Il linguaggio assembly

Introduzione al linguaggio macchina

Che cos'è l'assembly
Elementi del linguaggio
Memoria di programma
Registri interni e di I/O
Registri particolari
Rappresentazione dell'informazione
Algebra in complemento a 2

Il linguaggio assembly

- È lo strumento di programmazione più vicino alla realtà fisica-elettronica di un elaboratore
  - Gli elementi del linguaggio corrispondono agli elementi dell'architettura del processore
  - I programmi assembly sono codificati (cioè tradotti in valori binari) in modo diretto e inseriti nella memoria di programma dell'elaboratore per essere eseguiti
  - C'è relazione diretta tra programma assembly e tempo di esecuzione della macchina
- Considerazione: ogni macchina (o famiglia di macchine) ha il suo assembly

Un programma assembly...

- È una lista ordinata di istruzioni, contenenti riferimenti a elementi di vario tipo
  - Ciascuna istruzione ha una lunghezza nota e occuperà un preciso indirizzo nella memoria di programma

```
.org 100                    ;scrivi dall'indirizzo 100
MULT:    push    CNT        ;salva il contatore
         clr     OUT1       ;cancella la parte alta del risultato
         mov     OUT0,IN0   ;copia il moltiplicatore nella parte bassa
         ldi     CNT,8      ;inizializza il contatore
         lsr     OUT0       ;shifta il moltiplicatore a destra
M1:      brcc    M2         ;somma solo se C set
         add     OUT1,IN1   ;somma il moltiplicando a OUT1
M2:      ror     OUT1       ;ruota a destra la parte alta del risultato
         ror     OUT0       ;ruota a destra la parte alta del risultato
         dec     CNT        ;decrementa il contatore (non tocca C)
         brne    M1         ;ripeti per 8 volte
         pop     CNT        ;ripristina il contatore
         ret
```

Elementi del linguaggio (1)

- Istruzioni con i loro operandi
  - Le istruzioni sono individuate da un codice mnemonico e da alcuni simboli che indicano su quali valori operano (gli operandi)
  - Gli operandi sono variabili e/o costanti di tipo opportuno
- Etichette (label)
  - Partono dall’inizio della riga e sono concluse dai “:”
  - Indicano simbolicamente l’indirizzo di una istruzione nella memoria di programma

Elementi del linguaggio (2)

- I commenti
  - Sono elementi accessori importantissimi per la comprensione del funzionamento del programma e la sua manutenibilità
  - Sono introdotti (tipicamente) dal “;”
- Le direttive
  - Sono comandi dedicati al programma che ha il compito di tradurre l’assembly in codice eseguibile (assembler)
  - Sono introdotti dal “.”

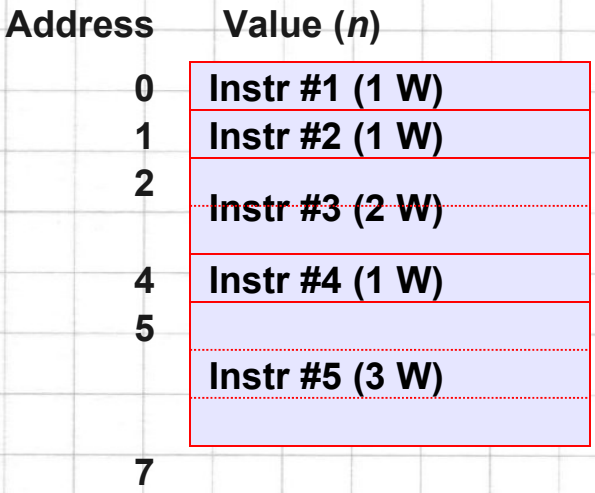
Tipologie di memoria

- Nell’elaboratore elementare sono presenti diversi tipi di memoria, a cui sono associati gli elementi del linguaggio
  - Memoria di programma
  - Memoria dei dati interna
    - Registri interni
    - Registri di I/O
    - Variabili logiche
      - Contengono informazioni sull’esito di ogni istruzione
  - Memoria dei dati estesa

Rappresentazione dell'informazione

- A ogni tipo di memoria (dimensione *n* bit) possono essere attribuiti valori binari (2^n) con diversi significati
  - Valori di tipo istruzione
  - Valori di tipo byte (8 b) o word (16 b) senza segno
  - Valori di tipo byte o word con segno
  - Valori di tipo carattere
  - Valori di tipo indirizzo
    - della memoria di programma
    - della memoria dati estesa
  - Valore di tipo salto (displacement)
  - Valore di tipo flag

Memoria di programma (1)



- Contiene in generale valori di tipo istruzione e ha dimensione di *m* parole da *n* bit
  - Ogni valore istruzione sarà rappresentato da 1, 2 o più parole da *n* bit
  - Ogni locazione è individuata da un indirizzo costituto da log2(*m*) bit

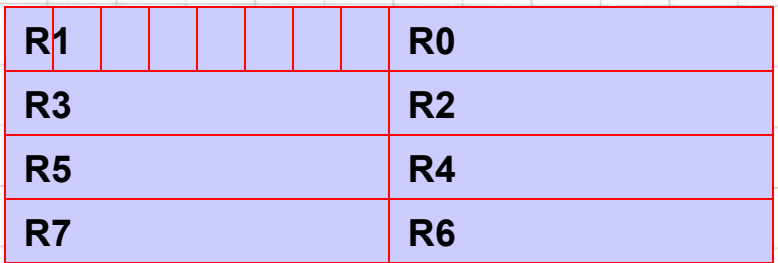
Memoria di programma (2)

- Il valore istruzione è un record articolato, costituito da più campi, variabili in generale per numero e dimensione
- È possibile individuare il campo “codice operativo” e uno o più campi per indicare gli operandi
- Esistono istruzioni con 0, 1, 2 o più operandi
  - Il primo operando indicato individua in genere anche la destinazione del risultato

Codice operativo	Op destinazione
Op sorgente	

Registri interni (1)

- Sono contenuti all'interno dell'elaboratore e sono generalmente le memorie più accessibili
  - Sono a volte raggruppati in banchi
  - Hanno dimensioni tipiche di 8 b (per piccole macchine), ma possono essere raggruppati per formare variabili di dimensioni maggiori
  - Sono individuati da sigle come R0, R1, ecc
    - Ma esistono direttive per assegnare loro nomi simbolici



Registri interni (2)

- Contengono in genere valori numerici, con o senza segno, o di tipo carattere
  - Sono i valori su cui vengono eseguite le operazioni previste dalle singole istruzioni
  - Esistono diverse leggi che associano al valore dei bit (rappresentante) un diverso valore (rappresentato)
- Alcuni registri interni (a volte tutti) sono specializzati per contenere indirizzi
  - Si definiscono “puntatori”
    - Si hanno puntatori alla memoria di programma o alla memoria dati estesa

Registri di I/O

- Si tratta di registri interni specializzati per la gestione e l'interfaccia con il mondo esterno
  - Spesso, alla periferia dell'elaboratore, esistono sistemi elettronici con funzioni specializzate
    - Conteggio, comunicazione a distanza, analisi di informazioni di tipo analogico, attivazione di attuatori, …
    - Sono definite "periferiche"
  - Attraverso i registri di I/O è possibile controllare il loro funzionamento e scambiare informazione
  - Sono indicati da codici numerici
    - Ma di norma i costruttori definiscono nomi simbolici significativi per riferirsi a questi registri (es.: PORTB, DDRA, ecc.)

Memoria dati estesa

- Oltre ai registri interni, spesso esiste una matrice di grandi dimensioni (anche kilobyte) utile per memorizzare dati
- Contiene lo stesso tipo di informazioni memorizzate nei registri interni ed è organizzata come array di *k* byte
  - Ogni dato sarà rappresentato da 1, 2 o più byte
  - Ogni locazione è individuata da un indirizzo costituto da log2(*k*) bit

Registri interni specializzati

- Alcuni registri interni hanno funzioni particolari
  - Possono essere legati al meccanismo di funzionamento del processore
    - PC (Program Counter)
  - Possono dare informazioni sullo svolgimento dello programma
    - STATUS (Registro di stato)
  - Permettono di realizzare particolari strutture dati
    - SP (Stack Pointer)

Registri particolari: PC

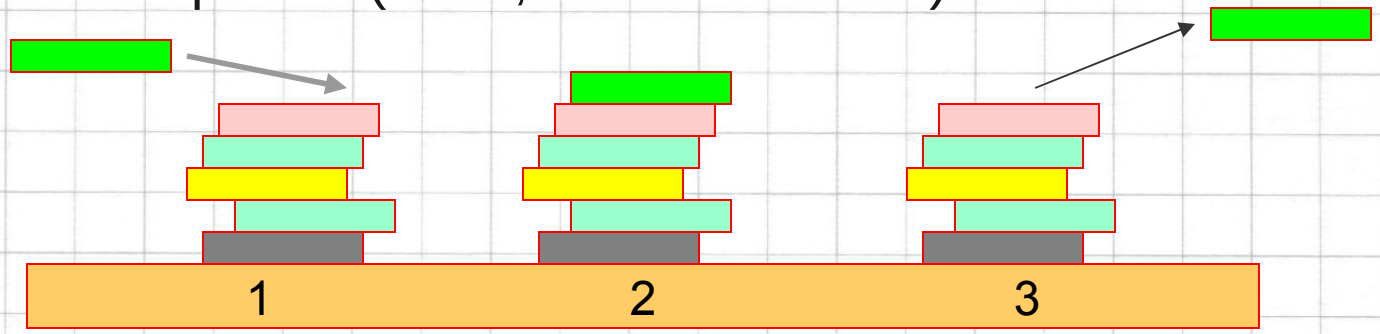
- Il contatore di programma
  - È una variabile che contiene un
  - L’indirizzo fa riferimento alla memoria di programma
- Significato
  - Indica l’istruzione che la macchina sta per eseguire
  - Viene aggiornata automaticamente dalla macchina
    - All’inizio del programma è inizializzata automaticamente a un valore noto (tipicamente 0)
    - Normalmente è incrementata di un numero pari alla dimensione dell’istruzione caricata (le istruzioni sono normalmente lunghe 1, 2 o 3 unità della memoria di programma)
    - Ci sono istruzioni che possono alterarne il contenuto in modo predefinito (salti)

Registri particolari: STATUS

- È una variabile che raggruppa bit il cui valore logico dà informazioni sull'esecuzione del programma
  - I bit del registro STATUS vengono definiti
  - Le informazioni principali provengono dall'esecuzione di istruzioni logico-aritmetiche
    - C (carry, riporto) indica la presenza di riporto o prestito in operazioni di somma/differenza tra interi senza segno
    - S (sign, segno) indica se il segno di una operazione è negativo (S = 1) o non negativo (S = 0)
    - Z (zero) indica se un'operazione ha dato risultato nullo (Z = 1)
    - P (parity, parità) indica se il numero di bit a 1 del risultato di un'operazione è pari (P = 0) o dispari (P = 1)

Registri particolari: SP (1)

- Nella memoria dati estesa si può creare una particolare struttura dati: la pila (stack)
  - Può essere utile per appoggiarci i valori contenuti nei registri interni che devono essere usati per altre operazioni
  - Come in una pila di libri, è possibile inserire nuovi libri (in cima) o recuperare il libro posto più in alto
    - L'ultimo oggetto inserito è quello disponibile per essere recuperato (LIFO, last in - first out)

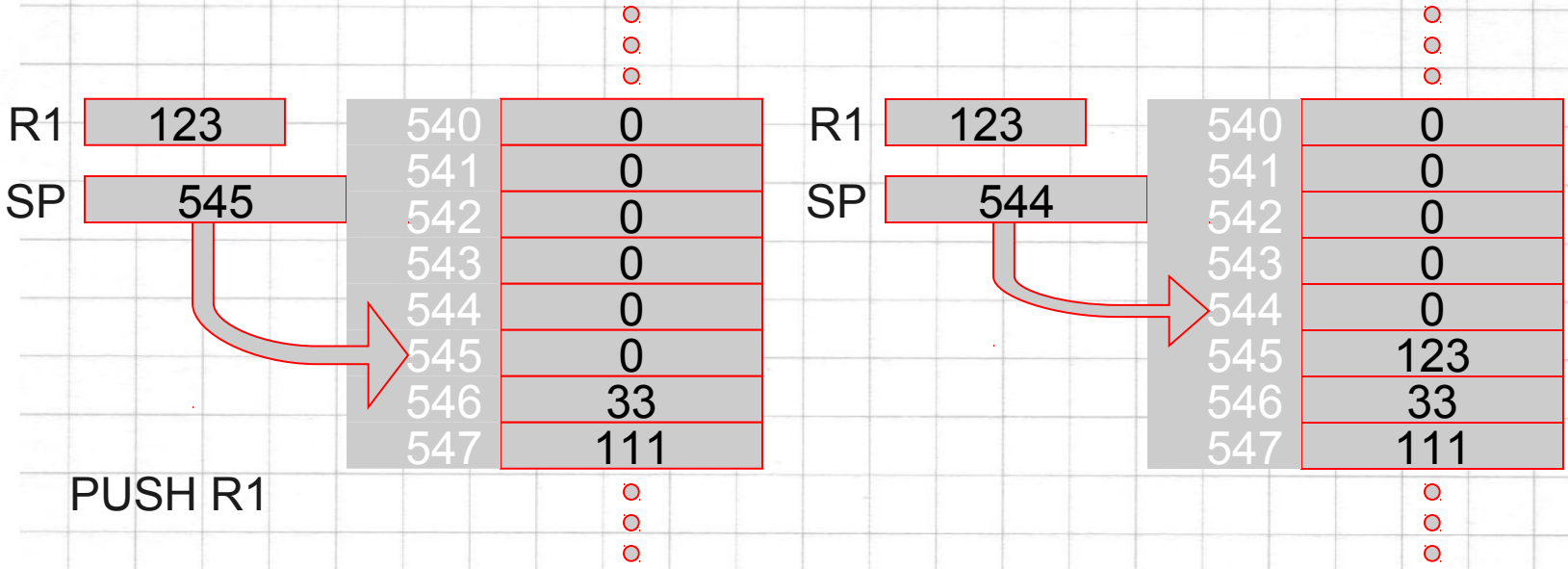


Registri particolari: SP (2)

- Come è gestito lo stack?
  - Il registro SP (stack pointer) contiene l'indirizzo della locazione in cima allo stack
    - Le dimensioni del registro SP sono quindi tali da contenere una variabile di tipo "indirizzo alla memoria dati estesa"
    - Si tratta di un numero binario senza segno compreso tra 0 e *k*
  - Il registro SP viene usato per gestire le operazioni di inserimento e prelievo dalla pila
    - Indica l'indirizzo della memoria dati estesa in cui il dato deve essere inserito o da cui deve essere prelevato
    - Il valore di SP è aggiornato in modo automatico, in modo da garantire il funzionamento della pila

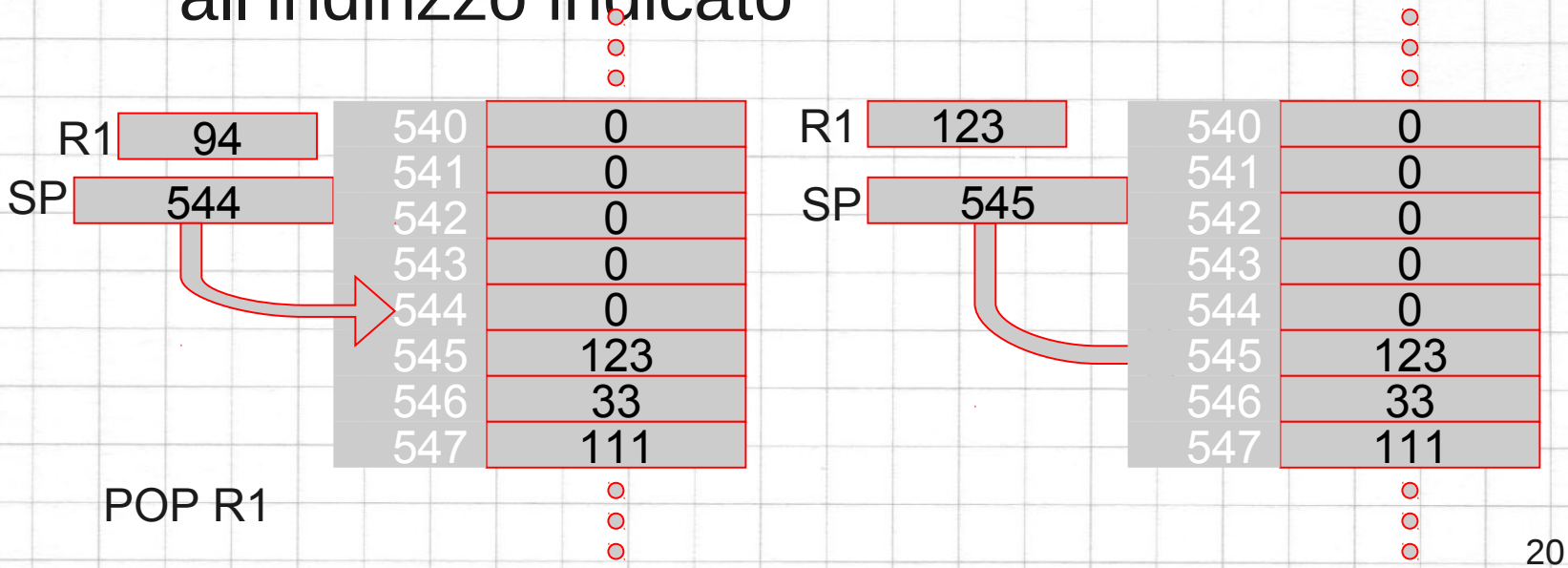
Registri particolari: SP (3)

- Inserimento nello stack (PUSH)
  - L'oggetto viene posto nella memoria all'indirizzo indicato da SP
  - Il valore di SP viene poi decrementato, in modo che punti una locazione libera



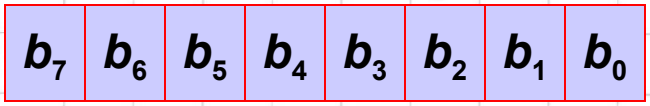
Registri particolari: SP (4)

- Prelievo dallo stack (POP)
  - Il valore di SP viene incrementato
    - Ritorna a puntare l'ultima cella in cui era stato inserito un dato
  - Viene prelevato il valore dalla memoria all'indirizzo indicato



Rappresentazione

- Per descrivere il funzionamento delle istruzioni e capire che tipo di informazioni abbiamo nelle diverse memorie, è necessario avere presenti le principali leggi di rappresentazione
- Iniziamo con le variabili di tipo numerico o carattere contenute in memorie da 8 b (byte)
- Indichiamo i bit del rappresentante con
  - b_7 (MSB), b_6, b_5, b_4, b_3, b_2, b_1, b_0 (LSB)
  - MSB (LSB): most (least) significant bit



Rappresentazione - byte (1)

- Interi binari senza segno
  - È la codifica numerica più immediata
  - Esprime valori compresi tra 0 e 255
  - $x = 128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0$
- Codifica BCD
  - Mantiene traccia della base 10, usata normalmente dagli “umani”
  - Esprime valori compresi tra 0 e 99 (alcune combinazioni di valori non sono ammesse)
  - $x = 10(8b_7 + 4b_6 + 2b_5 + b_4) + 8b_3 + 4b_2 + 2b_1 + b_0$

Rappresentazione - byte (2)

- Interi binari con segno
  - È detta codifica in “complemento a 2”
  - Esprime valori compresi tra -128 e 127
    - $x = -128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0$
  - Ha l’importante proprietà di poter essere gestita dagli stessi operatori di somma e differenza dei numeri interi senza segno
  - Questa rappresentazione può essere adatta a valori di tipo salto

Rappresentazione - byte (3)

- Valori di tipo carattere
  - Il valore della variabile è un simbolo alfanumerico
    - Lettere alfabetiche maiuscole e minuscole
    - Numeri
    - Simboli di punteggiatura, di valuta, matematici, grafici
    - Carattere speciali, come l’indicazione di “a capo”
  - Viene assegnato in modo convenzionale, facendo riferimento a tabelle di codifica standard
    - La più diffusa è la codifica ASCII

Rappresentazione - word

- Valori di tipo interi binari senza segno
  - Esprime valori compresi tra 0 e $(2^{16} - 1)$
  - $x = 2^{15}b_{15} + 2^{14}b_{14} + \ldots + 2^{3}b_{3} + 2^{2}b_{2} + 2b_{1} + b_{0}$
  - Questa rappresentazione si presta bene per esprimere valori di tipo indirizzi in piccole macchine, in cui la quantità di memoria è ($\leq$ 64 k) celle
- Valori di tipo interi binari con segno
  - Esprime valori compresi tra -2^{15} e $(2^{15} - 1)$
  - $x = -2^{15}b_{15} + 2^{14}b_{14} + \ldots + 2^{3}b_{3} + 2^{2}b_{2} + 2b_{1} + b_{0}$
  - Rappresentazione in complemento a 2

Ricordiamo l'algebra in complemento a 2

- Il MSB determina il segno del numero rappresentato
  - Quindi b_7 ha valore di bit del segno
- L'inversione di segno si ottiene con operazioni semplici
  - Complemento di bit ($b_i' = 1 - b_i$) e incremento
- La somma e la differenza si ottengono operando sui bit delle rappresentazioni come fossero interi senza segno
  - Proprietà fondamentale della rappresentazione in complemento a 2 che ne ha sancito il successo
  - Cambia il senso dei flag

Leggi di rappresentazioni

- Relazione tra bit di codice *b* e valore rappresentato *xu* (senza segno) e *x* (con segno)
  - Prima di tutto, le leggi di rappresentazione dirette
    - $xu = 128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0$
    - $x = -128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0$
  - Si ottiene
    - $x = xu - 256b_7$
    - $xu = 256b_7 + x$
  - Possiamo quindi invertire la legge
    - Se $0 \leq x \leq 127$, $b_7 = 0$ e $xu = x$
    - Se $-128 \leq x \leq -1$, $b_7 = 1$ e $xu = 256 + x = 256 - |x|$