Esercizio riassuntivo di traduzione e assemblaggio da linguaggio C ad assembly Motorola 68000.

Ultima modifica: 10 Maggio 2005.
Autore: Daniele Paolo Scarpazza.
Per segnalare eventuali difficoltà o problemi, scrivetemi a: daniele.scarpazza@elet.polimi.it.
Esercizio liberamente ispirato al tema d'esame del 15/7/2002, esercizio 2.

Prima parte

Codificate in linguaggio assembly Motorola 68000 il programma C riportato di seguito, rispettando le seguenti richieste, che indicano come svolgere la traduzione e quali convenzioni di chiamata applicare:

- usate propriamente i record di attivazione e le istruzioni per la loro gestione;
- dedicate il registro A6 a contenere il *frame pointer* e il registro A7 a contenere lo *stack pointer*;
- il chiamante impili i parametri in ordine inverso rispetto a come appaiono sul prototipo (detto esplicitamente: prima l'ultimo, poi il penultimo, ..., infine il primo);
- allocate le variabili locali nel record di attivazione secondo l'ordine in cui sono dichiarate: (se la variabile *b* è dichiarata dopo *a*, allora *b* apparirà nella pila "sopra" la variabile *a)*;
- è compito del chiamante abbattere la porzione di pila che contiene i parametri in ingresso per il chiamato, dopo aver completato la chiamata;
- la funzione chiamata deve salvare i registri che usa prima di usarli, e deve ripristinarli appena prima di ritornare;
- dedicate il registro D0 al valore di ritorno della funzione (non è necessario salvare il registro D0);
- traducete le espressioni realizzando la copertura a tegole dell'albero sintattico; in dettaglio:
  - numerate i nodi interni secondo una visita postordine;
  - ottenere una copertura dei nodi con le tegole mostrate a lezione;
  - non effettuare ottimizzazioni, anche quando il codice ottenuto è palesemente inefficiente;
  - allocate i registri a partire da D1 (lasciare D0 riservato al valore di ritorno);
  - se necessario, assumete un numero di registri illimitato (quindi non solo D0-D7 ma anche D8, D9, ...);
  - codificate le tegole in assembly secondo la numerazione indotta, salvo le eccezioni opportune (push di parametri);
- assumete che la dimensione degli interi sia di 32 bit (*long word*, secondo la terminologia M68000);
- non considerate la funzione main() come un sottoprogramma; traducete invece il suo codice all'entry point del programma, ignorando parametri e valori di ritorno;
- ponete l'origine del segmento codice all'indirizzo $1000.

Per l'espressione marcata dalla freccia, disegnate l'albero sintattico, svolgete la numerazione dei nodi, la copertura con tegole e l'allocazione dei registri.

Per le funzioni f1 ed f2, disegnate uno schema semplificato dello stato dello stack così come si trova immediatamente dopo la fase di salvataggio dei registri.

Il codice C da tradurre si trova nella pagina seguente.

```
int f2 (int a, int b)
{
	int c;

	c = a + 3;

	return (a + b) - c;
}

int f1 (int i)
{
	int j, r;

	j = 7;

	r = f2(i,j) + (i-f2(j,i)); /* <- */

	if (r != 3)
		r = i + r;
	else
		r = r - i;

	return r + j;
}

int main(){

	f1(11);

}
```

Soluzione – Prima Parte:

Nota: per agevolare la comprensione della soluzione, nel codice assembly abbiamo lasciato il codice C originale sotto forma di commento.

```
                                    *File: esercizio-1.X68
                                    ORG         $1000
*int f2 (int a, int b)
                                F2:
*{
*     int c;
                                    LINK        A6, #-4             * spazio var. loc.
                                    MOVEM.L     D1-D3,-(SP)         * salvo registri

*     c = a + 3;
                                    MOVE.L      8(A6), D1           * leggo a
                                    ADD.L       #3, D1              * sommo 3
                                    MOVE.L      D1, -4(A6)          * salvo c
*     return (a + b) - c;
                                    MOVE.L      8(A6), D1           * leggo a
                                    MOVE.L      12(A6), D2          * leggo b
                                    ADD.L       D2,D1               * sommo a,b
                                    MOVE.L      -4(A6), D3          * leggo c
                                    SUB.L       D3,D1               * sottraggo

                                    MOVE.L      D1, D0              * val. rit.
                                    MOVEM.L     (SP)+,D1-D3         * riprist. reg.
                                    UNLK        A6
                                    RTS
*}

*int f1 (int i)
                                F1:
*{
*     int j, r;
                                    LINK        A6, #-8             * spazio var. loc.
                                    MOVEM.L     D1-D7,-(SP)         * salvo registri
*     j = 7;
                                    MOVE.L      #7, -4(A6)          * assegno j
*     r = f2(i,j) + (i-f2(j,i));
                                    * prima chiamata alla funzione f2
                                    MOVE.L      8(A6), D1           * 1: leggo i
                                    MOVE.L      -4(A6), D2          * 3: leggo j
                                    MOVE.L      D2, -(A7)           * 4: push j
                                    MOVE.L      D1, -(A7)           * 2: push i
                                    JSR         F2                  * 5: chiamo f2
                                    MOVE.L      D0, D3              * leggo val. rit.
                                    ADDQ.L      #8, A7              * elimino param

                                    MOVE.L      8(A6), D4           * 6:leggo i

                                    * seconda chiamata alla funzione f2
                                    MOVE.L      -4(A6), D5          * 7: leggo j
                                    MOVE.L      8(A6), D6           * 9: leggo i
                                    MOVE.L      D5, -(A7)           * 8: push j
                                    MOVE.L      D6, -(A7)           * 10: push i
                                    JSR         F2                  * 11: chiamo f2
                                    MOVE.L      D0, D7              * leggo val. rit.
                                    ADDQ.L      #8, A7              * elimino param
```

```
                                        SUB.L       D7, D4          * sottraggo
                                        ADD.L       D3, D4          * sommo
                                        MOVE.L      D4, -8(A6)      * salvo r
*     if (r != 3)
                                        MOVE.L      -8(A6), D1      * leggo r
                                        CMP.L       #3, D1          * confronto r,3
                                        BEQ         ELSE_BRANCH
*           r = i + r;
                                        MOVE.L      -8(A6), D1      * leggo r
                                        MOVE.L      8(A6), D2       * leggo i
                                        ADD.L       D2, D1          * sommo
                                        MOVE.L      D1, -4(A6)      * salvo r
                                        BRA         IF_END
*     else
                                    ELSE_BRANCH:
*           r = r - i;
                                        MOVE.L      -8(A6), D1      * leggo r
                                        MOVE.L      8(A6), D2       * leggo i
                                        SUB.L       D2, D1          * sottraggo
                                        MOVE.L      D1, -4(A6)      * salvo r
                                    IF_END:
*     return r + j;
                                        MOVE.L      -8(A6), D1      * leggo r
                                        MOVE.L      -4(A6), D2      * leggo j
                                        ADD.L       D1, D2          * sommo
                                        MOVE.L      D2, D0          * val. rit.

                                        MOVEM.L     (SP)+,D1-D3     * riprist. reg.
                                        UNLK        A6
                                        RTS
*}

*int main(){
                                    START:
*     f1(11);
                                        MOVE.L      #11, D1
                                        MOVE.L      D1, -(A7)       * push parametro
                                        JSR         F1              * chiamo F1
                                        ADDQ.L      #4, A7          * elimino param

                                        STOP        #$2000
                                        END         START
*}
```

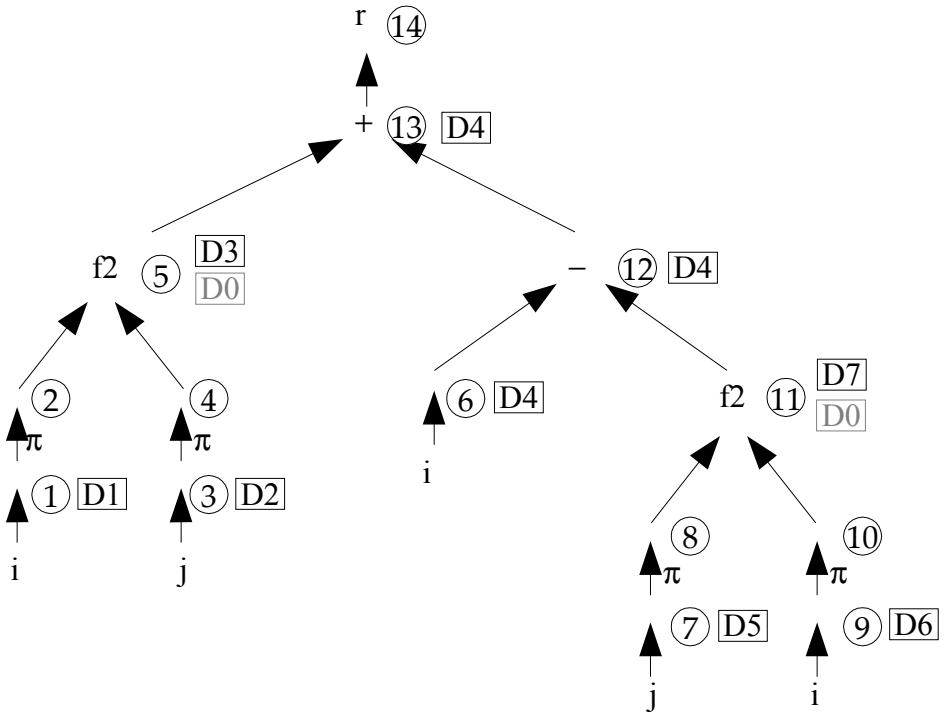
Schema dello stack per la funzione f2(), immediatamente dopo il salvataggio dei registri.

FP-16	:	[registro salvato D1	]	<SP
FP-12	:	[registro salvato D2	]	
FP-8	:	[registro salvato D3	]	
FP-4	:	[variabile locale c	]	
FP	:	[puntatore frame precedente	]	<FP
FP+4	:	[indirizzo di ritorno	]	
FP+8	:	[parametro a	]	
FP+12	:	[parametro b	]	

Schema dello stack per la funzione f1(), immediatamente dopo il salvataggio dei registri.

FP-36	:	[registro salvato D1	]	<SP
FP-32	:	[registro salvato D2	]	
FP-28	:	[registro salvato D3	]	
FP-24	:	[registro salvato D4	]	
FP-20	:	[registro salvato D5	]	
FP-16	:	[registro salvato D6	]	
FP-12	:	[registro salvato D7	]	
FP-8	:	[variabile locale r	]	
FP-4	:	[variabile locale j	]	
FP	:	[puntatore frame precedente	]	<FP
FP+4	:	[indirizzo di ritorno	]	
FP+8	:	[parametro 1	]	

Albero sintattico coperto (accenno) dell'espressione `r = f2(i,j) + (i-f2(j,i));`



Seconda parte

Assemblate il programma ottenuto nella prima parte.
Riportate nella tabella dei simboli, i simboli incontrati e il loro valore numerico.
Riportate nella tabella di assemblaggio il risultato dell'assemblaggio.

Nel calcolo degli ingombri seguite le regole esposte a lezione.
Assumete tutti gli spiazzamenti contenibili in 1 parola;
Assumete che tutte le costanti immediate occupino 32 bit, con la seguente eccezione: le istruzioni ADD e SUB, se accompagnate da operandi immediati piccoli (valori da 1 a 8) vengono codificate come ADDQ e SUBQ e occupano una sola parola.

Ricordate che MOVEM occupa 2 parole;
Ricordate che l'istruzione JSR codifica l'indirizzo del salto, mentre le istruzioni Bcc codificano lo spiazzamento rispetto alla parola che contiene lo spiazzamento stesso.

Soluzione – Seconda Parte:

```
Ind.  N.Par. Etich.      Istruzione          Operandi codificati
1000   2  F2:       LINK    A6, #-4
1004   2           MOVEM.L D1-D3,-(SP)
1008   2           MOVE.L  8(A6), D1
100C   1           ADD.L   #3, D1
100E   2           MOVE.L  D1, -4(A6)
1012   2           MOVE.L  8(A6), D1
1016   2           MOVE.L  12(A6), D2
101A   1           ADD.L   D2,D1
101C   2           MOVE.L  -4(A6), D3
1020   1           SUB.L   D3,D1
1022   1           MOVE.L  D1, D0
1024   2           MOVEM.L (SP)+,D1-D3
1028   1           UNLK    A6
102A   1           RTS

102C   2  F1:       LINK    A6, #-8
1030   2           MOVEM.L D1-D7,-(SP)
1034   4           MOVE.L  #7, -4(A6)
103C   2           MOVE.L  8(A6), D1
1040   2           MOVE.L  -4(A6), D2
1044   1           MOVE.L  D2, -(A7)
1046   1           MOVE.L  D1, -(A7)
1048   2           JSR     F2                  $1000
104C   1           MOVE.L  D0, D3
104E   1           ADDQ.L  #8, A7
1050   2           MOVE.L  8(A6), D4
1054   2           MOVE.L  -4(A6), D5
1058   2           MOVE.L  8(A6), D6
105C   1           MOVE.L  D6, -(A7)
105E   1           MOVE.L  D5, -(A7)
1060   2           JSR     F2                  $1000
1064   1           MOVE.L  D0, D7
1066   1           ADDQ.L  #8, A7
1068   1           SUB.L   D7, D4
106A   1           ADD.L   D3, D4
106C   2           MOVE.L  D4, -8(A6)
1070   2           MOVE.L  -8(A6), D1
1074   3           CMP.L   #3, D1
107A   2           BEQ     ELSE_BRANCH         $1090 - $107A - $2 = $14
107E   2           MOVE.L  -8(A6), D1
1082   2           MOVE.L  8(A6), D2
1086   1           ADD.L   D2, D1
1088   2           MOVE.L  D1, -4(A6)
108C   2           BRA     IF_END              $109E - $108C - $2 = $10
1090     ELSE_BRANCH:
1090   2           MOVE.L  -8(A6), D1
1094   2           MOVE.L  8(A6), D2
1098   1           SUB.L   D2, D1
109A   2           MOVE.L  D1, -4(A6)
109E     IF_END:
109E   2           MOVE.L  -8(A6), D1
10A2   2           MOVE.L  -4(A6), D2
10A6   1           ADD.L   D1, D2
10A8   1           MOVE.L  D2, D0
10AA   2           MOVEM.L (SP)+,D1-D3
10AE   1           UNLK    A6
10B0   1           RTS

10B2   1 START:    MOVE.L  #11, D1
```

```
10B4  1         MOVE.L  D1, -(A7)
10B6  2         JSR     F1                      $102C
10BA  1         ADDQ.L  #4, A7
10BC  2         STOP    #$2000
10C0            END     START
```

Tabella dei simboli:

Simbolo	Valore
F2	1000
F1	102C
ELSE_BRANCH	1090
IF_END	109E
START	10B2