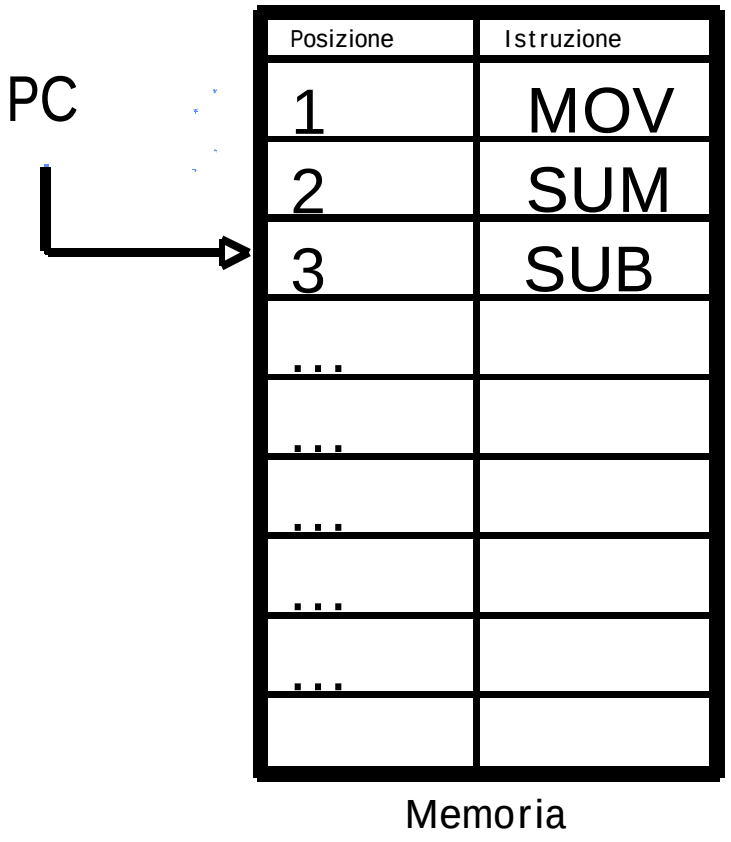
Es. una istruzione di alto livello del tipo y = f(x1, x2, …, xn);

viene tradotta dal compilatore in una sequenza di PUSH dei parametri, a partire dall'ultimo, seguita dall'istruzione di chiamata a f; il valore y viene reso attraverso un registro predefinito (sempre lo stesso, per una data architettura).
Per accedere ai parametri nello stack, il sottoprogramma usa il BP.

La CPU – Registi IR e PC

Il registro istrizioni contiene volta per volta le istruzioni che il processore deve eseguire.

Uno speciale registro contatore, spesso detto Program Counter o Prossima Istruzione, si occuperà di controllare l'ordine con il quale vengono eseguite le istruzioni del programma, tenendo presente quale istruzione il processore sta eseguendo in quel determinato momento.



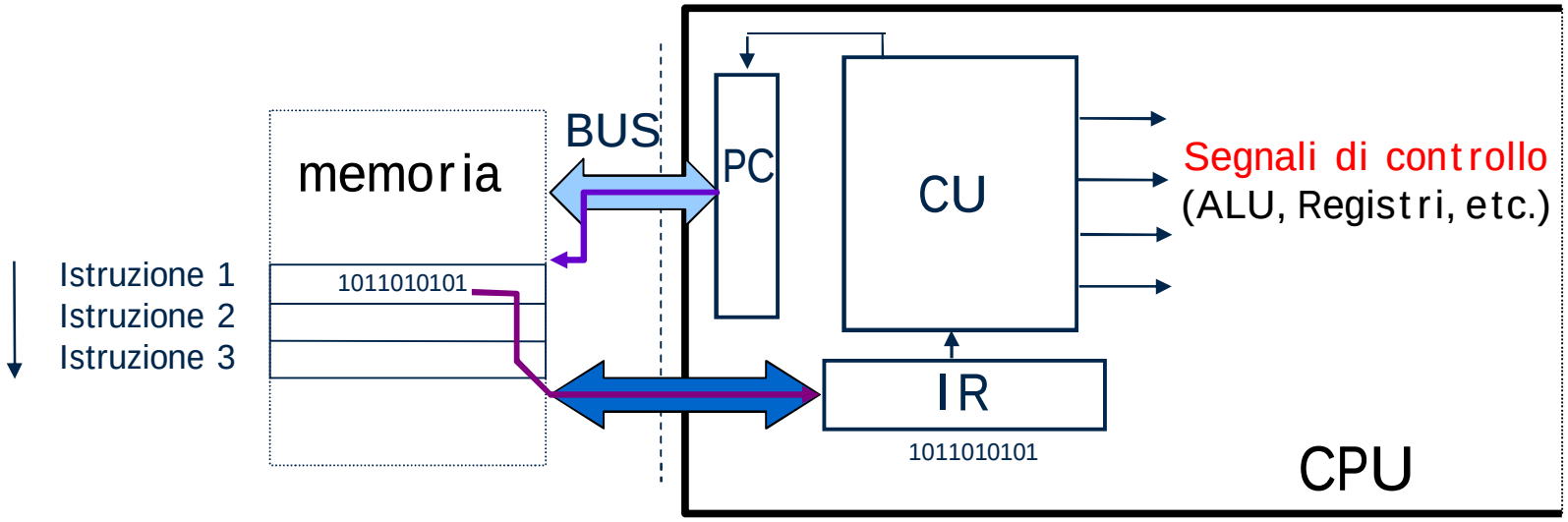
Unità di Controllo

- Il coordinamento tra le varie componenti all'interno della CPU è svolto dall'unità di controllo
- ogni componente esegue solo le azioni che gli vengono richieste dall'unità di controllo
- il controllo consiste nel coordinamento dell'esecuzione temporale delle operazioni sia internamente all'unità di elaborazione sia negli altri elementi funzionali

Unità di Controllo

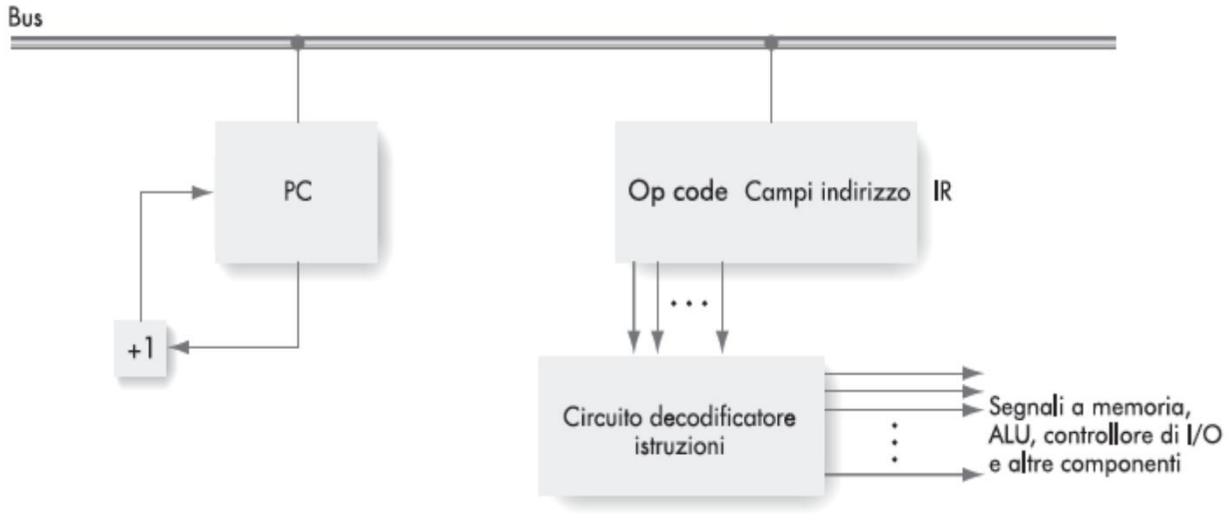
Esegue le istruzioni della CPU prelevandole dalla memoria

- La prossima istruzione da prelevare è individuata dall'indirizzo di memoria presente nel registro *Program Counter* (PC)
- L'istruzione attualmente in esecuzione è memorizzata nel registro istruzione (IR)
- Dopo aver prelevato un'istruzione, il PC viene incrementato, per fare riferimento all'istruzione successiva



Unità di Controllo

- Le connessioni attraverso il bus non sono attive tutte contemporaneamente.
- L'*unità di controllo* seleziona i cammini che risultano attivi in un determinato istante.

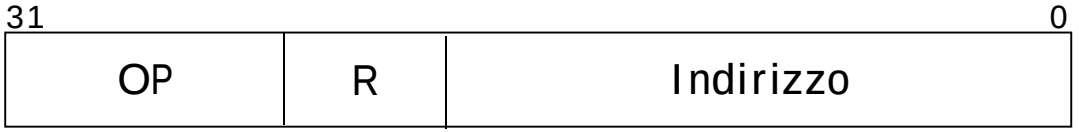


Elaborazione Istruzioni

- Le istruzioni di un programma corrispondono ad operazioni elementari di elaborazione
  - operazioni aritmetiche
  - operazioni relazionali (confronto tra dati)
  - operazioni su caratteri e valori di verità
  - altre operazioni numeriche
- L'elaboratore sa svolgere poche tipologie di operazioni elementari ma in modo molto efficiente
- Può eseguire decine o centinaia di milioni di istruzioni al secondo

Codifica Istruzioni

Un possibile formato per la codifica di istruzioni su parole di 32 bit:



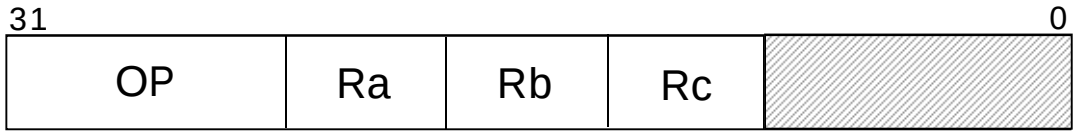
Es. LOAD R,Var

LOAD è il codice mnemonico per l'operazione di trasferimento dati dalla memoria. R è il registro destinazione. Var è il nome simbolico di una posizione di memoria (indirizzo). Quindi l'istruzione equivale all'istruzione RTL

$R \leftarrow M[Var]$

Codifica Istruzioni

Un altro possibile formato per la codifica di istruzioni su parole di 32 bit:



Es. ADD R1,R2,R3

ADD è il codice mnemonico per l'operazione di somma. R1 è il registro destinazione. R2 e R3 sono i registri che contengono gli addendi. Quindi l'istruzione equivale all'istruzione RTL

R1 ← R2+R3

Set di istruzioni

- Ogni processore viene progettato con un set di istruzioni specifico denominato ISA (Instruction Set Architecture o Instruction Set)
- Ogni istruzione è implementata da un microprogramma.
- Ogni istruzione dell'ISA è contraddistinta da un numero specifico, denominato Operation Code (Op. Code)
- Ogni istruzione necessita di un numero preciso e definito di parametri che, assieme all'Op.Code, determinano la lunghezza dell'istruzione (in byte).

Set di istruzioni - caratteristiche

COMPLETEZZA:

- Deve essere possibile valutare qualunque funzione che sia calcolabile con una disponibilità ragionevole di memoria.

EFFICIENZA:

- Istruzioni usate frequentemente devono essere eseguite rapidamente.
- Il controllo sull'efficienza deve essere misurato sul codice generato da compilatori.
- Fornire primitive, non soluzioni.

Set di istruzioni - caratteristiche

REGOLARITA':

- Le istruzioni devono comportarsi in modo omogeneo rispetto ai modi di indirizzamento.
- Quando ogni istruzione consente tutti i modi di indirizzamento possibili su tutti gli operandi: ORTOGONALITA'

COMPATIBILITA':

- Compatibilità sorgente: sono compatibili i codici mnemonici del linguaggio assembler.
- Compatibilità binaria: sono compatibili i codici macchina.
- Il codice macchina deve essere eseguibile su processori successivi della stessa famiglia.
- 8080 $\Rightarrow$ 8086 $\Rightarrow$ 80186 $\Rightarrow$ ... $\Rightarrow$ Pentium $\Rightarrow$ ...
- PDP11 $\Rightarrow$ VAX

Set di istruzioni - Tipi

TRASFERIMENTO DATI

- MOVE — Trasferisce dati tra registri
- LOAD — Trasf. dati dalla memoria
- STORE — Trasf. dati alla memoria
- EXCHANGE — Scambia dati
- SET/RESET — Pone ad 1/0
- PUSH/POP — Gestione stack

OPERAZIONI ARITMETICHE

- ADD/SUB — Somma/differenza
- MULT/DIV — Moltiplicazione/divisione
- ABS — Valore assoluto
- NEG — Cambio segno
- INC/DEC — Incrementa/decrementa

Set di istruzioni - Tipi

OPERAZIONI LOGICHE

- AND
- OR
- NOT
- XOR

OPERAZIONI PER LA MANIPOLAZIONE DEI BIT

- SHIFT traslazione
- ROTATE rotazione

CONTROLLO DEL FLUSSO

- JUMP salto incondizionato
- JUMP COND salto condizionato
- CALL (COND) salto a sottoprogramma
- RET (COND) uscita da sottoprogramma

Set di istruzioni - Tipi

CONTROLLO CPU

- HALT blocco operazioni
- WAIT/HOLD blocco operazioni con ripresa condizionata
- NOP non svolge operazioni

INGRESSO E USCITA

- INPUT (READ) trasferimento dati da I/O verso memoria o registro
- OUTPUT (WRITE) trasferimento dati verso porta di I/O

CISC

- Complex instruction set computer (CISC) indica un'architettura per microprocessori formata da un set di istruzioni contenente istruzioni in grado di eseguire operazioni complesse come la lettura di un dato in memoria, la sua modifica e il suo salvataggio direttamente in memoria tramite una singola istruzione.
- Ogni singola istruzione ha un data path a più cicli.
  - Il data path è il percorso dei dati all'interno del Processore, attraverso l'attuale istruzione, e i suoi cicli sono scanditi dal clock della CPU.

CISC

Criteri di progettazione:

- le istruzioni non hanno dimensione fissa
- il campo del codice di operazione può occupare più o meno bit
- numero ampio di formati: il codice di operazione non identifica

univocamente il formato delle istruzioni, e da un formato all'altro i campi che identificano uguali entità possono occupare differenti posizioni

Esempi di processori CISC:
Intel x86, Zylog Z80, Motorola 68000

Il mondo dei PC desktop e laptop è dominato dalla famiglia Intel x86 che continua ad adottare lo stile CISC per mantenere la compatibilità con le applicazioni.

RISC

- RISC, acronimo dell'inglese Reduced Instruction Set Computer, indica una filosofia di progettazione di architetture per microprocessori formate da un set di istruzioni contenente istruzioni in grado di eseguire operazioni semplici che possono essere eseguite in tempi simili.
- I più comuni processori RISC sono:AVR, PIC, ARM, DEC Alpha, PA-RISC, SPARC, MIPS e POWER.

RISC

- Un'architettura RISC (Reducted Instruction Set Code), possiede un data path a singolo passo.
- Il set di istruzioni di una architettura RISC è limitato, contiene istruzioni di lunghezza costante (con un numero di operandi fisso), con fase di Decode breve e senza microprogrammi da eseguire nel processore: ogni istruzione è eseguita direttamente in hardware con pochi cicli di clock.
- Un'elaborazione RISC appare nettamente più veloce (almeno di un ordine 10).
- Un'istruzione CISC - con molti passi nel data path - equivale a numerose istruzioni RISC con data path singolo.
- I programmi per ISA RISC sono molto più lunghi di analoghi programmi per ISA CISC.

CISC o RISC?

Prima degli anni 80 si riteneva che repertori estesi di istruzioni fossero preferibili, per facilitare la costruzione di compilatori.

L'uso efficiente della memoria centrale (allora costosa e poco performante) era il primo obiettivo di qualunque soluzione architettonica. *Repertorio esteso significa programmi più corti che occupano meno memoria ma che richiedono una unità di controllo più complessa.* La memoria di controllo era più veloce di quella centrale, per cui portare funzionalità nella prima avrebbe comunque migliorato le prestazioni della macchina.

Con l'avvento delle *memorie a semiconduttore* (che rimpiazzarono quelle a nuclei magnetici) e delle *memorie cache*, e la constatazione che *l'80% delle istruzioni eseguite corrisponde al solo 20% del repertorio esteso*, presero piede i microprocessori RISC.

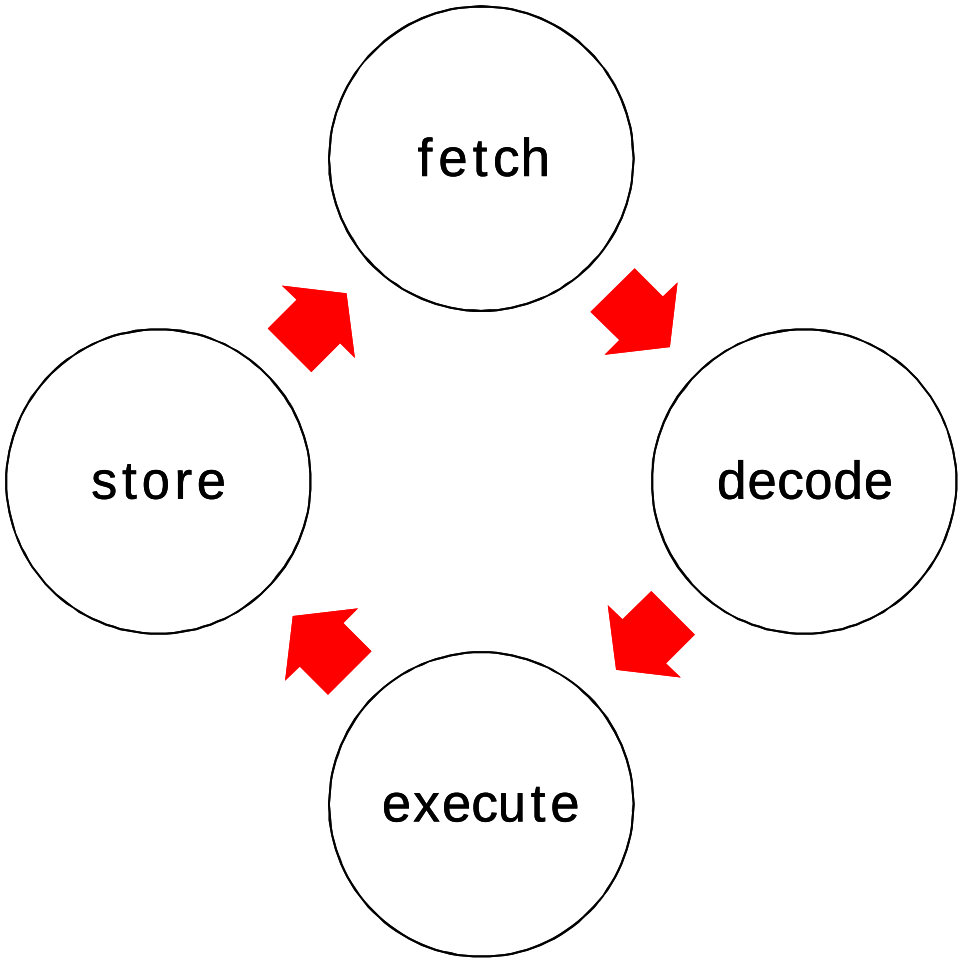
Il Ciclo di Elaborazione

Ogni istruzione contiene in modo esplicito o implicito l'indicazione dell'istruzione successiva.

- *Modo esplicito*: il codice dell'istruzione contiene l'indirizzo della cella di memoria dove è contenuta l'istruzione successiva.
- *Modo implicito*: l'indirizzo della istruzione successiva è contenuto in un registro interno alla CPU.

La sequenza delle istruzioni è definita dal programma.

Il Ciclo di Elaborazione



Il Ciclo di Elaborazione

- Fetch
  - l'unità di controllo pone sul bus degli indirizzi il valore del Program Counter e legge dalla memoria il codice dell'istruzione da eseguire. La fase di FETCH è eguale per tutte le istruzioni che ovviamente si differenziano nella fase di ESECUZIONE. Al termine della fase di ESECUZIONE viene eseguita la fase di FETCH della istruzione successiva.
- Decode
  - l'unità di controllo decodifica l'istruzione e legge i parametri (Operand Fetch) che vengono memorizzati nei Registri.
- Execute
  - viene avviato il microprogramma relativo all'Op.Code. La frequenza in base alla quale vengono eseguiti i microprogrammi è regolata dal clock di CPU (frequenza del Microprocessore).
- Store
  - al termine della fase di Execute gli eventuali risultati, posti nei Registri, vengono scritti sul Bus dall'UC, o verso la Memoria, o verso l'I/O.

Il Ciclo di Elaborazione – Fetch / Decode

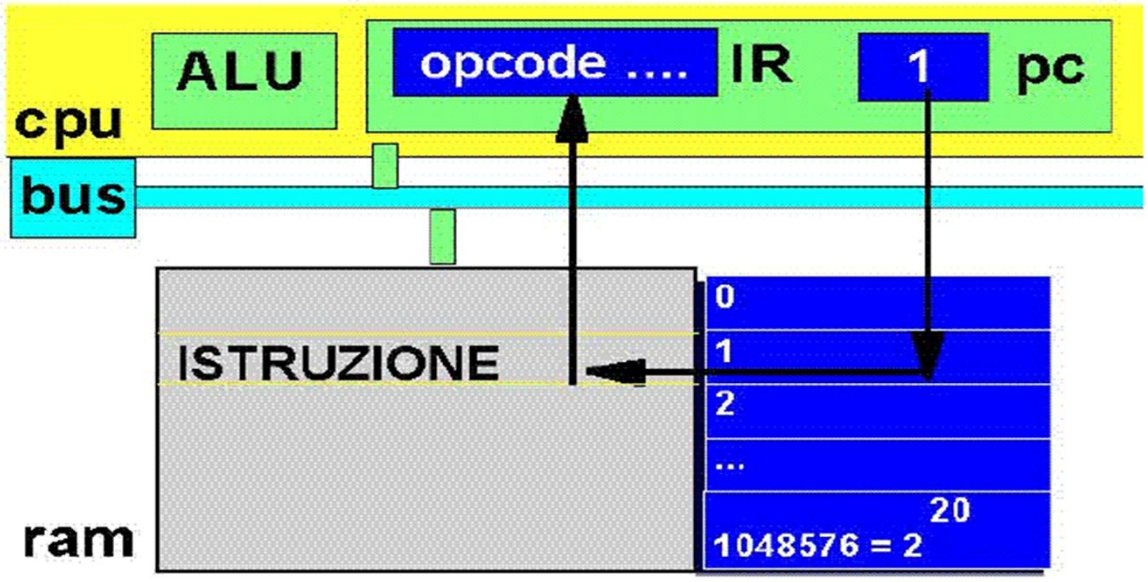
La fase di FETCH consiste nelle seguenti operazioni di trasferimento dati tra registri:

- MAR ← PC
- MBR ← M(MAR) ; lettura in memoria dell'istruzione
- IR ← MBR ; codice istruzione all'unità di controllo

Nella fase di DECODE:

- si individua il tipo dell'operazione e gli operandi (dati) usati
- si trasferiscono i dati nei registri opportuni

Il Ciclo di Elaborazione – Fetch / Decode



Il Ciclo di Elaborazione - Execute

- Prima di iniziare la fase di esecuzione, si aggiorna il PC con l'indirizzo dell'istruzione successiva.
- *Attenzione! l'istruzione successiva non è per forza nella successiva cella di memoria (vedi salti e chiamate a sottoprogrammi).*
- *ciascuna azione viene richiesta al componente opportuno*