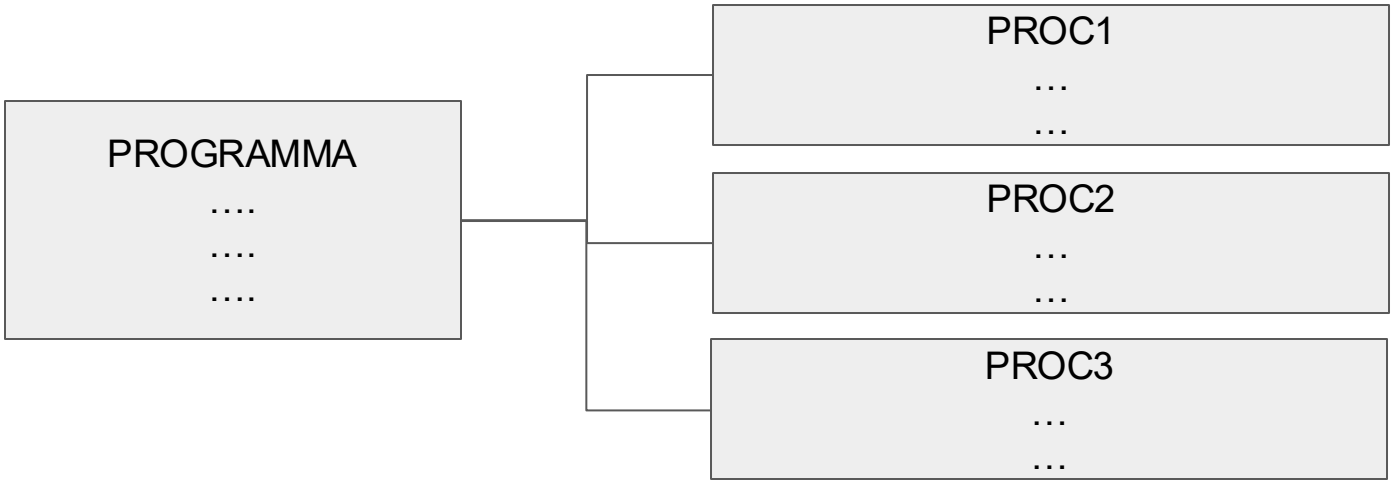
Le procedure in linguaggio Assembly

Docente: Prof. Mario Perna **Email:** prof.perna.mario@darzo.net **Revisione:** 04.04.2025

Introduzione alle procedure

Le procedure sono un insieme di istruzioni che possono essere richiamate in altri punti del programma. Vengono anche chiamate routine o sottoprogrammi



Perchè utilizzare le procedure ?

- Per **riutilizzare il codice** senza riscriverlo.
- Per **organizzare meglio** il programma
- Per **dividere** un problema complesso in parti più semplici.

Istruzioni fondamentali

CALL nome_procedura

- Serve per **chiamare** (eseguire) una procedura.
- Salva l'indirizzo di ritorno sullo **stack**, così il programma può riprendere da dove aveva lasciato.

Istruzioni fondamentali

RET

- Serve per **ritornare** dalla procedura al punto del programma dove era stata chiamata.
- Preleva l'indirizzo di ritorno dallo stack e riprende l'esecuzione da lì.

Come funziona dietro le quinte?

1. Il programma esegue `CALL MiaProcedura`.
2. L'**indirizzo dell'istruzione successiva** viene **messo sullo stack**.
3. L'esecuzione passa al codice della procedura.
4. Quando la procedura esegue `RET`, il programma **estrae l'indirizzo** dallo stack e **continua da lì**.

ESEMPIO (PARTE 1)

```
.model small
.stack 100h
.data
    msg db 'Ciao dal sottoprogramma!$'
.code
main:
    mov ax, @data
    mov ds, ax
```

ESEMPIO (PARTE 2)

```
call StampaMessaggio    ; chiama la procedura


mov ah, 4Ch             ; termina il programma
int 21h
```

ESEMPIO (PARTE 3)

```
; -------- PROCEDURA --------
StampaMessaggio:
    mov ah, 09h             ; funzione DOS per stampare stringa
    lea dx, msg             ; carica l'indirizzo del messaggio
    int 21h
    ret                     ; ritorna al punto dopo il CALL
```

Passare dati alla procedura: registri

METALINGUAGGIO

```
function somma(a){
        return a;
}
```

Passare dati alla procedura: registri

LINGUAGGIO ASSEMBLY

```
mov dl, 'A'     ; passo il carattere 'A'

call stampa     ; chiamo la procedura

stampa:
    mov ah, 02h     ; funzione per stampare un carattere
    int 21h         ; stampa il carattere contenuto in
DL
    ret
```

Salvare dati prima di modificarli: lo stack (PUSH/POP)

Cos'è lo stack?
Lo **stack** (in italiano, **pila**) è una **struttura dati di tipo LIFO** (*Last In, First Out*), utilizzata per **memorizzare temporaneamente dati** in modo ordinato. In Assembly, lo stack è una **zona di memoria** gestita automaticamente dalla CPU tramite il **registro SP (Stack Pointer)**, che indica la **cima della pila**.

Salvare dati prima di modificarli: lo stack (PUSH/POP)

```
mov ax, 1234h      ; valore importante
push ax            ; lo salvo sullo stack
call procedura
pop ax             ; lo riprendo (come era prima)


procedura:
    mov ax, 0      ; modifica AX (ma è temporanea)
    ret
```

ESERCITAZIONE

Scrivi un programma in Assembly che visualizza a schermo i caratteri **'X'** e **'Y'**, usando una **procedura unica** chiamata due volte.

Ogni carattere deve essere passato alla procedura tramite lo **stack**.

Suggerimenti:

- Usa la funzione DOS `AH = 02h` per stampare un carattere (`DL` = carattere)
- Passa il carattere con `PUSH` prima della `CALL`
- Dentro la procedura fai `POP AX`, poi `MOV DL, AL`

SOLUZIONE (parte 1)

```
.model small
.stack 100h
.code
main:
    mov ax, @data
    mov ds, ax

    ; Passa e stampa 'X'
    mov al, 'X'
    push ax
```

SOLUZIONE (parte 2)

```
call stampa_carattere
; Passa e stampa 'Y'
 mov al, 'Y'
 push ax
 call stampa_carattere

 ; Termina programma
 mov ah, 4Ch
 int 21h
```

SOLUZIONE (parte 3)

```
stampa_carattere:
    pop ax      ; recupera il carattere passato
    mov dl, al  ; DL = carattere
    mov ah, 02h ; funzione di stampa DOS
    int 21h
    ret
```