Sottoprogrammi in linguaggio assembly del Motorola 68000



Daniele Paolo Scarpazza daniele.scarpazza@elet.polimi.it

Politecnico di Milano Ultimo aggiornamento: 10 Maggio 2005

Bibliografia

- Sezioni 4.9, 5.6 del libro di testo:
 Hamacher, Vranesic & Zaky
 Introduzione all'architettura dei calcolatori
 McGraw-Hill Science
- Esempi importanti di traduzione da C ad assembly:
 www.eventhelix.com/RealtimeMantra/
 Basics/CToAssemblyTranslation.htm
- Esempi sull'uso del record di attivazione:
 www.cwru.edu/cse/eeap/282/
 25_stack_frames.html

Strumenti

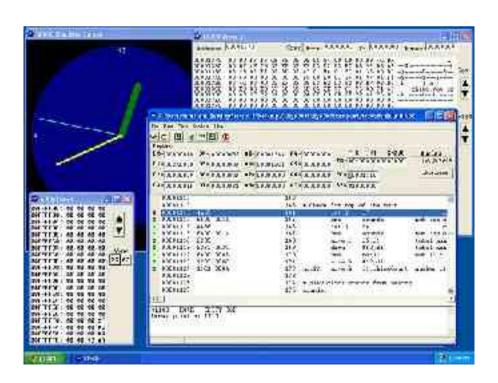
Strumento da usare:

EASy68K

Editor/Assembler/Simulator for the 68000

Disponibile presso:

http://www.monroeccc.edu/ckelly/easy68k.htm



Modello di chiamata di sottoprogramma

- Interessa le seguenti scelte:
 - Dove vanno posti i parametri in ingresso?
 (nei registri? nella pila? in entrambi?)
 - In quale ordine si caricano i parametri sulla pila?
 (solitamente le push appaiono in ordine inverso)
 - Chi deve salvare i registri?
 (il chiamante? il chiamato? quali? partizionamento caller-save e callee-save registers?)
 - Come si passa il valore di ritorno?
 (in un registro? sulla pila? come?
 solitamente sovrascrivendo i parametri)
 - Chi abbatte la pila al ritorno? (il chiamante? il chiamato?)
 - Si usa una catena di record di attivazione (stack frames) oppure no?
- Le scelte confluiscono nella cosiddetta
 Application Binary Interface (ABI) di un certo sistema operativo;
 in assenza di sistema operativo, le scelte sono libere;

Chiamata di sottoprogramma

• Istruzione di salto a sottoprogramma:

```
BSR (branch to subroutine) salto "vicino"

JSR (jump to subroutine) salto "lontano"

salto incondizionato che impila il program counter

e salta all'indirizzo specificato;
```

• Istruzione di ritorno da sottoprogramma: **RTS** (*return from subroutine*) disimpila il *program counter* e salta all'indirizzo appena disimpilato;

Passaggio dei parametri nei registri

• Esempio di programma (~Fig. 5.11, remake di 5.8) che calcola la somma di un array di interi a 32 bit.

• Parametri di ingresso: in A2 l'indirizzo e in D1 la lunghezza dell'array;

• Valore di ritorno: lascia la somma calcolata in D0;

```
$2000
         ORG
         EOU
                  $2000
                                       * we store the sum here
SUM
ARRAY
         DC . L
                  1,2,3,4,5,6,7,8,9,10 * an array of 10 integers
                                       * the array length
LENGTH
        EQU
                   10
                  $1000
         ORG
         ******************
DO SUM
                  #1,D1
                                 * initialize the loop index to N-1
         SUBO.L
                                 * use D0 as accumulator; clear it
         CT<sub>1</sub>R<sub>4</sub>T<sub>1</sub>
                  D0
                                 * sum current cell to DO
LOOP
         ADD.L (A2)+, D0
                  D1, LOOP
                                 * decrement index D1 and loop
         DBRA
         RTS
                                 * return from subroutine
         *******************
START
                   #ARRAY, A2
                                 * set array location parameter to A2
         MOVEA.L
                                 * set array length parameter to D1
         MOVE. To
                   #LENGTH, D1
                  DO SUM
                                 * call the subroutine
         BSR
                  DO, SUM
                                 * save the final sum to variable
         MOVE . L
         STOP
                   #$2000
                   START
         END
```

Passaggio dei parametri sulla pila

- Modifica del programma precedente (~Fig. 5.12) con passaggio di parametri e del valore di ritorno sulla pila;
- La gestione dello stack è esplicita e non fa uso di frame pointer;
- Attenzione al modello di chiamata scelto:
 - push dei parametri nello stesso ordine in cui sono "pensati",
 cioè prima la lunghezza dell'array (primo parametro)
 e poi indirizzo dell'array (secondo parametro);
 di solito succede il contrario;
 - valore di ritorno che sovrascrive il primo parametro;
 - salvataggio dei registri da parte del chiamato;
 il chiamato salva i soli registri che lui sa di usare;
 - il chiamante rimuove i parametri dalla pila dopo la chiamata;

Passaggio dei parametri sulla pila

- Modifica del programma precedente (~Fig. 5.12) con passaggio di parametri e del valore di ritorno sulla pila;
- La gestione dello stack è esplicita e priva di frame pointer;

```
ORG
                      $2000
                                             * the variable where we store the sum
SUM
           EOU $2000
ARRAY
           DC.L 1,2,3,4,5,6,7,8,9,10
                                             * an array of 10 integers
                                             * the array length
LENGTH
           EQU
                      $1000
           ORG
                      D0-D1/A2, -(A7)
                                             * save D0, D1 and A2 on the stack
DO SUM
           MOVEM.L
                                             * retrieve array length from the stack
           MOVE.L
                      16(A7), D1
                                             * initialize the loop index to N-1
           SUBQ.L
                      #1,D1
                                             * retrieve array location from the stack
                      20(A7), A2
           MOVEA.L
                                             * use D0 as accumulator; clear it
           CT<sub>1</sub>R<sub>2</sub>T<sub>1</sub>
                      D0
     LOOP ADD.L
                      (A2) + , D0
                                             * sum current cell to D0
                      D1, LOOP
                                             * decrement index D1 and loop
           DBRA
                                             * save the result onto the stack
           MOVE.L
                      D0, 20(A7)
           MOVEM.L
                       (A7) + D0 - D1/A2
                                             * restores the registers from the stack
                                             * return from subroutine
           RTS
           MOVE . L
                      \#ARRAY, -(A7)
                                             * push array location parameter
START
                      \#LENGTH, -(A7)
                                             * push array length parameter
           MOVE . L
                      DO SUM
                                             * call the subroutine
           BSR
                      4(A7), SUM
                                             * retrieve the result from the stack
           MOVE, I
           ADD.L
                      #8, A7
                                             * reposition the stack pointer
           STOP
                      #$2000
           END
                       START
```

Stato della pila:

FFFFE4	
FFFFE8	
FFFFEC	
FFFFF0	
FFFFF4	
FFFFF8	
FFFFFC	

```
START
       MOVE.L
                  #ARRAY, - (A7)
                                   * array location
                  #LENGTH, - (A7)
                                   * array length
       MOVE . L
       BSR
                  DO SUM
                                   * call
       MOVE.L
                  4(A7), SUM
                                   * result
                                   * reposition SP
                  #8, A7
       ADD.L
                 D0-D1/A2,-(A7)
DO SUM MOVEM.L
                                   * save regs
                  16(A7), D1
       MOVE.L
                                   * array length
                  20 (A7), A2
                                   * array location
       MOVEA.L
                                   * loop
       MOVE.L
                 D0, 20(A7)
                                   * write result
                  (A7) + D0 - D1/A2
       MOVEM.L
                                   * restores regs
       RTS
                                   * return
```

Stato della pila:

1 •)
\sim 1	
(I I I	
	di

FFFFE4	STAR
FFFFE8	
FFFFEC	
FFFFF0	DO_St
FFFFF4	
FFFFF8	
FFFFFC (#ARRAY	

```
MOVE.L
             #ARRAY, - (A7)
                              * array location
  MOVE L
             #LENGTH, - (A7)
                              * array length
  BSR
             DO SUM
                              * call
             4(A7), SUM
                              * result
  MOVE . L
             #8, A7
                              * reposition SP
  ADD.L
            D0-D1/A2, -(A7)
UM MOVEM.L
                              * save regs
             16(A7), D1
  MOVE.L
                              * array length
             20 (A7), A2
                              * array location
  MOVEA.L
                              * loop
  MOVE L
            D0, 20(A7)
                              * write result
             (A7) + D0 - D1/A2
  MOVEM.L
                              * restores regs
  RTS
                              * return
```

Stato della pila:

		START	MOVE.L	#ARRAY, -(A7)	*	arr
FFFFE4		1	MOVE.L	#LENGTH,-(A7)	*	arr
			BSR	DO_SUM	*	cal
FFFFE8			MOVE.L	4(A7), SUM	*	res
FFFFF			ADD.L	#8, A7	*	rep
			• • •			
FFFFEC						
FFFFF0						
FFFFFO		DO_SUM	MOVEM.L	D0-D1/A2,-(A7)	*	sav
			MOVE.L	16(A7), D1	*	arr
FFFFF4			MOVEA.L	20 (A7) , A2	*	arr
			• • •		*	100
FFFFF8	A000000A		MOVE.L	DO, 20 (A7)	*	wri
rrrro	(#LENGTH)		MOVEM.L	(A7) + D0 - D1/A2	*	res
	00002000		RTS		*	ret
FFFFFC						
	(#ARRAY)					

```
ray location
ray length
sult
position SP
ve regs
ray length
ray location
gc
ite result
stores regs
turn
```

Stato della pila:

FFFFE4

FFFFE8

FFFFEC

FFFFF0

FFFFF4

FFFFF8

FFFFFC

0000102E (PC rit) 000000A #LENGTH)

00002000

(#ARRAY)

```
#ARRAY, - (A7)
START
       MOVE L
                                   * array location
                 #LENGTH, - (A7)
                                   * array length
       MOVE L
       BSR
                 DO SUM
                                   * call
                                   * result
       MOVE L
                 4(A7), SUM
                 #8, A7
                                   * reposition SP
       ADD L
DO SUM MOVEM.L
                 D0-D1/A2, -(A7)
                                   * save regs
                 16(A7), D1
                                   * array length
       MOVE . L
                 20 (A7), A2
                                   * array location
       MOVEA.L
                                   * loop
                 DO, 20(A7)
                                   * write result
       MOVE . L
                  (A7) + D0 - D1/A2
       MOVEM.L
                                   * restores regs
       RTS
                                   * return
```

Stato della pila:

FFFFE4 FFFFE8 FFFFEC FFFFF0 0000102E FFFFF4 (PC rit) 000000A FFFFF8 #LENGTH) 00002000 FFFFFC (#ARRAY)

```
#ARRAY, - (A7)
START
       MOVE L
                                  * array location
                 #LENGTH, - (A7)
                                  * array length
       MOVE L
       BSR
                 DO SUM
                                  * call
                                  * result
                 4(A7), SUM
       MOVE L
                                  * reposition SP
                 #8, A7
       ADD L
```

```
DO SUM MOVEM.L
                 D0-D1/A2,-(A7)
                                   * save regs
                  16(A7), D1
                                   * array length
       MOVE L
                  20 (A7), A2
                                   * array location
       MOVEA.L
                                   * loop
                 D0, 20(A7)
                                   * write result
       MOVE.L
                  (A7) + D0 - D1/A2
       MOVEM.L
                                   * restores regs
       RTS
                                   * return
```

Stato della pila:

FFFFE4	
FFFFE8	0000000 (D0)
FFFFEC	00000000 (D1)
FFFFF0	0000000 (A2)
FFFFF4	0000102E (PC rit)
FFFFF8	0000000A (#LENGTH)
FFFFFC	00002000 (#ARRAY)

```
#ARRAY, - (A7)
START
       MOVE . L
                                   * array location
                 #LENGTH, - (A7)
                                   * array length
       MOVE L
       BSR
                 DO SUM
                                   * call
                                   * result
       MOVE . L
                 4(A7), SUM
                 #8, A7
       ADD.L
                                   * reposition SP
```

DO SUM	MOVEM.L	D0-D1/A2,-(A7)	* save regs	3
	MOVE.L	16(A7), D1	* array ler	ngth
	MOVEA.L	20 (A7) , A2	* array loc	cation
	• • •		* loop	
	MOVE.L	D0, 20(A7)	* write res	sult
	MOVEM.L	(A7) + D0 - D1/A2	* restores	regs
	RTS		* return	
	MOVE.L MOVEM.L	D0, 20(A7)	* loop * write res * restores	sult

Stato della pila:

```
#ARRAY, - (A7)
                     START
                           MOVE L
                                                    * array location
FFFFE4
                                     #LENGTH, - (A7)
                                                    * array length
                           MOVE L
                           BSR
                                    DO SUM
                                                    * call
FFFE8 0000000
                                                    * result
                                     4(A7), SUM
                           MOVE.L
                                                    * reposition SP
                                     #8, A7
                           ADD.L
           (D0)
FFFFEC 0000000
        (D1)
FFFFF0 00000000
                     DO SUM MOVEM.L
                                    D0-D1/A2, -(A7)
        (A2)
                                                    * save regs
                                     16(A7), D1
                                                    * array length
                           MOVE L
FFFFF4 0000102E
                                     20 (A7), A2
                                                    * array location
                           MOVEA.L
        (PC rit)
                                                    * loop
FFFFF8 0000000A
                                    DO, 20(A7)
                           MOVE.L
                                                    * write result
                           MOVEM.L
                                     (A7) + D0 - D1/A2
                                                    * restores regs
27+$10 (#LENGTH)
                           RTS
                                                    * return
FFFFFC 00002000
A7+$12 (#ARRAY)
                       -16(A7) = \$FFFFE8 + \$10 = \$FFFFF8
```

Stato della pila:

#ARRAY, - (A7) START MOVE L * array location **#LENGTH**, - (A7) * array length MOVE L **BSR** DO SUM * call * result 4(A7), SUM MOVE L * reposition SP #8, A7 ADD.L

Codice:

```
FFFFE4
FFFE8 00000000
         (D0)
FFFFEC 0000000
      (D1)
FFFFF0 00000000
       (A2)
FFFFF4 0000102E
      (PC rit)
FFFFF8 0000000A
A7+$10 (#LENGTH)
FFFFFC 00002000
A7+$12 (#ARRAY)
```

```
DO SUM MOVEM.L
                 D0-D1/A2, -(A7)
                                   * save regs
                 16(A7), D1
                                   * array length
       MOVE L
                                   * array location
       MOVEA.L
                 20 (A7), A2
                                   * loop
                 DO, 20(A7)
       MOVE.L
                                   * write result
       MOVEM.L
                  (A7) + D0 - D1/A2
                                   * restores regs
       RTS
                                   * return
```

-20(A7) = \$FFFFE8 + \$12 = \$FFFFF8

Stato della pila:

```
#ARRAY, - (A7)
                     START
                           MOVE L
                                                    * array location
FFFFE4
                                     #LENGTH, - (A7)
                                                    * array length
                           MOVE L
                           BSR
                                     DO SUM
                                                    * call
FFFE8 00000000
                                                    * result
                                     4(A7), SUM
                           MOVE L
                                                    * reposition SP
                                     #8, A7
                           ADD.L
           (D0)
FFFFEC 0000000
        (D1)
FFFFF0 00000000
                     DO SUM MOVEM.L
                                     D0-D1/A2, -(A7)
         (A2)
                                                    * save regs
                                     16(A7), D1
                                                    * array length
                           MOVE L
FFFFF4 0000102E
                                                    * array location
                                     20 (A7), A2
                           MOVEA.L
        (PC rit)
                                                    * loop
FFFFF8 0000000A
                           MOVE.L
                                     D0, 20(A7)
                                                    * write result
                           MOVEM.L
                                     (A7) + D0 - D1/A2
                                                    * restores regs
A7+$10 (#LENGTH)
                           RTS
                                                    * return
FFFFFC 00000037
A7+$12 val.rit.
                       20(A7) = \$FFFFE8 + \$12 = \$FFFFFC
```

Stato della pila:

FFFFE4 FFFFE8 FFFFEC FFFFF0 0000102E FFFFF4 (PC rit) 000000A FFFFF8 #LENGTH) 00000037 FFFFFC val.rit.

```
#ARRAY, - (A7)
START
       MOVE L
                                   * array location
                  #LENGTH, - (A7)
                                   * array length
       MOVE L
       BSR
                 DO SUM
                                   * call
                                   * result
                 4(A7), SUM
       MOVE L
                                   * reposition SP
                 #8, A7
       ADD . L
                 D0-D1/A2, -(A7)
DO SUM MOVEM.L
                                   * save regs
                  16(A7), D1
                                   * array length
       MOVE . L
                  20 (A7), A2
                                   * array location
       MOVEA.L
                                   * loop
                 DO, 20(A7)
                                   * write result
       MOVE . L
       MOVEM.L
                  (A7) + D0 - D1/A2
                                   * restores regs
       RTS
                                   * return
```

Stato della pila:

FFFFE4	
FFFFE8	
FFFFEC	
FFFFF0	
FFFFF4	0000102E (PC rit)
FFFFF8	0000000A (#LENGTH)
FFFFFC	00000037 val.rit.

```
START
       MOVE . L
                  #ARRAY, - (A7)
                                   * array location
                  #LENGTH, - (A7)
                                   * array length
       MOVE . L
                                   * call
       BSR
                  DO SUM
       MOVE . L
                  4(A7), SUM
                                   * result
                                   * reposition SP
                  #8, A7
       ADD.L
                 D0-D1/A2,-(A7)
DO SUM MOVEM.L
                                   * save regs
                  16(A7), D1
       MOVE . L
                                   * array length
                  20 (A7), A2
                                   * array location
       MOVEA.L
                                   * loop
       MOVE L
                 D0, 20(A7)
                                   * write result
                  (A7) + D0 - D1/A2
       MOVEM.L
                                   * restores regs
       RTS
                                   * return
```

Stato della pila:

```
#ARRAY, - (A7)
                     START
                            MOVE L
                                                      * array location
FFFFE4
                                      #LENGTH, - (A7)
                                                      * array length
                            MOVE L
                            BSR
                                      DO SUM
                                                      * call
                                                      * result
                            MOVE L
                                      4(A7), SUM
FFFFE8
                                      #8, A7
                                                      * reposition SP
                            ADD L
FFFFEC
FFFFF0
                     DO SUM MOVEM.L
                                      D0-D1/A2, -(A7)
                                                      * save regs
                                      16(A7), D1
                                                      * array length
                            MOVE L
                            MOVEA.L
                                      20 (A7), A2
                                                      * array location
FFFFF4
                                                      * loop
FFFF8 0000000A
                                      DO, 20(A7)
                            MOVE.L
                                                      * write result
                            MOVEM.L
                                      (A7) + D0 - D1/A2
                                                      * restores regs
        (#LENGTH)
                            RTS
                                                      * return
FFFFFC 00000037
        val.rit.
                        4(A7) = \$FFFFF8 + \$4 = \$FFFFFC
```

Stato della pila:

Codice:

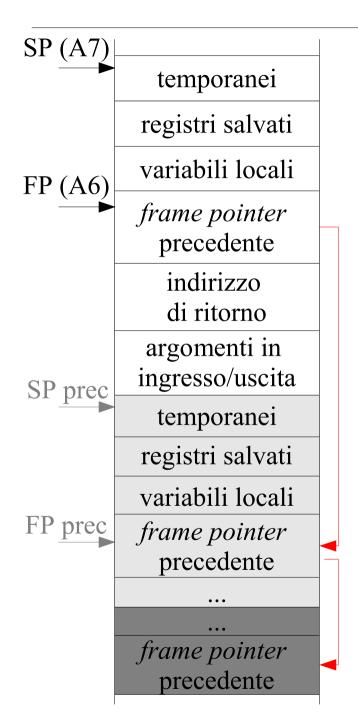
```
FFFFE4
FFFFE8
FFFFEC
FFFFF0
FFFFF4
FFFFF8
FFFFFC
```

```
#ARRAY, - (A7)
START
       MOVE L
                                  * array location
                 #LENGTH, - (A7)
                                  * array length
       MOVE L
       BSR
                 DO SUM
                                  * call
                                  * result
                 4(A7), SUM
       MOVE L
                 #8, A7
       ADD.L
                                  * reposition SP
```

```
D0-D1/A2, -(A7)
DO SUM MOVEM.L
                                  * save regs
                 16(A7), D1
                                  * array length
       MOVE L
       MOVEA.L
                                  * array location
                 20 (A7), A2
                                  * loop
                 DO, 20(A7)
       MOVE L
                                  * write result
       MOVEM. L
                 (A7) + D0 - D1/A2
                                  * restores regs
       RTS
                                  * return
```

-A7+8 = \$FFFFF8 + \$8 = \$1000000

Uso dei record di attivazione



- Una migliore realizzazione del meccanismo di chiamata a sottoprogramma fa uso dei record di attivazione (stack frame);
- Nel 68000 esistono istruzioni dedicate per allocare e distruggere i record di attivazione: LINK e UNLK;
- I record di attivazione formano una lista concatenata (in rosso nella figura);
- Per convenzione, il puntatore al record di attivazione corrente (*frame pointer*) è nel registro A6;
- Creazione del frame: LINK A6, #-N (N è la dimensione delle variabili locali)
- Distruzione del frame: UNLK A6
- I parametri hanno spiazzamento positivo,
 esempio: il primo parametro .L sta a
 8 (A6)
- Le variabili locali hanno spiazzamento negativo,
 esempio: la prima var. locale .L sta a -4 (A6)

Effetto dell' istruzione LINK

• Istruzione LINK A6, #N

variabili locali

frame pointer

precedente

1. [SP] = [SP] - 4 riservo una long word sulla pila

2. [M([SP])] = [A6] salvo vecchio frame pointer

variabili locali

precedente

frame pointer

3. [A6] = [SP] il *frame pointer* ora punta alla cima della pila

4. [SP] = [SP] + N riservo spazio per le variabili locali

(Attenzione: N è solitamente negativo)

variabili locali

frame pointer

precedente

variabili locali

frame pointer

precedente

Prima del punto 1: Dopo il punto 3: Dopo il punto 4: Dopo il punto 1: Dopo il punto 2: (JSR appena eseguita) variabili locali SP frame pointer frame pointer frame pointer (spazio) precedente precedente precedente SP indirizzo indirizzo indirizzo indirizzo indirizzo di ritorno di ritorno di ritorno di ritorno di ritorno parametri in parametri in parametri in parametri in parametri in ingresso/uscita ingresso/uscita ingresso/uscita ingresso/uscita ingresso/uscita registri / temp registri / temp registri / temp registri / temp registri / temp

variabili locali

precedente

► frame pointer

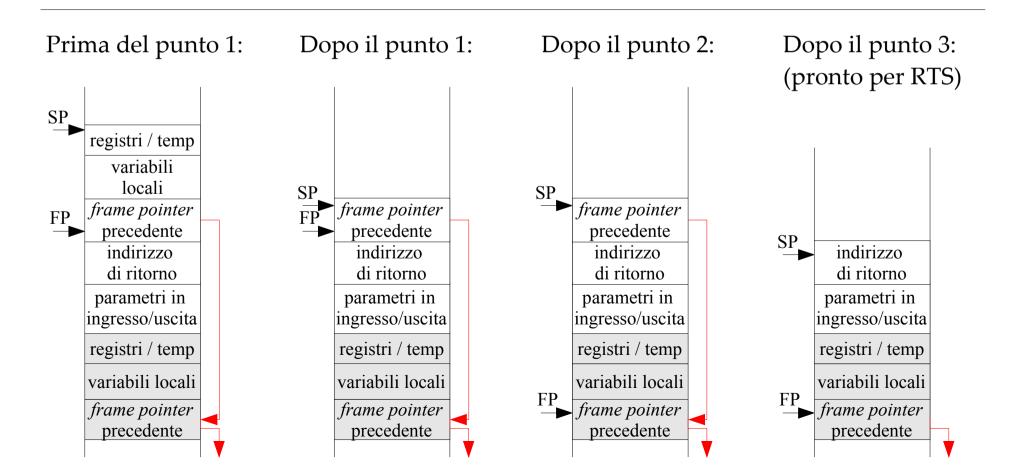
Effetto dell' istruzione UNLK

Istruzione UNLK A6

- 1. [SP] = [A6]
- 2. [A6] = [M([SP])]
- 3. [SP] = [SP] + 4

la nuova cima della pila è il FP precedente ripristino il vecchio *frame pointer*

abbasso la cima della pila di una long word



Schema di funzione in assembly

FUNZIONE:

UNLK

RTS

A6

```
LINK A6, #-N * Riservo spazio per le var locali
MOVEM.L D1-...,-(SP) * Salvo i registri che uso
. . .
* Preparazione a chiamata di funzione con due parametri
MOVE.L ..., -(A7) * Push secondo parametro
MOVE.L ..., -(A7) * Push primo parametro
JSR ALTRA FUNZIONE * Chiamata
ADDQ.L #8, A7 * Pop parametri
MOVEQ.L D0, ...
                * Uso il valore di ritorno di ALTRA FUNZIONE
MOVE.L ..., D0
                        * Imposto il valore di ritorno di FUNZIONE
MOVEM.L (SP)+,D1-...
                        * Ripristino i registri
```

* Elimino var. locali

* Ritorno al chiamante

Esempio di traduzione di funzione

Tradurre in assembly di un programma in linguaggio C:

```
int CallingFunction(int x)
{
   int y;
   CalledFunction(1,2);
   return 5;
}

void CalledFunction(int param1, int param2)
{
   int local1, local2;
   local1 = param2;
}
```

Applichiamo le seguenti convenzioni:

- per semplicità non salviamo i registri;
- il chiamato lascia il valore di ritorno in D0;
- il chiamante abbatte la pila dopo la chiamata;
- usiamo la catena dei record di attivazione;

Esempio di traduzione di funzione

Traduzione in assembly del programma di esempio:

```
int CallingFunction(int x)
                                 A6, #-4
                                                   * Riservo spazio per y
                         LINK
   int y;
   CalledFunction(1,2);
                         MOVE.L #2, -(A7)
                                                   * Push secondo parametro
                         MOVE.L #1, -(A7)
                                                   * Push primo parametro
                                                   * Chiamata
                         JSR
                               CalledFunction
                         ADDQ.L #8, A7
                                                   * Pop parametri
                         MOVEQ.L #5, D0
                                                   * Valore di ritorno in D0
   return 5;
                         UNLK
                                 A6
                                                   * Elimino var. locali
                                                   * Ritorno al chiamante
                         RTS
void CalledFunction(int param1, int param2)
                                 A6, #-8
  int local1, local2;
                                                   * Riservo spazio per var.
                         LINK
                                                   * Assegnamento par.>var.
  local1 = param2;
                         MOVE.L
                                 12(A6), -4(A6)
                                                   * Elimino var. locali
                                 A6
                         UNLK
                                                   * Ritorno al chiamante
                         RTS
```

Esempio completo

• È a disposizione un esercizio completo che illustra i meccanismi di chiamata a sottoprogramma, interamente svolto.