Sottoprogrammi

Calcolatori Elettronici

Argomenti

- Meccanismi di collegamento
- Meccanismi per il passaggio dei parametri
- Stack

Procedure

- Definizione:
  - **Procedura** o **subroutine** - Particolare sequenza di istruzioni su dati di volta in volta differenti

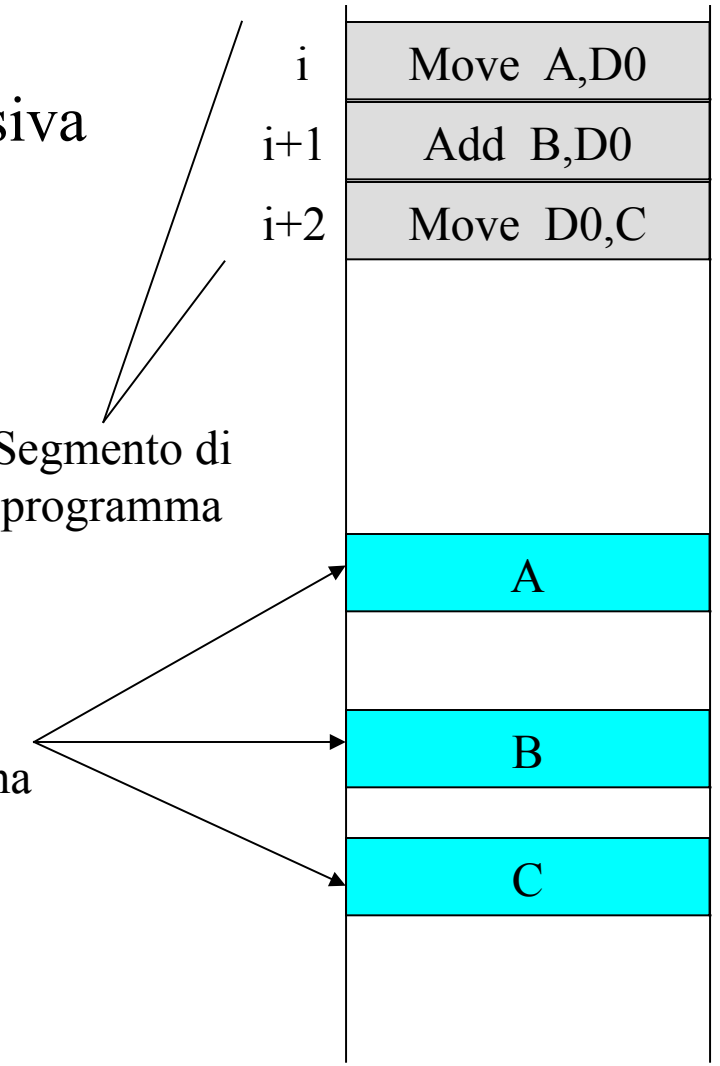
- Problematiche:
  1. **Collegamento** (o **linkage**) - Modo in cui un calcolatore rende possibili le operazioni di **chiamata** e di **ritorno** delle procedure
  2. **Passaggio dei parametri** - Modo in cui il programma chiamante rende disponibili le **informazioni di ingresso** alla subroutine e la subroutine rende disponibili al programma chiamante le **informazioni di uscita**

Meccanismi per il Collegamento

Esecuzione in sequenza lineare

- Ad ogni istruzione eseguita, il PC viene incrementato per puntare all'istruzione successiva

```
while (TRUE){
    Fetch;
    Execute
 }
```



Es: somma di due numeri

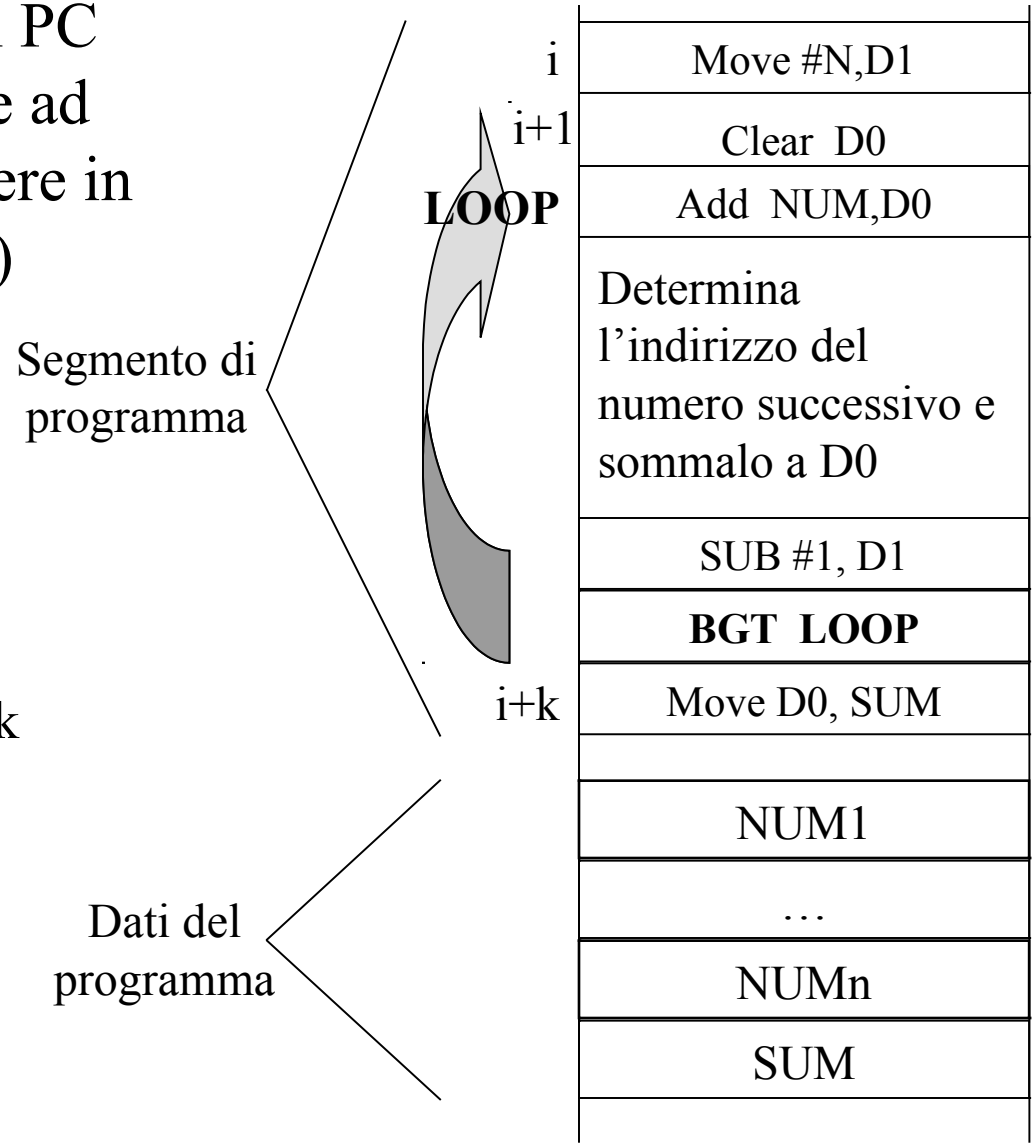
C ⟵ **[A]** + **[B]**

Esecuzione con istruzione di salto

- Ad ogni istruzione eseguita, il PC viene settato in modo da puntare ad un'opportuna istruzione (in genere in base al risultato di un confronto)

Es: somma di una lista di numeri

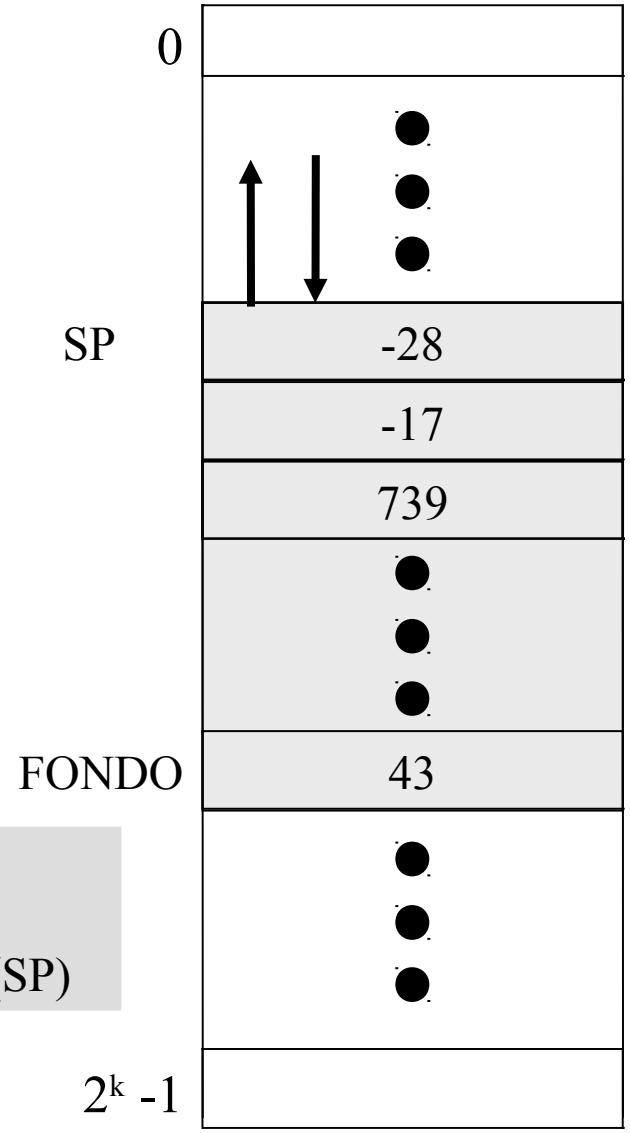
PC = i, i+1, LOOP, ..., LOOP, ..., i+k



Collegamento mediante stack

- Il processore viene dotato di istruzioni apposite:
  - *call_subroutine*
  - *return_from_subroutine*
- Gli indirizzi di ritorno vengono salvati in una struttura dati a pila, chiamata stack del processore:
    - In molti processori queste operazioni sono eseguite dall'istruzione *call_subroutine* (**Es:** 68K)
    - Facilita la gestione di **procedure annidate**
- Un registro particolare, chiamato Stack Pointer, punta alla cima della pila

Stack



@@@ - SP punta all'ultima locazione occupata

Push:

Move NEWITEM, -(SP)

Pop:

Move (SP)+, ITEM

Salto e ritorno da sottoprogramma

```
MAIN …                  Inizio programma principale
     JSR SUBR           Push dell'indirizzo di ritorno su stack e salto a subroutine
     …
     JSR SUBR           Push dell'indirizzo di ritorno su stack e salto a subroutine
     …

SUBR MOVE.W  D0,D2      Prima istruzione della procedura
     …
     RTS                Pop dell'indirizzo di ritorno
```

JSR

Nel momento della chiamata a sottoprogramma, l'istruzione JSR FUNC memorizza l'indirizzo di ritorno sullo stack così:

- il PC che indica l'istruzione successiva viene posto sopra allo stack
- lo SP viene decrementato di 4. Decrementato perchè lo stack cresce verso gli indirizzi piccoli, 4 perchè è la dimensione dell'indirizzo, cioè del registro PC.
- •il PC assume il valore dell'indirizzo FUNC, che è la locazione della prossima istruzione da eseguire. Sarà perciò eseguita la prima istruzione del sottoprogramma FUNC.

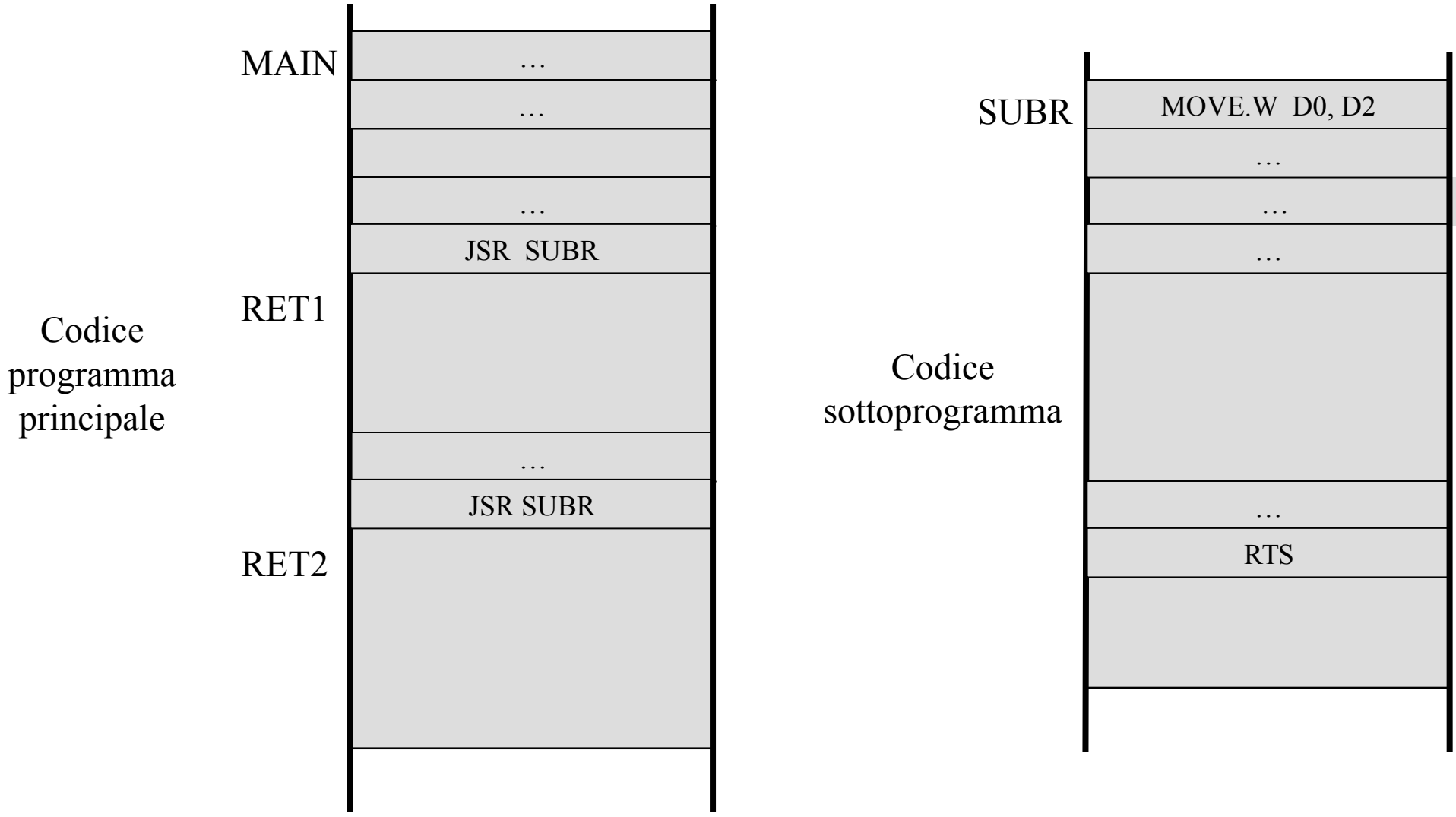
RTS

Il momento del ritorno da sottoprogramma, è quello in cui viene eseguita l'istruzione RTS, che:

- carica dalla cima dello stack il valore da assegnare al Program Counter,
- incrementa di 4 il valore dello Stack pointer, per eliminare il valore caricato lo stack come prima della chiamata.

E' importante notare che se il sottoprogramma ha modificato SP, prima della RTS deve rimetterlo al valore corretto, altrimenti la RTS carica nel PC un valore sbagliato ed effettua il ritorno ad un'indirizzo errato.

Mappa della memoria



Vantaggi e Svantaggi

- Vantaggi:
  - È possibile memorizzare la subroutine in una ROM
  - È possibile effettuare chiamate ricorsive
- Svantaggi:
  - Per essere veloce, richiede risorse hardware extra

Meccanismi per il Passaggio dei Parametri

I dati su cui il sottoprogramma deve lavorare possono essere passati secondo diverse modalità:

- mediante i registri,
- mediante aree dati in memoria,
- mediante lo stack di sistema.

I soluzione: Parametri in registri

- È possibile passare i parametri di input/output in registri del processore

Parametri in registri – Esempio *moltiplicazione*

```
MULT  CLR.W      D0          D0 accumula il risultato parziale
LOOP   ADD.W     D2,D0
       ADD.W     #-1,D1      decrementa il contatore
       BNE       LOOP        e ripete il giro
       RTS                   esce

MAIN  MOVE.W     MPY,D1      mette fattore in D1
      MOVE.W     MPCND,D2      mette fattore in D2
       JSR       MULT       salta a subroutine

PROD   DS.W      1           Riserva spazio di memoria per PROD
MPY    DC.W      3           Definisce il valore di MPY
MCND   DC.W      4           Definisce il valore di MCND
       END       MAIN
```

Parametri in registri – Vantaggi e svantaggi

- È veloce

- È possibile solo se si dispone di abbastanza registri per contenere tutti i dati
- Richiede uno stretto accordo tra chiamante e chiamato

Salvataggio dei registri

Bisogna tenere conto però di quali registri il sottoprogramma usa, e quindi "sporca".

Esiste un'istruzione che permette di salvare un'insieme di registri

- MOVEM A0/D2-D4/D6 ,Address
- MOVEM Address, A0/D2-D4/D6

in particolare per usare lo stack si può chiamare così:

- MOVEM A0/D2-D4/D6 , -(SP) decrementa lo SP
- MOVEM (SP)+ , A0/D2-D4/D6 incrementa lo SP

II soluzione: Parametri in aree di memoria

- È possibile passare i parametri di input/output in aree di memoria
- Tale area può:
  - Appartenere al programma chiamato
  - Appartenere al programma chiamante

Parametri in area di memoria - 1/3

- I parametri di input e output vengono messi in un'area di memoria di 10 word
- Il programma chiamante passa alla subroutine il base address dell'area di memoria mettendolo nel registro A0
- La subroutine accede ai parametri usando il based addressing

Esempio somma – main program

```
ORG $8100
off_op1     EQU    0
off_op2     EQU    2
offh_ris    EQU    4
offl_risEQU    6

op1      DC.W 30000
op2      DC.W 40000
Somma    DS.L 1
Areap  DS.L 8

START    MOVE.L   A0,-(SP)
MOVE.L  #Areap,A0 ;passo in A0 l'inizio dell'area param.
MOVE.W  op1,off_op1(A0) ;carico i param. ingresso nell'area
MOVE.W  op2,off_op2(A0)
JSR     SUB_SOMMA
MOVE.L  offh_ris(A0),somma recupero il risultato
STOP  #$00
```

Esempio somma – subroutine

```
SUB_SOMMA    MOVE.L D0,-(SP)      ;salvo i registri per le var. locali
             MOVE.W  off_op1(A0),D0    ;recupero i dati di ingresso
             ADD.W     off_op2(A0),D0
             BCS     RIPORTO
             MOVE.W  #0,offh_ris(A0)
             BRA     CARICA
RIPORTO      MOVE.W  #1,offh_ris(A0)
CARICA       MOVE.W   D0,offl_ris(A0)
             MOVE.L    (SP)+,D0
             RTS
             END     START
```

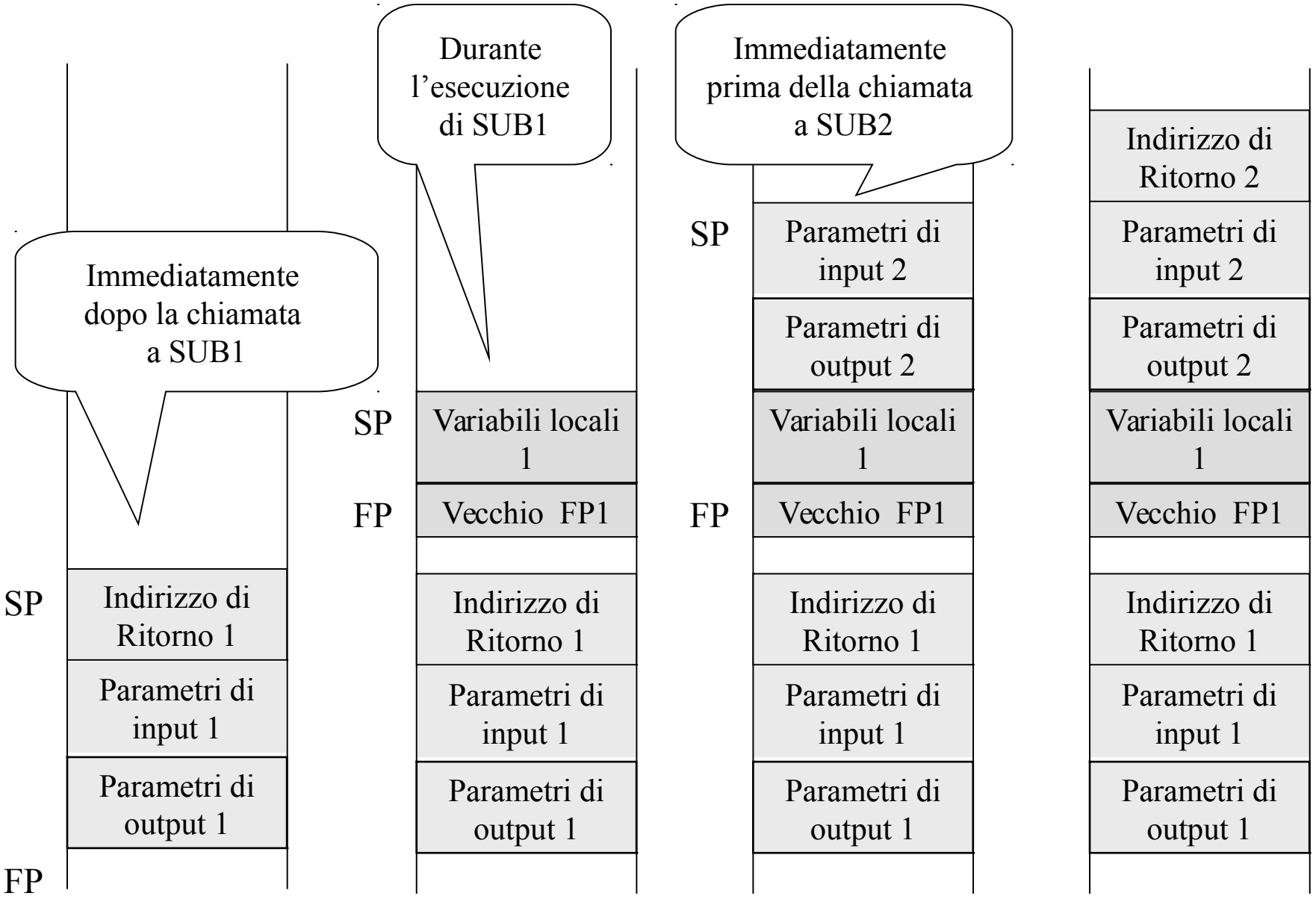
III soluzione: Parametri in area a Stack

- È possibile passare i parametri di input/output in aree organizzate a stack
- È una forma di allocazione dinamica della memoria

Parametri in stack

- I parametri di input, quelli di output el'indirizzo di ritorno vengono messi in un'area di memoria organizzata a stack
- Il programma chiamante
  - Riserva spazio sullo stack per i parametri di output
  - Mette i parametri di input sullo stack
- La subroutine
  - Eventualmente riserva spazio per I parametri locali
  - Sceglie un registro da usare come Frame Pointer
  - Inizializza FP a SP
  - Esegue il proprio codice
  - Mette i parametri di output sullo stack
  - Ripulisce lo stack

Evoluzione dello stack



Link and allocate

- Syntassi:
  - LINK An,#<displacement>
- Funzionamento:
  1. Esegue push su stack del contenuto del registro indirizzo specificato
  2. Il registro indirizzo specificato viene caricato con il nuovo valore dello stack pointer
  3. Il displacement viene esteso in segno e sommato a SP. Questo valore viene assegnato a SP.
  4. Durante l'esecuzione SP varia, mentre FP rimane costante

Parametri in stack – Vantaggi e svantaggi

- ↑ È possibile usare la stessa area di memoria sia per salvare gli indirizzi di ritorno che per passare i parametri di I/O
- ↑ È possibile usare lo stack per variabili temporanee

@@@ - L'allocazione della memoria è dinamica

- ↓ Per essere veloce, richiede risorse hardware extra

Esempio – Somma - Main

```
ORG $8100
op1 DC.W 3
op2 DC.W 4
somma       DS.L 1

START   MOVE.W  op1,-(A7)   ;carico i param. ingresso nello stack
        MOVE.W  op2,-(A7)
        SUB.L   #4,A7
        JSR SUB_SOMMA
        MOVE.L  (A7)+,somma     ;recupero il risultato
        STOP    #$00
```

Esempio – Somma - Subroutine

```
SUB_SOMMA   MOVE.L D0,-(A7)     ;salvo i registri per le var. locali
            MOVE.W   14(A7),D0   ;recupero i dati di ingresso
            ADD.W    12(A7),D0
            BCS  RIPORTO
            MOVE.W   #0,8(A7)
            BRA  CARICA
            RIPORTO MOVE.W   #1,8(A7)
            CARICA   MOVE.W  D0,10(A7)
            MOVE.L   (A7)+,D0
            RTS
            END  START
```