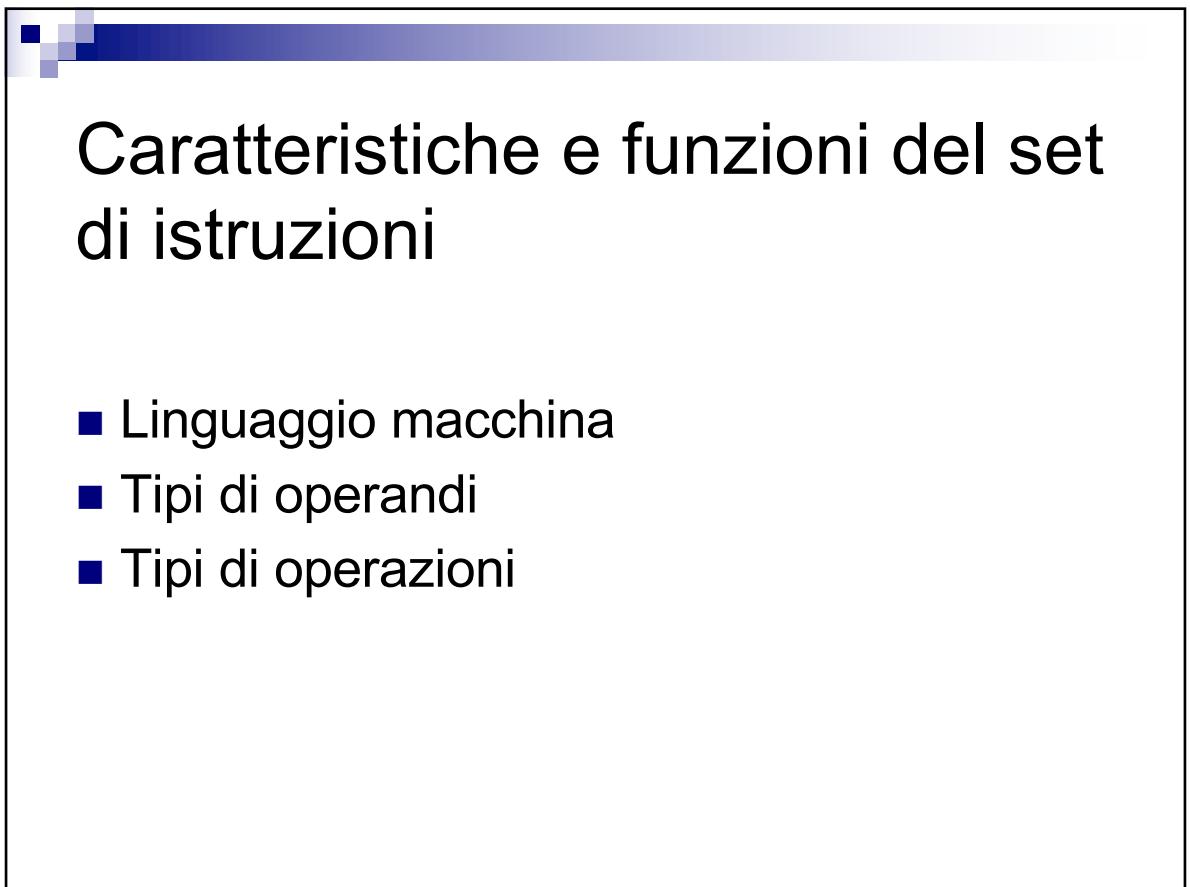


Linguaggio macchina



Caratteristiche e funzioni del set di istruzioni

- Linguaggio macchina
- Tipi di operandi
- Tipi di operazioni

Linguaggio macchina

- Insieme delle istruzioni (instruction set) che la CPU può eseguire

Elementi di un'istruzione macchina

- Codice operativo
 - Specifica l'operazione da eseguire
- Riferimento all'operando sorgente
 - Specifica l'operando che rappresenta l'input dell'operazione
- Riferimento all'operando risultato
 - Dove mettere il risultato
- Riferimento all'istruzione successiva

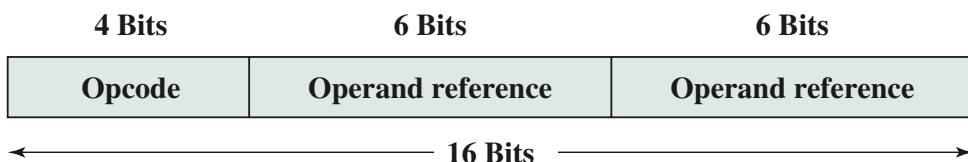
Dove sono gli operandi?

- Memoria centrale (o virtuale)
 - Si deve fornire l'indirizzo
- Registri della CPU
 - Ognuno ha un numero che lo identifica
- Dato immediato nella istruzione
- Dispositivi di I/O
 - Numero modulo o indirizzo di M

Rappresentazione delle istruzioni

- Istruzione = sequenza di bit
 - Divisa in campi
- Spesso viene usata una rappresentazione simbolica delle configurazioni di bit
 - es.: ADD, SUB, LOAD
- Anche gli operandi hanno una rappresentazione simbolica
 - Es.: ADD A,B

Esempio del formato di un' istruzione



Tipi di istruzioni

- Elaborazione dati
 - Istruzione aritmetiche e logiche, di solito sui registri della CPU
- Immagazzinamento dei dati in M o viceversa
- Trasferimento dei dati (I/O)
- Controllo del flusso del programma
 - Salto con o senza test

Quanti indirizzi sono necessari in una istruzione?

- Indirizzi necessari:
 - Un indirizzo per ogni operando (1 o 2)
 - Uno per il risultato
 - Indirizzo istruzione successiva
- Quindi al massimo quattro indirizzi
 - Ma molto raro, e sarebbe molto dispendioso
- Di solito 1, 2 o 3 per gli operandi/risultati

Numero di indirizzi

- 1 indirizzo
 - il secondo indirizzo è implicito
 - di solito si tratta di un registro (accumulatore)
 - situazione tipica nei primi calcolatori

Numero di indirizzi

■ Zero indirizzi

- tutti gli indirizzi sono impliciti
- utilizza una pila (stack)
 - Ad esempio, $c = a + b$ è realizzato come segue
 - push a
 - push b
 - add
 - pop c

Numero di indirizzi

Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	$A \leftarrow B \text{ OP } C$
2	OP A, B	$A \leftarrow A \text{ OP } B$
1	OP A	$AC \leftarrow AC \text{ OP } A$
0	OP	$T \leftarrow (T - 1) \text{ OP } T$

AC = accumulator

T = top of stack

$(T - 1)$ = second element of stack

A, B, C = memory or register locations

Numero di indirizzi

- Meno indirizzi → istruzioni più elementari (e più corte), quindi CPU meno complessa
 - Però più istruzioni per lo stesso programma → tempo di esecuzione più lungo
- Più indirizzi → istruzioni più complesse
- Indirizzo di M o registro: meno bit per indicare un registro
- RISC (Reduced Instruction Set Computer) verso CISC (Complex Instruction Set Computer)
 - Pentium sono CISC, PowerPC (Apple, IBM, Motorola) sono RISC

Numero di indirizzi

Instruction	Comment
SUB Y, A, B	$Y \leftarrow A - B$
MPY T, D, E	$T \leftarrow D \times E$
ADD T, T, C	$T \leftarrow T + C$
DIV Y, Y, T	$Y \leftarrow Y \div T$

(a) Three-address instructions

Instruction	Comment
MOVE Y, A	$Y \leftarrow A$
SUB Y, B	$Y \leftarrow Y - B$
MOVE T, D	$T \leftarrow D$
MPY T, E	$T \leftarrow T \times E$
ADD T, C	$T \leftarrow T + C$
DIV Y, T	$Y \leftarrow Y \div T$

(b) Two-address instructions

Instruction	Comment
LOAD D	$AC \leftarrow D$
MPY E	$AC \leftarrow AC \times E$
ADD C	$AC \leftarrow AC + C$
STOR Y	$Y \leftarrow AC$
LOAD A	$AC \leftarrow A$
SUB B	$AC \leftarrow AC - B$
DIV Y	$AC \leftarrow AC \div Y$
STOR Y	$Y \leftarrow AC$

(c) One-address instructions

Figure 12.3 Programs to Execute $Y = \frac{A - B}{C + (D \times E)}$

Progettare un insieme di istruzioni

- Repertorio
 - quante e quali operazioni
- Tipi di dato
 - su quali dati
- Formato
 - lunghezza, numero indirizzi, dimensione campi, ...
- Registri
 - numero dei registri della CPU indirizzabili dalle istruzioni
- Indirizzamento
 - modo di specificare gli indirizzi degli operandi

Tipi degli operandi

- Indirizzi (interi senza segno)
- Numeri
 - Limite al modulo
 - Limite alla precisione
- Caratteri
- Dati logici

Numeri

- Interi (virgola fissa)
- Virgola mobile
- Quando ci sono soprattutto operazioni di I/O, si usano i decimali impaccati
 - Cifra decimale = 4 bit (0=0000, 1=0001, 2=0010, ..., 8=1000, 9=1001)
 - Inefficiente: solo 10 delle 16 configurazioni vengono usate
 - Es.: 246 = 0010 0100 0110
 - Più lunga della notazione binaria, ma evita la conversione

Caratteri

- Codice ASCII (American Standard Code for Information Exchange)
- Carattere = 7 bit → 128 caratteri in totale
- Caratteri alfabetici + caratteri di controllo
- Di solito 8 bit: un bit per controllo di errori di trasmissione (controllo di parità)
 - Settato in modo che il numero totale di bit a 1 sia sempre pari (o sempre dispari)
 - Es.: 0011100 → ottavo bit a 1
 - Se si ricevono 8 bit con n.ro dispari di 1, c'è stato un errore di trasmissione

Dati logici

- n bit, invece che un singolo dato
- Per manipolare i bit separatamente

Tipi di operandi

- Indirizzi
- Numeri
 - Interi o virgola mobile
- Caratteri
 - Es.: Codice ASCII (American Standard Code for Information Exchange): 7 bit per ogni carattere (128 caratteri), ottavo bit per controllo (settato in modo che il numero totale di 1 sia pari)
- Dati logici
 - Sequenza di bit invece che un singolo dato

Tipi di dati x86

- 8 (byte), 16 (parola), 32 (doppia parola), o 64 (quadword) bit, e 128 (quadword doppia)
- L'indirizzamento è per unità di 8 bit
- Una doppia parola di 32 bit inizia da un indirizzo divisibile per 4
- Non necessario allineamento indirizzi per le strutture dati in memoria
- Allineamento per trasferimento dati (bus)

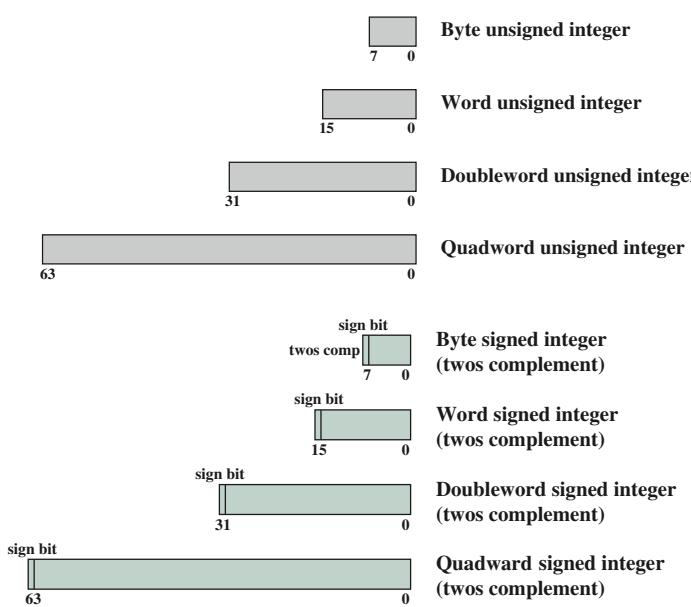
Tipi di dato specifici

General	Byte, word (16 bits), doubleword (32 bits), quadword (64 bits), and double quadword (128 bits) locations with arbitrary binary contents.
Integer	A signed binary value contained in a byte, word, or doubleword, using twos complement representation.
Ordinal	An unsigned integer contained in a byte, word, or doubleword.
Unpacked binary coded decimal (BCD)	A representation of a BCD digit in the range 0 through 9, with one digit in each byte.
Packed BCD	Packed byte representation of two BCD digits; value in the range 0 to 99.
Near pointer	A 16-bit, 32-bit, or 64-bit effective address that represents the offset within a segment. Used for all pointers in a nonsegmented memory and for references within a segment in a segmented memory.
Far pointer	A logical address consisting of a 16-bit segment selector and an offset of 16, 32, or 64 bits. Far pointers are used for memory references in a segmented memory model where the identity of a segment being accessed must be specified explicitly.

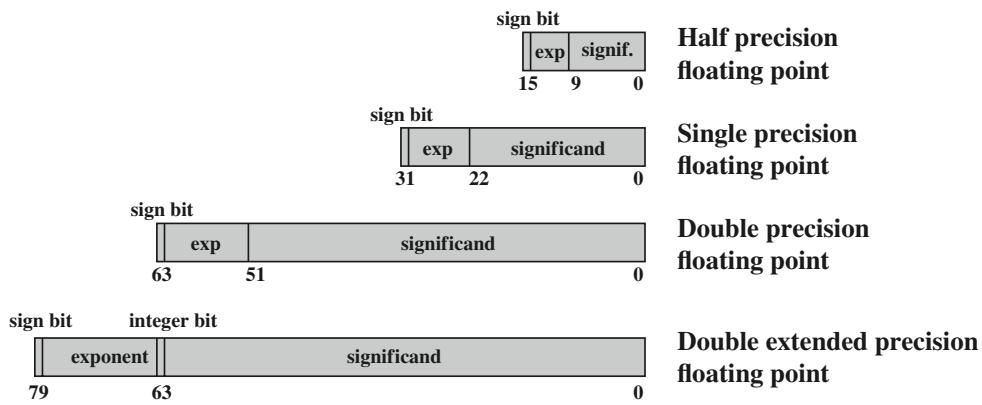
Tipi di dato specifici

Bit field	A contiguous sequence of bits in which the position of each bit is considered as an independent unit. A bit string can begin at any bit position of any byte and can contain up to 32 bits.
Bit string	A contiguous sequence of bits, containing from zero to $2^{23} - 1$ bits.
Byte string	A contiguous sequence of bytes, words, or doublewords, containing from zero to $2^{23} - 1$ bytes.
Floating point	See Figure 12.4.
Packed SIMD (single instruction, multiple data)	Packed 64-bit and 128-bit data types.

Formati di dato numerici x86



Formati di dato numerici x86



Tipi di operazioni

- Trasferimento dati
- Aritmetiche
- Logiche
- Conversione
- I/O
- Sistema
- Trasferimento del controllo

Trasferimento dati

- Deve specificare
 - Sorgente: dove è il dato da trasferire
 - Destinazione: dove va messo
 - Lunghezza del dato da trasferire
- Diverse scelte
 - Esempio: codici operativi diversi per trasferimenti diversi (L, LH, LR, LER, LE, LDR, LD in IBM 370) o stesso codice (MOV in VAX) ma specifica nell'operando

Aritmetiche

- Somma, sottrazione, moltiplicazione, divisione
- Interi con segno sempre
- Spesso anche per numeri in virgola mobile
- Possono includere anche:
 - Incremento
 - Decremento
 - Negazione
 - Valore assoluto

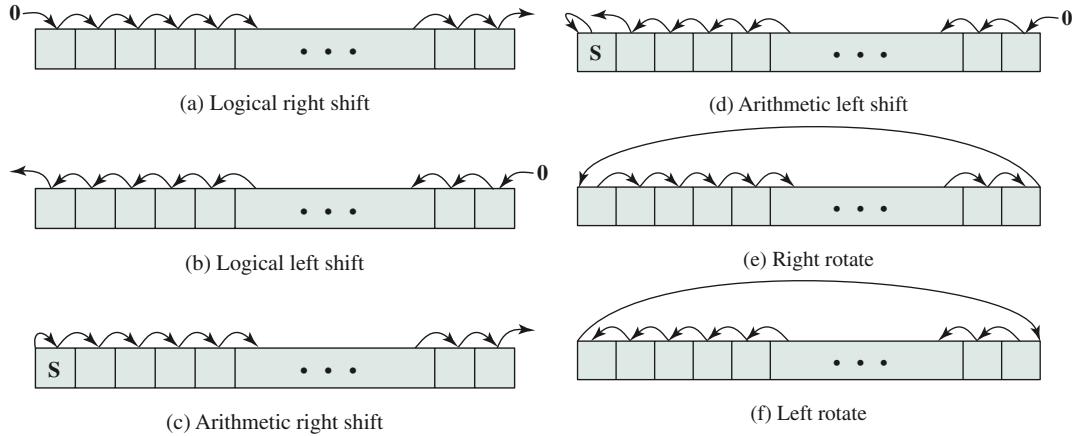
Logiche

- Operazioni sui bit
- AND, OR, NOT, XOR, EQUAL
- Possono essere eseguite in parallelo su tutti i bit di un registro
 - And come maschera

Esempio

- Parole da 16 bit con 2 caratteri (8 bit ciascuno)
- Per inviare il carattere di sinistra a un modulo di I/O:
 - Carico la parola in un registro a 16 bit
 - AND del registro con 1111111100000000
 - Traslo a destra per 8 volte
 - Mando al modulo di I/O il registro (legge gli 8 bit più a destra)
 - Per il carattere di destra, AND con 0000000011111111, e non serve la traslazione

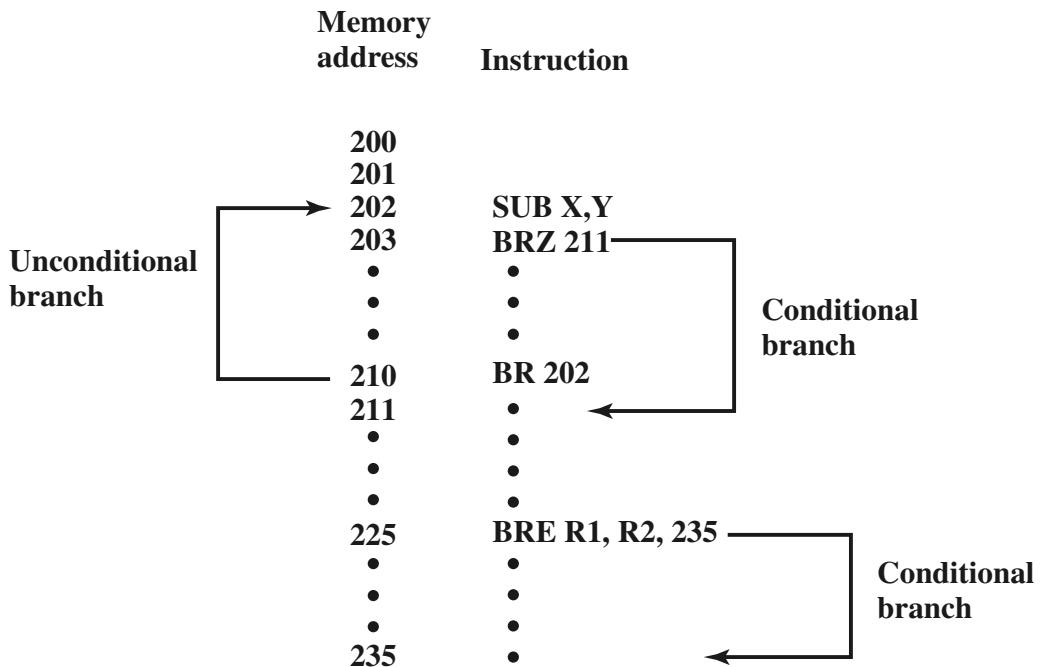
Operazioni di shift e rotazione



Transferimento del controllo

- Salto condizionato (branch)
 - Es.: salta a x se il risultato è 0
 - Registro condizione o più operandi
 - Es.: BRE R1, R2, X
 - Perché saltare?
 - Istruzioni da eseguire varie volte
 - Decidere cosa fare sulla base del verificarsi di certe condizioni
 - Programmazione modulare

Salto condizionato



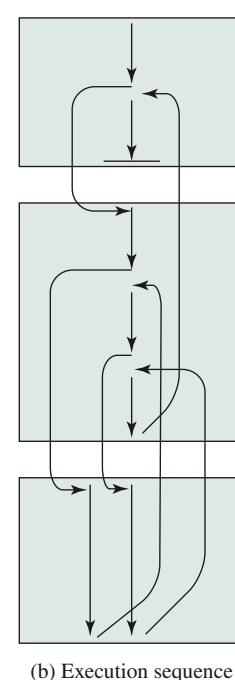
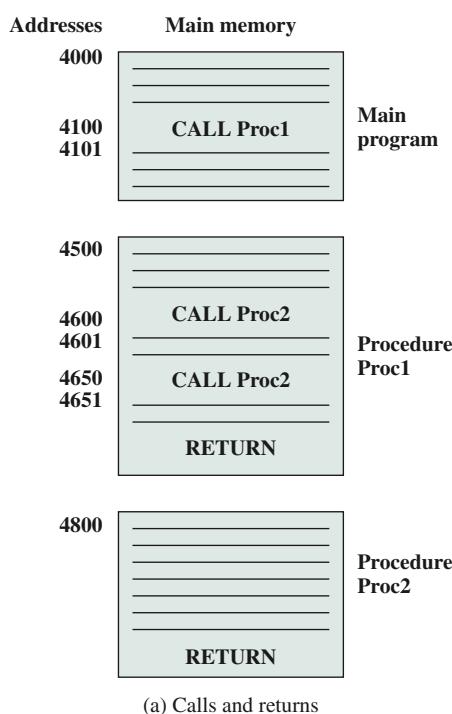
Transferimento del controllo

- Salto incondizionato (skip)
 - Scavalca un'istruzione e passa alla successiva
 - Non ha operandi
- Per usare lo spazio operandi
 - es.: incrementa e salta se 0 (istruzione ISZ)

Chiamate di procedura

- Procedura: pezzo di programma a cui si dà un nome, in modo da eseguirlo (chiamarlo) da qualunque punto di un programma indicando il suo nome
 - Risparmio codice: scrivo solo una volta un pezzo di codice
 - Modularità: posso affidare la scrittura di una procedura ad un altro programmatore
- Due istruzioni: chiamata e ritorno
 - Entrambe di salto

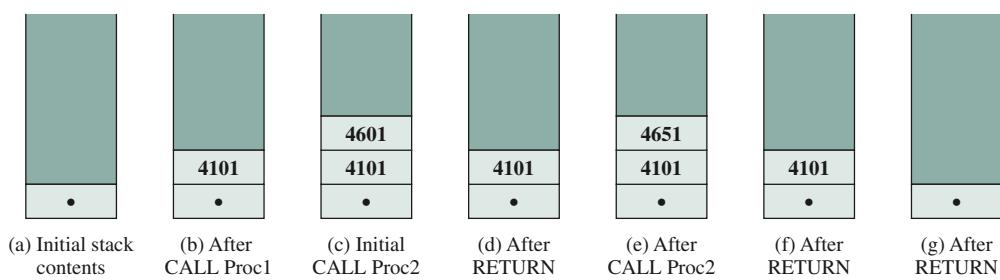
Chiamate di procedura annidate



Indirizzo di ritorno

- Luoghi per memorizzare l'indirizzo di ritorno
 - Un registro
 - CALL X provoca:
 $RN \leftarrow PC + D$ (D =lunghezza istruzione)
 $PC \leftarrow X$
 - Inizio della procedura chiamata
 - CALL X provoca:
 $X \leftarrow PC + D$
 $PC \leftarrow X+1$
 - Cima della pila: porzione di M dove le scritture/lettture avvengono sempre in cima
 - Gli indirizzi di ritorno vengono memorizzati in cima alla pila, uno dopo l'altro, e vengono presi nell'ordine inverso alla chiusura delle procedure

Uso della pila



Linguaggio assembly

- Indirizzi numerici → indirizzi simbolici
 - Per operandi o istruzioni
- Codici operativi → simboli
- Assemblatore: programma che traduce dal linguaggio assembly al linguaggio macchina

Nota a margine: ordinamento byte e bit in memoria

- Come sono memorizzati dati e indirizzi in memoria ?
 - con che ordine per i byte ?

Es. Parola da 4 byte: 12345678_{hex}

	Address	Value		Address	Value	
Big endian	184	12		184	78	
	185	34		185	56	
	186	56		186	34	
	187	78		187	12	

Little endian

- con che ordine per i bit ? Big/Little endian