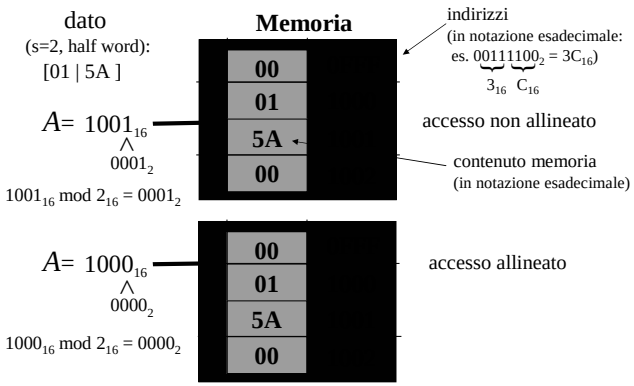
Accesso a memoria

- A questo livello di astrazione, la memoria viene vista come un array di byte
- Per ogni richiesta di un dato ad un certo indirizzo, la CPU ottiene un numero di byte determinato dall'ampiezza del Data Bus e dalle modalità di **allineamento** dei dati in memoria
- Un accesso ad un dato (composto da *s* byte) all'indirizzo *A* si dice **allineato** se *A* mod *s* = 0

Formato del dato	*Allineato (ultimi 3 bit di A)*	*Non allineato*
byte	0,1,2,3,4,5,6,7	mai
half word (2 byte)	0,2,4,6	1,3,5,7
word (4 byte)	0,4	1,2,3,5,6,7
double word (8 byte)	0	1,2,3,4,5,6,7

Accesso a memoria



Accesso a memoria

- Memoria non allineata (Intel x86)
  - Accessi facilitati ed ottimizzati
  - Maggiore complessità di progettazione
- Memoria allineata (SPARC, Motorola 68k)
  - Accesso più complesso a dati non allineati
  - Richiede compilatori efficienti per preservare capacità di accesso adeguata a dati non allineati

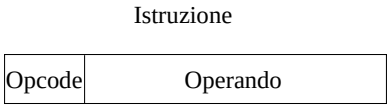
Modalità di indirizzamento

- Immediato
- Diretto
- Indiretto
- Registro
- Registro indiretto
- Spiazzamento (Displacement)
- Pila

Indirizzamento Immediato

- L'operando è parte dell'istruzione (nel campo indirizzo)
- ad esempio: **`ADD 5`**
  - somma 5 al contenuto dell'accumulatore
  - 5 è l'operando
- Non fa uso di alcun riferimento alla memoria da dove prelevare il dato
- Veloce (non deve accedere alla memoria)
- Limitato in rango (dipende dal numero di bit del campo utilizzato per specificare il dato immediato)

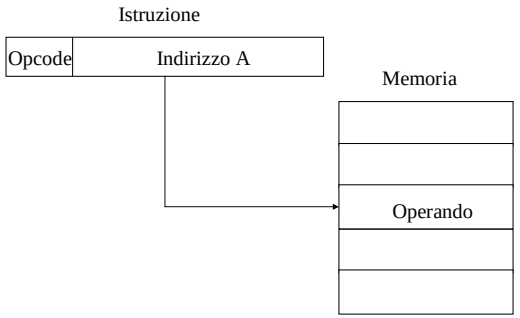
Indirizzamento Immediato (diagramma)



Indirizzamento Diretto

- Il campo indirizzo contiene l'indirizzo dell'operando
- Indirizzo effettivo (EA) = campo indirizzo A
- ad esempio: **`ADD A`**
  - somma il contenuto della cella di memoria con indirizzo A all'accumulatore
  - preleva l'operando dalla cella di memoria di indirizzo A
- Singolo accesso (riferimento) di memoria per prelevare l'operando
- Non necessita di effettuare calcoli per stabilire l'EA dell'operando
- Limitato nello spazio di indirizzamento dal numero di bit adottati per il campo indirizzo

Indirizzamento Diretto (diagramma)



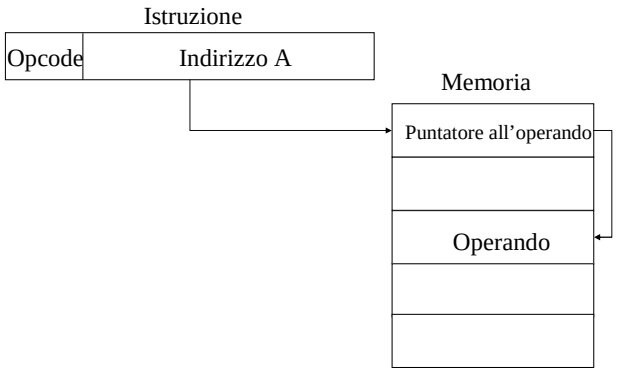
Indirizzamento Indiretto (1)

- Il campo indirizzo contiene A, l'indirizzo di una cella di memoria il cui contenuto corrisponde all'indirizzo dell'operando
- EA = [A] (l'indirizzo effettivo è il contenuto della cella di indirizzo A, identificato dalla notazione [A])
- ad esempio: **`ADD [A]`**
  - somma il contenuto della cella il cui indirizzo è contenuto nella cella di indirizzo A (cioè [A]) all'accumulatore

Indirizzamento Indiretto (2)

- Possibilità di indirizzare un ampio spazio di memoria
- 2^n celle, dove n = lunghezza della parola ("dimensione" di una cella di memoria)
- Si può applicare a cascata (ma non vale la pena…)
  - ad esempio: EA = [[[A]]]
- Accessi multipli alla memoria per prelevare l'operando
- Più lento dei precedenti

Indirizzamento Indiretto (diagramma)



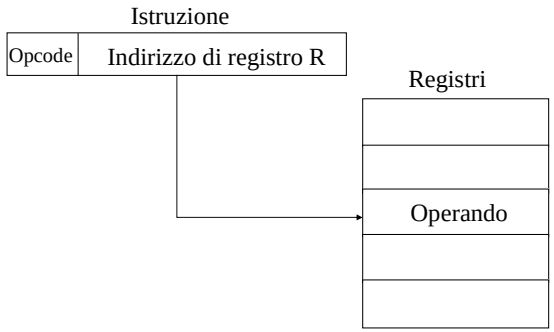
Indirizzamento Registro (1)

- L'operando è contenuto in un registro il cui nome si trova nel campo indirizzo
- EA = R
- Numero limitato di registri
- Il campo indirizzo è molto piccolo (pochi registri)
  - Istruzioni più corte
  - Fetch delle istruzioni più veloce

Indirizzamento Registro (2)

- Nessun accesso alla memoria
- Esecuzione della istruzione molto veloce
- Spazio di indirizzamento molto limitato (scegliere uno fra i registri disponibili)
- Avere più registri aiuta le prestazioni
  - Responsabilità di uso efficiente spostata al programmatore in assembler, e al compilatore
  - In C e C++, la dichiarazione **`register int a;`** mappa direttamente la variabile **`a`** in un registro

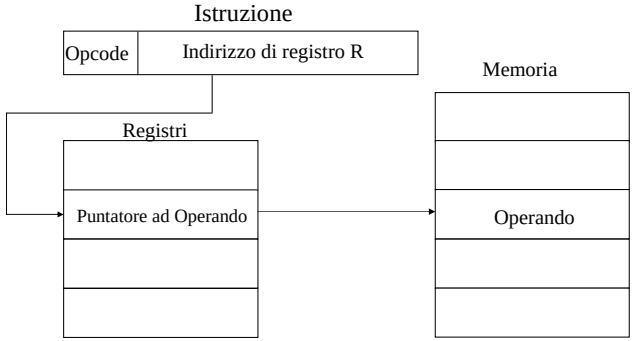
Indirizzamento Registro (diagramma)



Indirizzamento Registro Indiretto

- Analogo all'indirizzamento indiretto
- EA = [R]
- L'operand è nella cella di memoria puntata dal contenuto del registro R
- Spazio di indirizzamento grande (2^n)
- Un accesso di memoria in meno rispetto all'indirizzamento indiretto

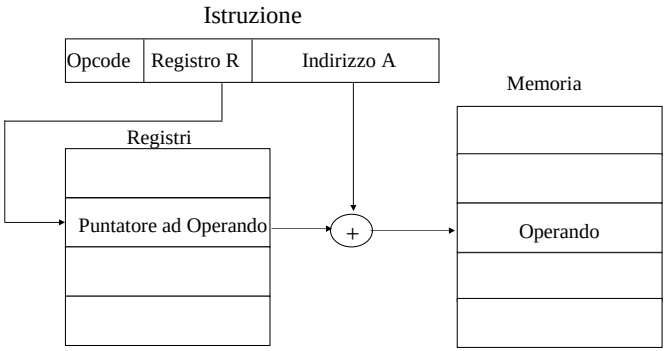
Indirizzamento Registro Indiretto



Indirizzamento con Spiazzamento

- EA = A + [R]
- Il campo indirizzo contiene due valori
  - A = indirizzo base
  - R = registero che contiene il valore di spiazzamento
  - o vice versa (dipende dall'uso)

Indirizzamento con Spiazzamento



Indirizzamento Relativo

- E' una versione dell'indirizzamento con spiazzamento
- R = PC, Program counter
- EA = A + [PC]
- Preleva l'operando a "distanza" A celle dalla locazione corrente puntata da PC

Indirizzamento a Registro-Base

- E' una versione dell'indirizzamento con spiazzamento
- Lo spiazzamento è specificato da A
- R contiene il puntatore all'indirizzo base
- R può essere esplicito o implicito (esempio: registri di segmento in 80x86)

Indirizzamento Indicizzato

- E' una versione dell'indirizzamento con spiazzamento
- A = indirizzo base
- R = spiazzamento
- EA = A + [R]
- Ideale per accedere agli elementi di un array
  - EA = A + [R]
  - R++

Combinazioni

- Alcune macchine ammettono l'utilizzo combinato dell'indirizzamento indiretto e dell'indicizzazione:
  - Postindicizzazione
    - EA = [A] + [R]
    - Utile per accedere ad un blocco di dati di formato fisso
  - Preindicizzazione
    - EA = [A + [R]]
    - Usato per la costruzione di tabelle di salto a più vie

Indirizzamento a Pila

- Operando (implicito) sulla cima della pila
- Esempio già visto
- Registro SP (*stack pointer*) punta alla cima della pila, oppure, se i due elementi in cima alla pila si possono trovare in registri della CPU, punta al terzo elemento della pila
- Riferimenti a locazioni della pila di fatto sono indirizzi a registro indiretto

Formato delle istruzioni

- Come vengono codificate in binario le istruzioni ?
- Bisogna specificare:
  - di quale operazione si tratta (**opcode**);
  - il tipo/i di indirizzamento di memoria usato/i (**address mode(s)**)
- Formati:
  - variabile | op. e #operandi | addrs. specifier 1 | … | addrs. specifier n |
  - fisso | op. | addrs. field 1 | addrs. field 2 | addrs. field 3 |
  - ibrido | op. | addrs. field 1 |
    | op. | addrs. field 1 | addrs. field 2 |
    ...

    stessa istruzione a lunghezza multipla, ma ognuna in formato fisso

Formato delle istruzioni

- Esempi:
  - variabile (VAX) :

    ```
    addl3 r1,737(r2),(r3)
    ```

  - fisso (DLX)

    ```
    add r1,r2,r3
    ```

  - ibrido (80x86)

fadd	operandi e risultato in stack
fadd st(i)	un operando in registro i sotto la stack ...
fadd mem32	un operando è una cella di memoria a 32 bit ...