Assemblaggio e collegamento

Daniele Paolo Scarpazza
Dipartimento di Elettronica e Informazione
Politecnico di Milano

3 Giugno 2004

Assemblaggio [§7.3]

- A prima vista problema banale:
  una riga di assembly => una istruzione
- in realtà: piccoli problemi, fra cui *riferimenti in avanti*, esempio:

```
            ...
            GOTO        L1
            ...
L1:         IINC A, 1
            ...
```

  per assemblare una istruzione di salto in avanti devo già conoscere a quale indirizzo corrisponde l'etichetta.
- Soluzioni possibili (con vantaggi e svantaggi):
  - assemblaggio in due passate (l'input viene letto 2 volte)
  - caricamento di tutto il listato in memoria in una forma intermedia, elaborazione della forma intermedia;

Primo passo

- Costruzione della tabella dei simboli:
  - coppie <nome, valore> di etichette e constanti;
  - l'assemblatore mantiene l'ILC *(instruction location counter),* variabile che contiene l'indirizzo dell'istruzione che sta assemblando in quell'istante;
    - inizializzo ILC a 0 all'inizio dell'assemblaggio;
    - ad ogni istruzione letta, incremento ILC della dimensione di quella istruzione
    - man mano che incontro le etichette, aggiungo nella tabella dei simboli la coppia:
      <nome etichetta, valore attuale di ILC>;

Primo passo: tabelle usate

- Tabella dei simboli:
  - potrebbe contenere informazioni aggiuntive (lunghezza del campo dati, informazioni di rilocazione, ...)
- Tabella delle pseudoistruzioni
- Tabella degli opcode:
  - indica come tradurre uno mnemonico in uno degli opcode corrispondenti a seconda degli argomenti; esempio:

Mnemonico	Primo operando	Secondo operando	Opcode (hex)	Lungh
AAA	---	---	37	1
ADD	registro EAX	immediato 32 bit	05	5
ADD	registro	registro	01	2
AND	registro EAX	immediato 32 bit	25	5
AND	registro	registro	21	2
...	...	...	...	...

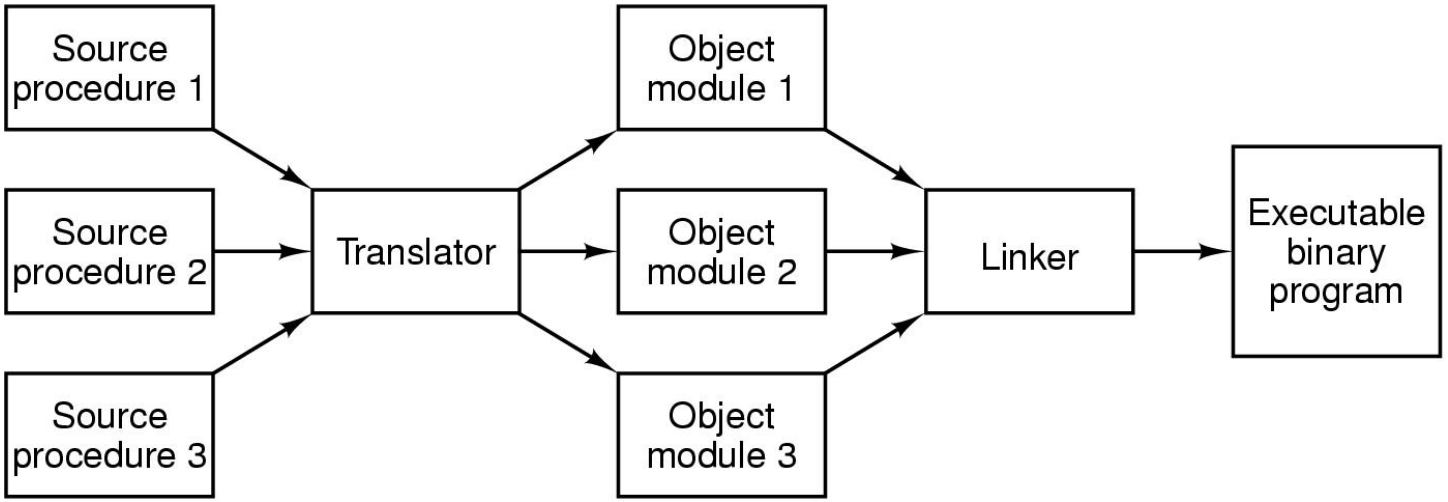
(cfr. Intel IA-32 Intel Architecture Software Developer's Manual, Volume 2, Appendix B.2)

Secondo passo

- Generazione e emissione del codice oggetto:
  - opcode e lunghezza di ogni istruzione sono già noti
  - completamento includendo i valori delle etichette
  - emissione del codice oggetto
- Generazione delle informazioni di rilocazione;

Collegamento [§7.4]

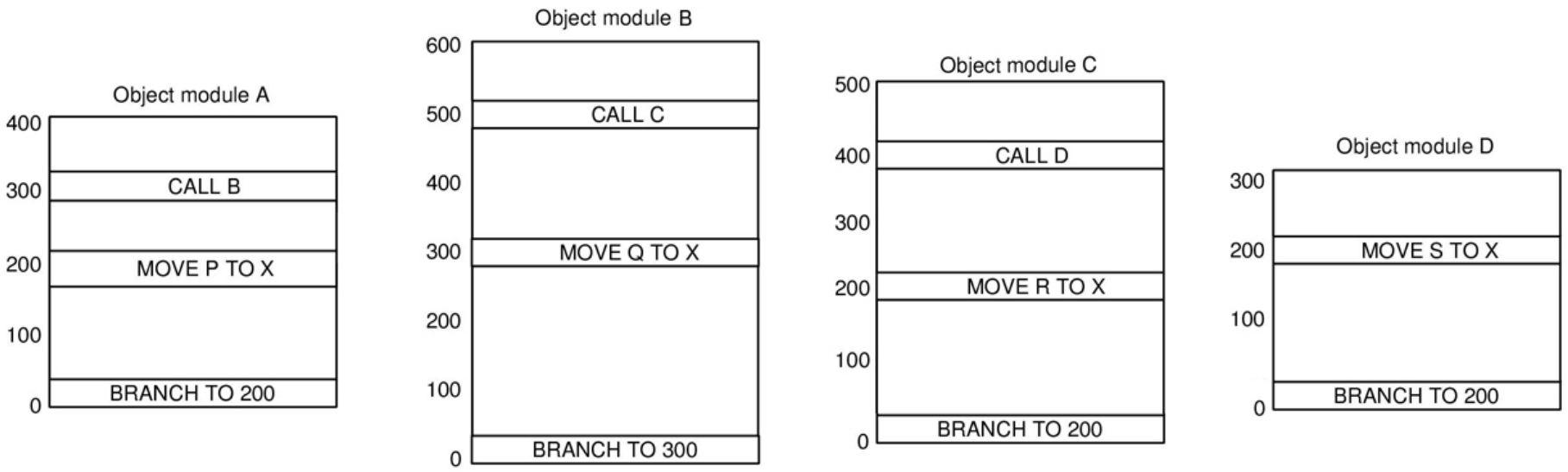
- I diversi moduli di sorgente assembly e i diversi sotto-programmi contenuti vengono assemblati separatamente l'uno dall'altro;
- è necessario collegare gli oggetti insieme, formando un unico eseguibile unitario;



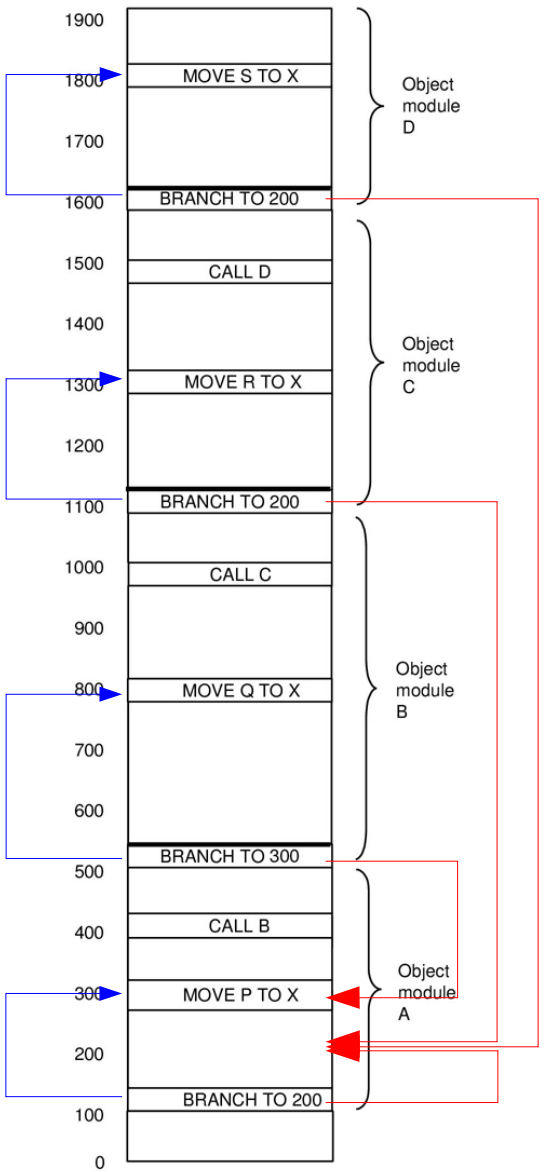
- lo strumento che si svolge le mansioni di collegamento si chiama *linker*, ed è separato dall'assemblatore (e.g.: **`as`**, **`ld`**).

Perché serve il linker

- L'ILC dell'assemblatore parte da 0 per ogni modulo: assume che ogni modulo sia caricato a partire dall'indirizzo 0, condizione non realizzabile;
  - perchè le prime pagine di RAM sono locazioni riservate;
  - perchè ci possono essere più moduli;
- Esempio: quattro moduli (A, B, C, D)
  - ciascuno ha come 1ª istruzione un salto all' *entry point*
  - all'entry point c'è una MOVE



Prima mansione: accodamento



- Accodamento in sequenza di tutti i moduli: non basta!
- Tutti i salti sono stati corrotti

  Legenda:
  - salti desiderati
  - salti effettivi

- Le chiamate a funzioni esterne sono ancora non risolte (prive di indirizzo);

Fusione dello spazio di indirizzamento

Fasi:

- costruzione della tabella dei moduli oggetto e delle relative lunghezze;
- assegnazione di un indirizzo di inizio ad ogni modulo;
- modifica di tutte le istruzioni di accesso alle memoria: aggiunta a tutti gli indirizzi di una *costante di rilocazione* corrispondente all'indirizzo di inizio del proprio modulo;
- modifica di tutte le istruzioni che fanno riferimento ad un altro modulo: aggiornamento con l'indirizzo corretto;