Corso di Calcolatori Elettronici I

Esempi di programmazione assembly

ing. Alessandro Cilardo

Corso di Laurea in Ingegneria Biomedica

Programma con matrici

File: `programma013.a68`

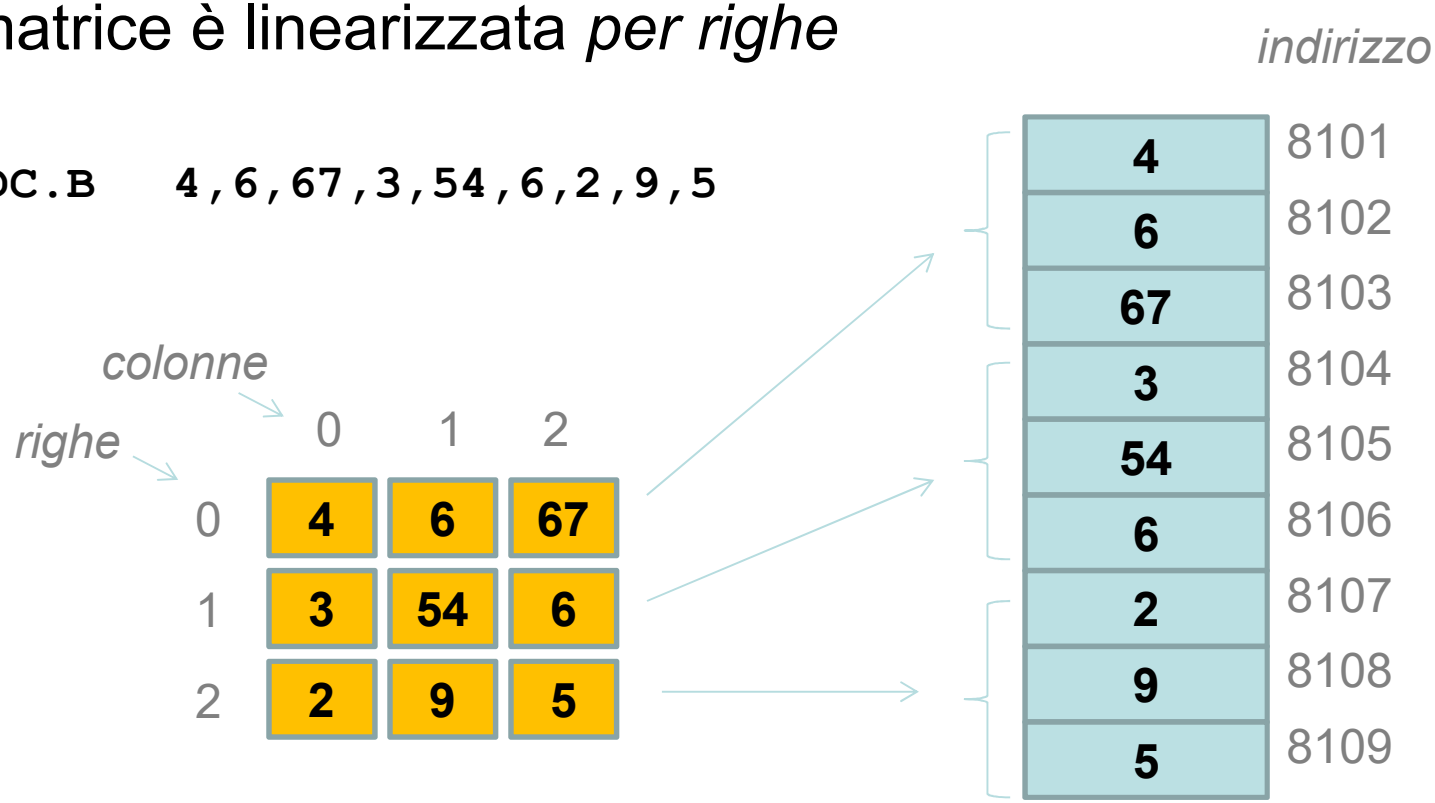
- Scrivere un programma che contenga in memoria una matrice di byte di dimensione **RG** x **CL** (**RG** righe e **CL** colonne)
- Il programma deve trovare il valore minimo all'interno della matrice
- Considerare ad esempio **RG**=3 e **CL**=3

Programma con matrici

File: `programma013.a68`

- La matrice viene *linearizzata* per essere disposta in memoria, byte dopo byte.
- La matrice è linearizzata *per righe*

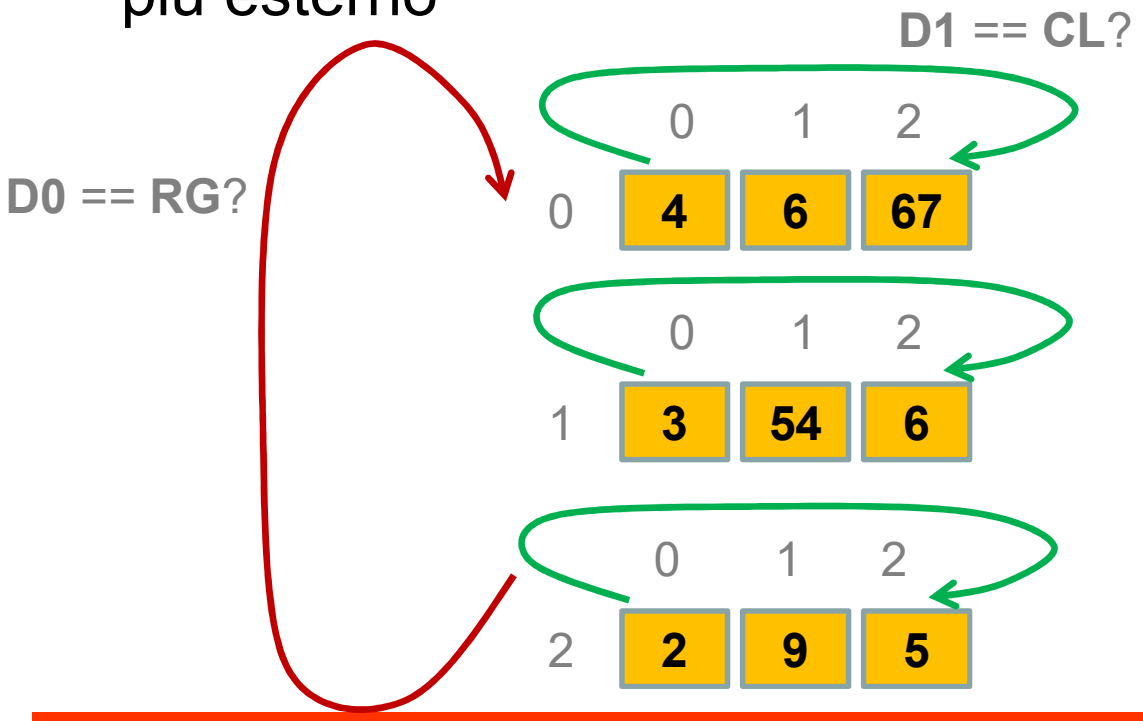
```
MAT     DC.B    4,6,67,3,54,6,2,9,5
```



Programma con matrici

File: `programma013.a68`

- Scorriamo la matrice riga per riga e, per ogni riga, colonna per colonna
- Serve un ciclo *innestato*: un ciclo all'interno di un ciclo più esterno



ognuno dei due cicli avrà la propria variabile di conteggio, ad esempio **D0** per il ciclo sulle righe e **D1** per il ciclo sulle colonne Pertanto, **D0** verrà inizializzato con **0** ed incrementato fino ad eguagliare **RG**, mentre **D1** sarà incrementato fino a **CL**

Programma con matrici

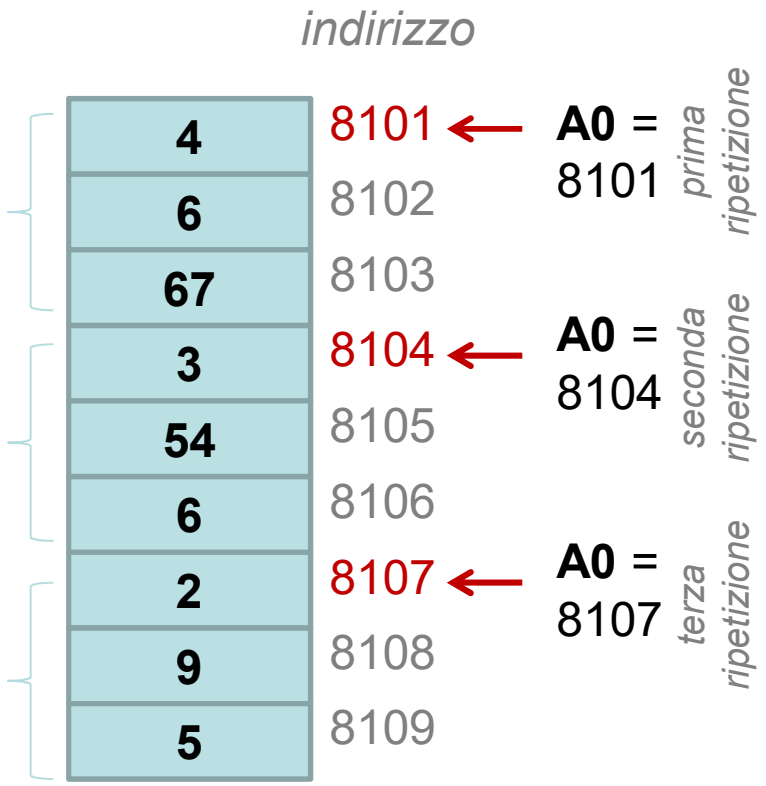
File: `programma013.a68`

- Per la ricerca del minimo, inizializziamo una variabile (scegliamo il registro **D3**) con il massimo valore che un byte con segno può avere (**$7F**).
- Durante lo scorrimento, viene prelevato ogni elemento della matrice e confrontato con **D3**.
- Se il valore appena letto dalla matrice è minore di **D3**, esse viene usato per sovrascrivere **D3** e diventa così il nuovo *minimo temporaneo*
- L'ultimo valore che rimarrà in **D3** dopo lo scorrimento della matrice rappresenterà il valore più basso presente

Programma con matrici

File: `programma013.a68`

- Per “puntare” ad ogni elemento della matrice, viene usato il registro **A0**, che conterrà l’indirizzo del primo elemento di ogni riga
- L’indirizzo dell’elemento da prelevare si può ottenere come somma di **A0** e del registro **D1**, il contatore di colonne usato all’interno di ciascuna riga



Programma con matrici

File: `programma013.a68`

```
        ORG     $8000
MAIN    MOVE.B #$7F,D3
        MOVE.L #MAT,A0
        CLR D0
LOOP    CLR D1
LOOP2   MOVE.B (A0,D1),D4
        CMP.B D3,D4
        BGE SKIP
        MOVE.B D4,D3
SKIP    ADD #1,D1
        CMP #CL,D1
        BNE LOOP2
        ADD #CL,A0
        ADD  #1,D0
        CMP #RG,D0
        BNE LOOP
        MOVE.B D3,MIN
FINE    JMP     FINE
        ORG     $8100
MIN     DS.B    1
RG      EQU             3
CL      EQU             3
MAT     DC.B    4,6,67,3,54,6,2,9,5
        END             MAIN
```

Inizializza il minimo termporaneo in **D3** *col byte più grande*

Pone in **A0** *l'indirizzo del primo byte della prima riga*

Inizializza con **0** *il valore dei due contatori* **D0** *e* **D1**

Preleva in **D4** *l'elemento della matrice all'indirizzo* **A0**+**D1**

Verifica se è il nuovo minimo. Nel caso, sostituisce il valore di **D3** *con quello appena prelevato in* **D4**

Incrementa il contatore di colonna **D1**. *Se il contatore è arrivato al valore massimo* **CL**, *non ripete più.*

Aggiorna il puntatore al primo byte della prossima riga

Incrementa il contatore di colonna **D0**. *Se il contatore è arrivato al valore massimo* **RG**, *non ripete più.*

Pone il valore minimo nella locazione MIN

Programma con vettori

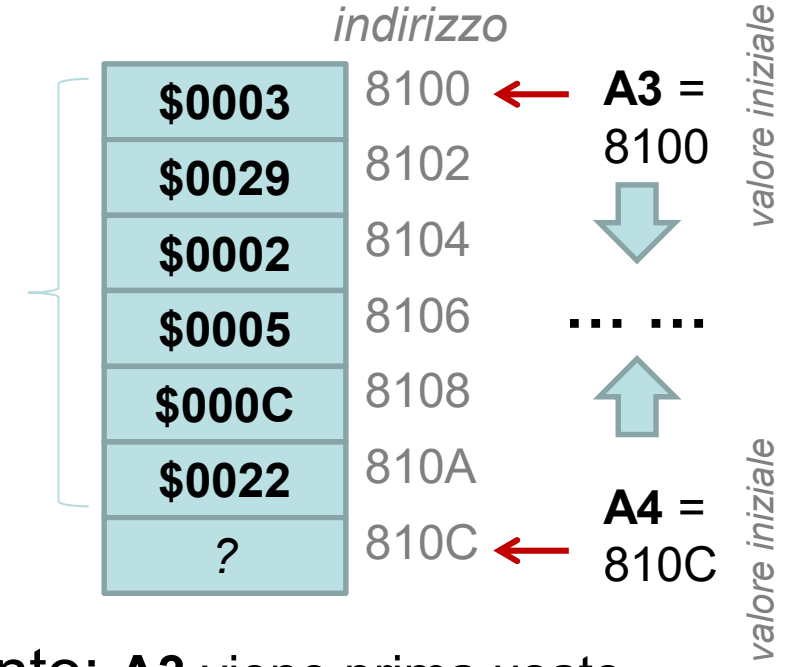
File: `programma014.a68`

- Il programma deve contenere un vettore di **N** word (due byte)
- Le istruzioni del programma devono rovesciare il contenuto del vettore:
- Ad esempio
  - Vettore prima dell'esecuzione: **[$3, $29, $2, $5, $C, $22]**
  - Vettore prima dell'esecuzione: **[$22, $C, $5, $2, $29, $3]**
- Usare la modalità di indirizzamento con autoaggiornamento

Programma con vettori

File: `programma014.a68`

- Useremo due registri indirizzo (scegliamo **A3** e **A4**), che scorrerrano il vettore: **A3** dall'inizio fino a metà, **A4** invece dalla fine fino a metà del vettore
- I contenuti delle due locazioni aventi per indirizzi **A3** ed **A4** verranno invertiti (*swap*)



- Per **A3** useremo il **post**-incremento: **A3** viene prima usato come indirizzo e poi incrementato: partirà dal primo indirizzo (**8100**)
- Per **A4** useremo il **pre**-decremento: **A4** viene prima decrementato e poi usato: partirà dal primo indirizzo successivo al vettore (**810C**), in modo che il primo indirizzo usato sarà **810A**

Programma con vettori

File: `programma014.a68`

```
        ORG     $8000
MAIN    MOVE.L  #VET,A3
        MOVE.L  #(VET+2*N),A4
        CLR D0
LOOP    CMP #(N/2),D0
        BEQ FINE
        MOVE    (A3),D1
        MOVE    -(A4),(A3)+
        MOVE    D1,(A4)
        ADD     #1,D0
        BRA LOOP
FINE    JMP FINE
        ORG     $8100
N       EQU             6
VET     DC.W
        $3,$29,$2,$5,$C,$22
        END             MAIN
```

Inizializza i due registri "puntatore" **A3** *ed* **A4**. **A3** *partirà dall'indirizzo del primo elemento del vettore. Il valore iniziale di* **A4** *(indirizzo della prima word successiva al vettore) è calcolato con un'espressione (***VET+2N***) valutata staticamente dall'assemblatore*

Usa **D0** *come registro contatore. Parti da zero (con* **CLR***) ed incrementa fino a quando* **D0** *arriva al valore* **N/2**, *metà della dimensione del vettore*

Swap: le tre **MOVE** *invertono i contenuti delle due locazioni e, contestualmente, nel momento opportuno, attuano l'auto-aggiornamento dei due puntatori* **A3** *ed* **A4**. *In questo modo, al termine dell'iterazione del ciclo,* **A3** *punterà al prossimo elemento, mentre A4 sarà l'indirizzo in memoria successivo al prossimo elemento da invertire.*

Programma con stringhe

File: `programma015.a68`

- Scrivere un programma che contenga due stringhe (vettori di caratteri), una stringa più lunga **STR**, ed una più corta **TKN**
- Il programma deve verificare se la stringa più corta **TKN** è contenuta in quella più lunga **STR**

Programma con stringhe

File: `programma015.a68`

Area dati del programma:

- Una stringa è un *vettore di caratteri*
- Un carattere si indica tra apici singoli e rappresenta la codifica in codice ASCII di quel carattere
- As esempio, **'c'** è di fatto il byte di valore **$63**

```
        ORG     $8100
N       EQU     6
STR     DC.B    'c','i','a','o','.','.'
M       EQU     2
TKN     DC.B    'i','a'
RES     DS.B    1
        END     MAIN
```

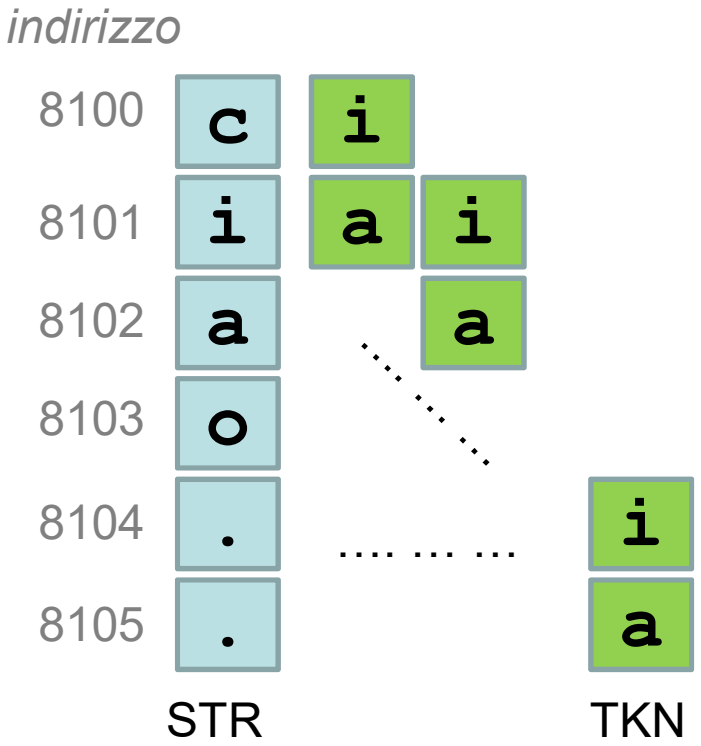
stringa di 6 caratteri, etichettata **STR**

stringa di 2 caratteri, etichettata **TKN**

Programma con stringhe

File: `programma015.a68`

- per verificare se **TKN** è inclusa in **STR**, confrontiamo tutti i caratteri di **TKN** (sono 2 nell'esempio) con i primi due caratteri di **STR**, poi con i successivi due (il secondo ed il terzo), etc.
- Se **STR** è lunga **N** e **TKN** lunga **M**, l'ultimo confronto comincia a partire dal carattere in **STR** in posizione **N-M**

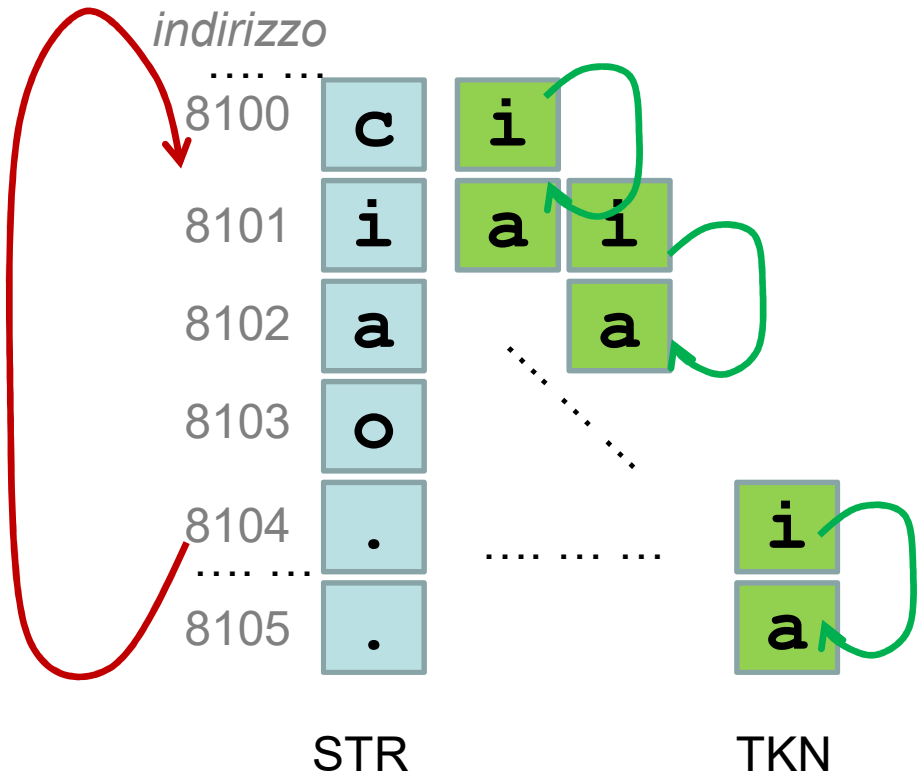


Nell'esempio, **N=6** e **M=2**: l'ultimo confronto parte dal carattere di indice **4** e copre i caratteri in posizione **4** e **5**, ovvero il penultimo e l'ultimo carattere di **STR**

Programma con stringhe

File: `programma015.a68`

- Ricorriamo a due cicli innestati
- Uno scorre tutti i caratteri di **STR** da cui può partire un confronto (nell'esempio, i caratteri in posizione **0**…**4**)
- Per ogni carattere di partenza in **STR**, il ciclo interno scorre tutti i caratteri di **TKN**. Alla prima differenza con il corrispondente carattere di **STR**, concludiamo che **TKN** non è contenuta in **STR** a partire da quella posizione



Programma con stringhe

File: `programma015.a68`

```
        ORG     $8000
MAIN    CLR     RES
        MOVE.L  #STR,A0
        MOVE.L  #TKN,A1
        CLR     D0
LOOP1   CMP     #(N-M),D0
        BGT     FINE
        CLR     D1
LOOP2   CMP     #M,D1
        BEQ     HIT
        MOVE.B  (A0,D1),D2
        CMP.B   (A1,D1),D2
        BNE     EXIT
        ADD     #1,D1
        BRA     LOOP2
HIT     MOVE.B  #1,RES
        BRA     FINE
EXIT    ADD     #1,D0
        ADD     #1,A0
        BRA     LOOP1
FINE    JMP     FINE
```

0 *nella locazione* **RES** *indica che* **TKN** *non è stata trovata*

A0 *punterà ai caratteri della stringa* **STR**, *mentre* **A1** *punterà al primo carattere della stringa* **TKN**

Ripeti fino a quando **D0** *diventa* **N-M** *(compreso), che corrisponde a confrontare* **TKN** *con l'ultima parte di* **STR**

Ripeti fino a quando **D1** *diventa* **M**: *se questo è accaduto, abbiamo riconosciuto l'intera stringa* **TKN** *all'interno di* **STR**

Sposta il carattere di **STR** *in posizione* **A0+D1** *e confrontalo con il carattere di* **TKN** *in posizione* **D1**

Se sono diversi, esci dal ciclo interno e spostati un carattere oltre all'intero di **STR**

1 *nella locazione* **RES** *indica che* **TKN** *è stata trovata*

La stringa **TKN** *è stata trovata: termina qui il programma*

A0 *diventa l'indirizzo del successivo carattere in* **STR**