

Architettura dei Sistemi a. a. 2008-09

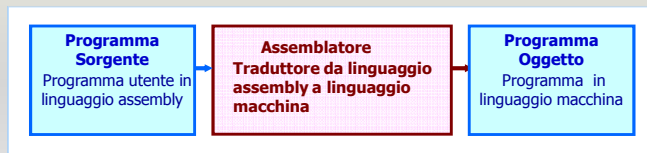
II) Architettura degli Elaboratori

- **Livello Linguaggio Assembly**

Livello di Linguaggio Assembly

□ Livello di linguaggio assembly

- ❖ Livello implementato tramite **traduzione**
- ❖ Il traduttore è chiamato **assemblatore**
- ❖ Il linguaggio assembly è essenzialmente la **rappresentazione simbolica** di un linguaggio macchina



- Quando il linguaggio sorgente è un linguaggio ad alto livello, come JAVA o C, e il linguaggio destinazione è un linguaggio macchina numerico oppure usa una sua rappresentazione simbolica il traduttore viene chiamato **compilatore**

Linguaggio Assembly

□ Linguaggio assembly

- ❖ Rappresentazione simbolica del linguaggio macchina
 - Insieme completo di **nomi simbolici** (codici mnemonici) per rappresentare specifiche stringhe di 0 e 1
 - Insieme di **regole** (sintassi) per l'uso dei codici mnemonici nello specificare istruzioni
- ❖ Proprietà del linguaggio assembly
 1. Possibilità di accesso a tutte le funzionalità e tutte le istruzioni della ISA
 - Se la macchina ha un bit di overflow, un programma in linguaggio assembly lo può controllare, quello in Java no
 - Il linguaggio C è un incrocio fra linguaggio ad alto livello e linguaggio assembly: la sintassi è quella di un linguaggio ad alto livello, ma fornisce possibilità di accesso alla macchina tipiche dei linguaggi assembly
 2. Sfruttamento ottimale delle caratteristiche tipiche della macchina da programmare

Linguaggio Assembly

- ❖ **Vantaggi** della programmazione in linguaggio assembly
 - Possibilità di produrre programmi compatti e quindi eseguibili più velocemente rispetto a programmi in linguaggio ad alto livello
 - Applicazioni in cui la velocità e la dimensione sono fattori critici: programmi di sistema quali i compilatori e le routine di I/O e **applicazioni embebbed**, come il codice di una smart card o di un telefono cellulare...
- ❖ **Svantaggi** della programmazione in linguaggio assembly
 - Programmi difficili da scrivere e leggere
 - Bisogna conoscere la organizzazione della macchina
 - Praticamente impossibile scrivere grandi programmi

Linguaggio Assembly

□ Formato di istruzione

Etichetta	Operazione	Operandi	Commento
-----------	------------	----------	----------

❖ Campi

- Parti del formato di istruzione, separate da un delimitatore opportuno (normalmente uno o più spazi)
 1. **Etichetta (Label)**
 - Codice mnemonico per riferimento a **istruzioni o dati**
 2. **Operazione**
 - Codice mnemonico per rappresentare un **codice operativo** o un **comando (direttiva)** all'assemblatore
 3. **Operandi**
 - Codici mnemonici per rappresentare gli **indirizzi** degli operandi
 4. **Commento**
 - Informazioni che non hanno effetto sul processo di assemblaggio

Linguaggio Assembly

❖ Label

- Il linguaggio assembly fornisce al programmatore la possibilità di definire **simboli utilizzati per riferire oggetti** (istruzioni, dati, segmenti...) presenti all'interno di un programma
- Una label può essere
 1. nome associato all' **indirizzo di memoria** in cui verrà inserita l'istruzione in linguaggio macchina generata a partire dalla istruzione in linguaggio assembly
 2. nome associato all' **indirizzo di un dato**

Il linguaggio assembly consente al programmatore di specificare informazioni indispensabili per la traduzione del programma sorgente in programma oggetto

Linguaggio Assembly

❑ Pseudoistruzione

- ❖ **Istruzione il cui codice operativo rappresenta un comando (direttiva)** all'assemblatore
 - Richiesta di assegnazione di spazio in memoria
 - Assegnazione di valori numerici ai nomi simbolici usati nel programma
 - Definizione di un nuovo nome simbolico
 - Definizione di macro e procedure
- ❖ Tali istruzioni non compaiono nel programma oggetto alla fine della fase di traduzione

VAR DW ? ; alloca spazio per una word alla quale è associato il nome VAR e per la quale non è specificato un valore iniziale
A DB 134 ; alloca spazio per un byte al quale è associato il nome A e per il quale è specificato il valore iniziale 134
SOMMA EQU 200 ; assegna il valore 200 al nome simbolico SOMMA

Linguaggio Assembly

❑ Macroistruzione

- ❖ **Pseudoistruzione che consente di assegnare un nome ad una sequenza di istruzioni usata frequentemente**
 - Definizione da parte del programmatore di una "nuova istruzione" a livello di linguaggio assembly
 - Tre fasi associate ad una macroistruzione:
 1. **Definizione:** solo le istruzioni presenti nella definizione sono tradotte in codice oggetto
 2. **Chiamata:** il nome della macroistruzione (label) è nel campo codice operativo
 3. **Espansione:** sostituzione, durante il processo di assemblaggio, delle istruzioni del corpo della macroistruzione indicate nella definizione

```
"label" MACRO  
  Istr.1  
  ....  
  Istr.k  
ENDM
```

Linguaggio Assembly

❖ Macro con parametri

- Si possono definire macro con **parametri formali** ed effettuare le chiamate con **parametri attuali**
- Quando la macro viene espansa i parametri formali vengono sostituiti dai corrispondenti parametri attuali

Definizione della macro che scambia il contenuto delle variabili P1 e P2

```
CHANGE    MOV EAX,P1
          MOV EBX,P2
          MOV P2,EAX
          MOV P1,EBX
          ENDM
```

Chiamate della macro con i parametri attuali

```
.....
          CHANGE P,Q
          .....
          CHANGE R,S
```

Linguaggio Assembly

□ Procedura

- ❖ **Pseudoistruzione** che consente di specificare il fatto che una sequenza di istruzioni costituisce una procedura
 - La dichiarazione di una procedura consente di associare al nome della procedura, che è a tutti gli effetti una etichetta, l'entry-point address
 - Il tipo definisce se la procedura è **near**, ovvero è definita all'interno dello stesso segmento di codice, o **far**, se è definita all'esterno
 - Se tale parametro non è specificato, viene assunto near
 - La direttiva **ENDP** è eseguita dall'assemblatore in modo differente a seconda del tipo di procedura:
 - se near la istruzione di ritorno preleva dallo stack solo il return address
 - se far la istruzione di ritorno preleva dallo stack il return address e l'indirizzo del suo segmento di codice

```
"Nome" PROC tipo
      Istr.1
      ....
      Istr.k
      ENDP
```

Linguaggio Assembly

□ Macroistruzioni e Procedure

❖ Chiamata

- La chiamata della macroistruzione è effettuata durante il processo di assemblaggio
- La chiamata della procedura è effettuata durante l'esecuzione del programma

❖ Corpo

- Il corpo della macroistruzione è ripetuto nel codice oggetto ad ogni chiamata
- Il corpo della procedura dà luogo ad un solo codice oggetto

Durante l'esecuzione del programma, nel caso di procedura vengono eseguite due istruzioni in più (la chiamata e il ritorno da procedura)

Assemblatore

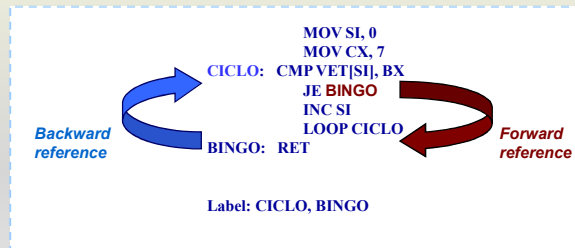
□ Assemblatore

- ❖ **Programma** che esegue il processo di traduzione da linguaggio assembly a linguaggio macchina
 - Sostituisce i **nomi** e i **simboli** con il valore che essi rappresentano
 1. i nomi simbolici per operazioni e modalità di indirizzamento sono sostituiti dai codici binari
 2. le etichette sono sostituite dai loro valori effettivi
- ❖ Il processo di traduzione da linguaggio assembly a linguaggio macchina è simile per le diverse macchine

Nel processo di traduzione si presenta il **problema delle Forward-References**, simboli referenziati prima di essere definiti

Assemblatore

❖ Riferimenti backward e forward ai simboli



- ❖ Per risolvere il problema dei simboli referenziati prima di essere definiti è necessario effettuare il **processo di traduzione in due passi**

Assemblatore

❖ Assemblatore a due passi

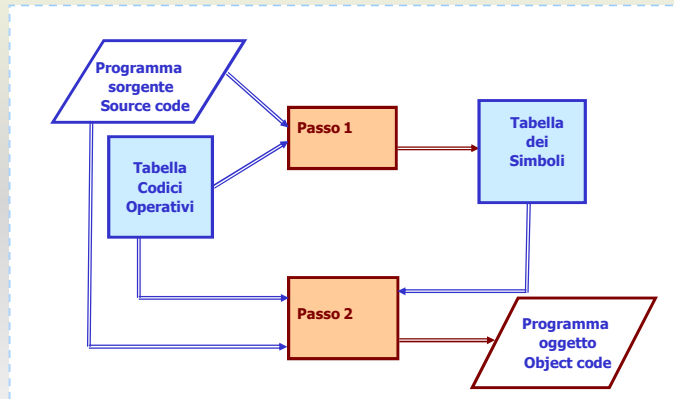
1. Primo passo

- Memorizzazione di tutti i simboli presenti nel programma sorgente e delle loro definizioni in una tabella (**Symbol Table**)
 - Un simbolo può essere definito come etichetta o mediante una pseudoistruzione che gli assegna un valore
- Assegnazione di un valore ad ogni simbolo, utilizzando una variabile detta **ILC (Instruction Location Counter)** che contiene l'indirizzo della istruzione che è in fase di assemblaggio

2. Secondo passo

- Traduzione di tutto il programma sorgente in programma oggetto, essendo noti i valori di tutti i simboli
- Generazione di informazioni utili per il linker
- Rilevazione di eventuali errori sintattici
 - uso di un simbolo non definito
 - simbolo definito più volte
 - codice operativo non consentito

Assemblatore

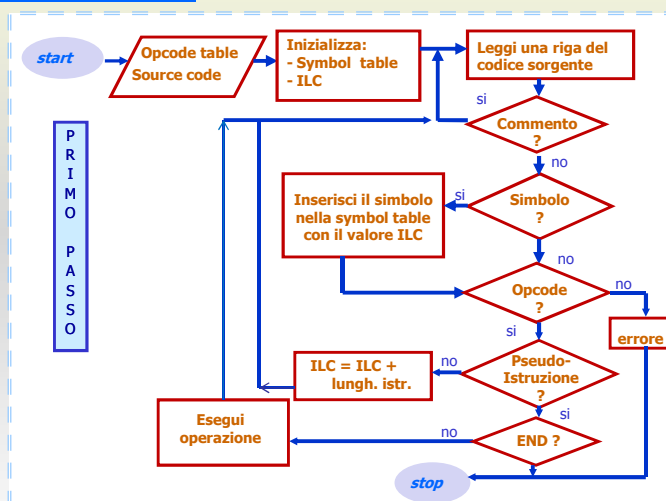


A.M. Fanelli

Architettura dei Sistemi a. a. 2008-09

15

Assemblatore



A.M. Fanelli

Architettura dei Sistemi a. a. 2008-09

16

Assemblatore

❖ Tabella dei simboli (Symbol Table)

- Tabella generata nel primo passo
- Ogni elemento della tabella dei simboli contiene il simbolo stesso, il suo valore numerico e altre informazioni come la lunghezza del campo dati associato al simbolo, i bit di rilocazione e la possibilità di accedere o meno al simbolo dall'esterno della procedura

Simbolo	Valore ILC	Altre informazioni
Ciclo	100	
Bingo	104	
IX	1500	

- Ciclo, Bingo: etichette per le istruzioni all'indirizzo 100 e 104
- IX: simbolo al quale è assegnato il valore 1500

Assemblatore

❖ Tabella dei codici (Opcode Table)

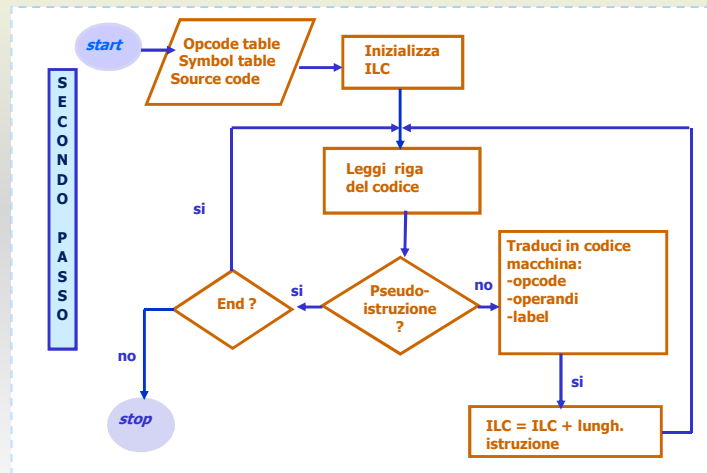
- Tabella con informazioni per ogni codice operativo simbolico del linguaggio assembly

Opcode simbolico	Opcode esadecimale	Lunghezza in byte	Tipo istruzione
Mov	5A	4	1
Inc	21	2	2
Je	1C	1	4

• Tipo istruzione

- Indicatore di tipo che raggruppa gli opcode a seconda del numero e tipo di operandi

Assemblatore



A.M. Fanelli

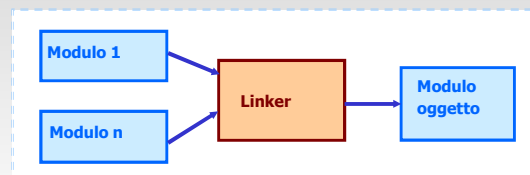
Architettura dei Sistemi a. a. 2008-09

19

Linker

□ Linker

- ❖ Programma che individua e **collega i moduli** (procedure), tradotti separatamente dall'assemblatore, di un programma
 - Ogni modulo oggetto ha il suo spazio di indirizzamento separato
 - Il processo di collegamento delle procedure tradotte in linguaggio macchina avviene a livello di linguaggio assembly
- ❖ Il linker esegue la funzione di "collegamento" dei **moduli oggetto** in modo da formare un unico **modulo oggetto**
 - Windows: moduli oggetto (*.obj) e modulo eseguibile (*.exe)
 - UNIX: moduli oggetto (*.o) e modulo eseguibile (nessuna estensione)



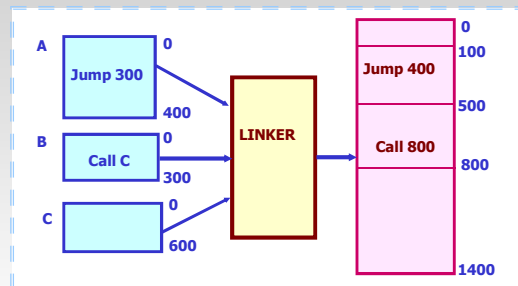
A.M. Fanelli

Architettura dei Sistemi a. a. 2008-09

20

Linker

- ❖ Il linker, per formare il modulo oggetto, fonde gli spazi di indirizzamento separati dei moduli oggetto in un unico **spazio di indirizzamento lineare**
- ❖ Il linker
 1. crea la **tabella dei moduli oggetto** e la loro lunghezza
 2. in base a tale tabella assegna un **indirizzo di inizio ad ogni modulo** oggetto
 3. riloca gli spazi di indirizzamento (**Rilocazione**) dei vari moduli, ossia in tutte le istruzioni che contengono un indirizzo di memoria somma a ciascun indirizzo una **"costante di rilocazione"** uguale all'indirizzo di inizio del modulo in cui la istruzione è contenuta
 4. assegna nella istruzione di chiamata ad una procedura (**External Reference**) l'indirizzo di partenza della procedura stessa



A. M. Fanelli

Architettura dei Sistemi a. a. 2008-09

21

Linker

- ❖ **Struttura di un modulo oggetto**

Nome e lunghezza del modulo
Tabella dei simboli usati nel modulo ma definiti in altri moduli e delle external reference
Istruzioni in linguaggio macchina
Dizionario di rilocazione
Fine modulo

- ❖ **Collegamento dinamico**
 - Tecnica per collegare procedure compilate separatamente nel momento in cui ciascuna procedura viene effettivamente chiamata
 - Molti programmi contengono procedure chiamate solo in particolari circostanze
 - Le DLL di Windows usano il collegamento dinamico
 - DLL (Dynamic Link Library)
 - Libreria composta da procedure che possono essere caricate in memoria e alle quali possono accedere più processi nello stesso istante

A. M. Fanelli

Architettura dei Sistemi a. a. 2008-09

22

Loader

❑ Loader

- ❖ Programma che esegue il processo di **caricamento del modulo oggetto in memoria**
- ❖ Determina il legame (**binding**) tra gli indirizzi logici del programma binario prodotto dal linker e gli indirizzi fisici della memoria in modo da fornire il **modulo assoluto o eseguibile**
- ❖ Meccanismi per facilitare la mappatura degli indirizzi logici in indirizzi fisici
 - Segmentazione, paginazione

Linker-Loader

