AXO – Architettura dei Calcolatori e Sistema Operativo

processo di assemblaggio

linguaggio assembly

- è il **linguaggio *simbolico*** che consente di programmare un calcolatore utilizzando le istruzioni del linguaggio macchina
- per essere eseguito, il programma scritto in ASSEMBLY va **tradotto in linguaggio macchina binario**, così da tradurre i codici mnemonici delle istruzioni in codici operativi, sostituire tutti i riferimenti simbolici degli indirizzi con la corispondente forma binaria, e riservare spazio di memoria per le variabili
- l'operazione di traduzione viene eseguita dall'ASSEMBLATORE
- se nel codice sorgente sono presenti riferimenti simbolici definiti in moduli esterni a quello assemblato, è necessario anche il LINKER (collegatore)

processo di assemblaggio - 1

- è un procedimento sequenziale che esamina, riga per riga, il codice sorgente assembler di ogni modulo (o del programma)
- è applicato modulo per modulo al programma e costruisce per ogni modulo la Tabella dei Simboli del modulo, generando il codice oggetto
- traduce i codici mnemonici delle istruzioni nei corrispondenti codici binari
- traduce i riferimenti simbolici del modulo (variabili, etichette di salto … nomi di funzioni) nei corrispondenti indirizzi numerici
  - i riferimenti simbolici esterni (p. es. simboli definiti in altri moduli - come nomi di sottoprogrammi esterni - o simboli globali) possono essere "memorizzati" nella Tabella dei Simboli esterni o globali
- ogni modulo assemblato di default parte dall'indirizzo 0; le direttive ORG modificano tale indirizzo. In sistemi dotati di meccanismi di memoria virtuale (MMU) tali indirizzi sono *indirizzi virtuali.*
- le tabelle utilizzate sono
  - la **tabella dei simboli del modulo**
  - la **tabella dei codici operativi e degli ingombri delle istruzioni**

ingombro per istruzioni che manipolano dati

- dimensione della parola di memoria: 16 bit
- dimensione della costante: 8, 16 o 32 bit
- dimensione dello spiazzamento: 8 o 16 bit
- dimensione dell'indirizzo di memoria: 16 o 32 bit
- l'istruzione è sempre costituita da una parola da 16 bit iniziale, detta "parola di codice operativo", cui si aggiungono eventualmente altre parole consecutive, dette "parole aggiuntive o di estensione"
  - l'ingombro max dell'istruzione è di 5 parole da 16 bit (1 parola di codice operativo + 4 parole di estensione)
- il codice operativo occupa una frazione breve della parola di codice operativo, spesso (ma non sempre) incorpora il segnale di dimensione del dato, e lascia spazio per argomenti non troppo lunghi
- eventuali argomenti (o parti di argomenti) di tipo costante, indirizzo o spiazzamento, occupano ciascuno una parola aggiuntiva oppure due parole aggiuntive a quella di codice operativo

istruzione (a indirizzo 0)		parola di cod. op	parole aggiuntive		parole totali
			arg1	arg2	
MOVE.B	#$01, D1	1	1	0	2
MOVE.W	#$0123, D1	1	1	0	2
MOVE.L	#$01234567, D1	1	2	0	3
MOVE.X	A1, D1	1	0	0	1
MOVE.X	(A1), D1	1	0	0	1
MOVE.X	(A1)+, D1	1	0	0	1
MOVE.X	-(A1), D1	1	0	0	1
MOVE.X	$0123 (A1), D1	1	1	0	2
MOVE.X	$01 (A1, D2.Y), D1	1	1	0	2
MOVE.X	$01 (A1, D2.Y), $23 (A3, A4.Y),	1	1	1	3
MOVE.X	$0123.W, D1	1	1	0	2
MOVE.X	$01234567.L, D1	1	2	0	3
MOVE.X	$0123.W, $456789AB.L	1	1	2	4
MOVE.W	#$0123, $456789AB.L	1	1	2	4
MOVE.X	$01234567.L, $89ABCDEF.L	1	2	2	5
MOVE.L	#$01234567, $89ABCDEF.L	1	2	2	5
ADD.X	D1, D2	1	0	0	1
ADD.X	$0123.W, D2	1	1	0	2
ADDX.	$01234567.L, D2	1	1	1	3
LSL.X	#1, D1	1	0	0	1

ingombro per istruzioni che manipolano indirizzi

- dimensione della parola di memoria: 16 bit
- dimensione dello spiazzamento: 8 o 16 bit
- dimensione dell'indirizzo di memoria: 16 o 32 bit
- l'istruzione è costituita da una parola da 16 bit iniziale, detta "parola di codice operativo", cui si aggiungono eventualmente altre parole consecutive, dette "parole aggiuntive o di estensione"; l'ingombro max dell'istruzione è di 3 parole da 16 bit (1 parola di codice operativo + 2 parole di estensione)
- il codice operativo occupa una frazione della parola di codice operativo, e lascia spazio per argomenti non troppo lunghi da codificare, vale a dire
  - modo di indirizzamento della destinazione
  - se la destinazione nomina un solo registro, il numero del registro indirizzo
  - eventuale condizione di salto
  - spiazzamento corto, cioè da 8 bit, in caso di indirizzamento relativo a PC (ma senza registro base, vedi sotto); si può usare solo con i salti di tipo B o branch (BRA e simili), non con quelli di tipo J o jump (JMP e simili)
- eventuali argomenti (o parti di argomenti) di tipo indirizzo o spiazzamento (da 16 bit oppure da 8 bit ma se c'è anche registro base), occupano ciascuno una parola oppure due parole aggiuntive a quella di codice operativo, secondo i casi. In particolare:
  - spiazzamento
    - se lungo occupa un'intera parola aggiuntiva da 16 bit
    - se corto ma in modo con base relativo a PC, occupa metà di una parola aggiuntiva, e l'altra metà contiene la specifica del registro base, come con l'analogo indirizzamento di dato
  - indirizzo: se corto (16 bit) occupa una parola, se lungo (32 bit) bit occupa due parole

esempi concreti

istruzione (all'indirizzo 0)	parola di cod. op	parole aggiuntive	parole totali
		dest	
JMP (A1)	1	0	1
BRA $12 (PC)	1	0	1
Bcc $12 (PC)	1	0	1
BRA $1234 (PC)	1	1	2
Bcc $1234 (PC)	1	1	2
JMP $12 (PC, A1)	1	1	2
JMP $1234.W	1	1	2
JMP $12345678.L	1	2	3

processo di assemblaggio - 2

- **etichette di salto**: causano il problema dei *riferimenti in avanti* che generalmente viene risolto tramite **traduzione in 2 passi;** l'operazione di traduzione si appoggia alle due tabelle viste prima

```
                ...
                CMP.L  #0, D0          // test whether D0 >= 0
                BGE    LABEL_POS       // jump to LABEL_POS if true
                NEG.L  D0              // else invert D0
LABEL_POS       ...
```

- per assemblare l'istruzione di salto in avanti devo già sapere a quale indirizzo corrisponde l'etichetta

processo di assemblaggio - 3

primo passo dell'Assemblatore

- costruzione della **Tabella dei Simboli del modulo**: la tabella contiene i riferimenti simbolici presenti nel modulo da tradurre e, al termine del primo passo, conterrà gli indirizzi numerici di tutti i simboli, tranne quelli esterni al modulo in esame
  - per le **etichette** associate a direttive dell'assemblatore che definiscono **costanti** simboliche (EQU) nella tabella dei simboli viene creata la coppia < etichetta, valore > e in ogni istruzione che fa riferimento al simbolo viene sostituito il valore
  - per **etichette** che definiscono **variabili** (spazio di memoria + eventuale inizializzazione), l'assemblatore riserva spazio, eventualmente inizializza la zona di memoria e crea nella tabella la coppia < etichetta, indirizzo >. In ogni istruzione che fa riferimento al simbolo viene sostituito l'indirizzo
  - nelle **etichette** presenti nelle **istruzioni di salto**, l'assemblatore deve generare un riferimento all'indirizzo dell'istruzione destinazione di salto

processo di assemblaggio - 4

primo passo dell'Assemblatore

- nella prima passata, l'assemblatore associa ad ogni istruzione la sua lunghezza in byte (ingombro) e l'indirizzo (d'inizio) dell'istruzione stessa
- in un'istruzione di salto, se l'etichetta non è ancora risolta (cioè non è presente nella tabella dei simboli), l'istruzione viene marcata "non risolta" e potrà venire tradotta completamente solo nella seconda passata
- quando l'assemblatore trova un'etichetta che identifica un'istruzione (destinazione di salto) nella tabella dei simboli viene creata la coppia < etichetta, indirizzo dell'istruzione >
- **etichette nelle istruzioni di salto**
  - se l'indirizzamento è relativo al contatore di programma (maggioranza dei casi) il valore dell'indirizzo numerico, presente nelle istruzioni di salto, deve rappresentre **lo spiazzamento** (positivo e negativo) tra l'indirizzo dell'istruzione di salto e l'indirizzo dell'istruzione destinazione di salto
  - per salti all'indietro l'etichetta può essere risolta nella prima passata calcolando la differenza tra indirizzo della destinazione di salto (presente nella tabella dei simboli) e indirizzo dell'istruzione di salto

processo di assemblaggio - 5

secondo passo dell'Assemblatore

- per salti in avanti questa operazione può essere fatta solo nella seconda passata esaminando le istruzioni “non risolte” e utilizzando la tabella dei simboli
- il programma viene scandito nuovamente e, tramite la Tabella dei Simboli del modulo, si calcolano gli spiazzamenti per le istruzioni di salto
  - nota bene: se il programma è costituito da più moduli assemblati separatamente, dopo l’assemblaggio di ogni modulo possono rimanere simboli non risolti (quelli cioè che sono definiti in un altro modulo)
- viene generato il programma in codice oggetto

processo di collegamento

- la fase di assemblaggio può generare più di un modulo oggetto (p. es. uno per ogni sottoprogramma) oppure il programma può far uso di moduli di libreria, che sono quindi esterni al programma stesso
- il **linker** o **collegatore** ha il compito di generare dai diversi moduli oggetto un **unico programma binario eseguibile**
  - si può anche dire che il linker ha il compito di generare *un solo spazio di indirizzamento* per tutto il programma
- in base alla lunghezza di ciascun modulo tradotto, è possibile calcolare l'indirizzo d'inizio di ogni modulo nel programma eseguibile e sostituire i riferimenti non risolti (per esempio nelle chiamate a sottoprogramma e nelle costanti globali) dell'intero programma
- anche il linker può far uso di opportune tabelle per il processo di collegamento (p. es. Tabella dei riferimenti globali)