



Linguagem de Montagem



Renato Ferreira


1



Overview

- Introdução ao Assembly
- Macro e Subrotina
- O Processo de Montagem
- Linkador e Carregador


2



Introdução

- Presente em quase todos computadores modernos
 - Característica do processador empregado
 - Linguagem assembly
- Abstração simbólica da linguagem de máquina
- Implementada por tradução
 - Tradução direta: assembler
- Execução em dois passos
 - Geração do código de máquina
 - Execução do novo programa diretamente pelo hardware

3



Características

- Linguagem assembly pura
 - Cada comando corresponde a 1 instrução de máquina
- Construção de alto nível
 - É mais simples programar em assembly do que diretamente em linguagem de máquina
 - Uso de nomes e endereços simbólicos
 - Instruções mnemônicas: ADD, SUB ...
- Acesso a todas as características do hardware alvo
 - Tudo que pode ser feito na máquina, pode ser feito em assembly
- Restrita à família de processadores alvo

4

Por que Assembly?

- Programar em assembly é difícil
 - Demora mais pra escrever um programa
 - Depuração e manutenção complicadas
- Duas razões:
 - Desempenho
 - Acesso

5

Desempenho

- Em termos de tamanho
 - Crítico para sistemas embutidos
- Em termos de velocidade
 - Lei 90 - 10
 - Implementar partes críticas em assembly
- Experiências práticas: MULTICS
 - Função reescrita em 3 meses:
 - 26 vezes menor, 50 vezes mais rápida
 - Função reescrita em 2 meses:
 - 20 vezes menor, 40 vezes mais rápida

6

Acesso

- Algumas tarefas exigem acesso direto ao hardware
 - Tratamento de interrupções do hardware
 - Controladores de dispositivos
- Sistemas modernos (Linux)
 - Acesso às funções do processador para troca de contexto
 - Boot

7

Razões

- Aspecto desempenho
 - Compiladores atuais são muito eficientes
 - Geram código de boa qualidade
 - Homem x máquina
 - Hardware está ficando muito complicado
- Aspecto acesso – motivo mais importante
- Estudo de arquiteturas
 - Assembly está diretamente relacionada ao hardware
 - Permite entender como a máquina realmente trabalha do ponto de vista do hardware

8

Formato das Instruções

- Está diretamente associado ao hardware
 - Porém, possuem características comuns
- Instruções Assembly
 - Representam os comandos da máquina
 - 4 campos
- Pseudo-instruções
 - Usadas para reservar espaço para dados
 - Apenas tipos básicos

9

Exemplos

Label	Opcode	Operands	Comments
FORMULA:	MOV	EAX,I	! register EAX = I
	ADD	EAX,J	! register EAX = I + J
	MOV	N,EAX	! N = I + J
I	DW	3	! reserve 4 bytes initialized to 3
J	DW	4	! reserve 4 bytes initialized to 4
N	DW	0	! reserve 4 bytes initialized to 0
(a)			
Label	Opcode	Operands	Comments
FORMULA:	MOV.EI	I, D0	! register D0 = I
	ADD.L	J, D0	! register D0 = I + J
	MOV.EI	D0, N	! N = I + J
I	DC.L	3	! reserve 4 bytes initialized to 3
J	DC.L	4	! reserve 4 bytes initialized to 4
N	DC.L	0	! reserve 4 bytes initialized to 0
(b)			
Label	Opcode	Operands	Comments
FORMULA:	SETHI	!%I(I),%R1	! R1 = high-order bits of the address of I
	LD	[%R1+%LO(I)],%R1	! R1 = I
	SETHI	!%H(J),%R2	! R2 = high-order bits of the address of J
	LD	[%R2+%LO(J)],%R2	! R2 = J
	NOP		! wait for J to arrive from memory
	ADD	%R1,%R2,%R2	! R2 = R1 + R2
	SETHI	!%H(N),%R1	! R1 = high-order bits of the address of N
	ST	!%R2,%R1+%LO(N)	
I:	WORD 3		! reserve 4 bytes initialized to 3
J:	WORD 4		! reserve 4 bytes initialized to 4
N:	WORD 0		! reserve 4 bytes initialized to 0
(c)			

10

Campo de label

- Usado para atribuir um nome simbólico para uma variável ou endereço
 - Campo opcional
- Formato
 - Posição fixa ou separador
 - Tamanho fixo ou variável
- Nomes de registradores
 - Declarações implícitas do assembler

11

Campo de operação

- Representações simbólicas das instruções do hardware
 - Critério de quem fez o montador
 - Intel: MOV
 - Motorola: MOVE
 - Sun: ST e LD
 - Variantes de operações
 - tamanho do dado tratado

12

Campo de operandos

- Especifica os alvos das operações
 - Endereços
 - instruções de desvio
 - Variáveis
 - posição de memória sendo acessada
 - Constantes
 - valores para operações aritméticas
 - Registradores
 - variáveis locais do processador

13

Campo de comentário

- Usado para acrescentar informações relevantes ao programa
 - Facilitar a compreensão do mesmo
 - Programa assembly é praticamente incompreensível sem esse campo
 - Consumo humano
- Formato geral
 - Indicador de início (;)
 - Até o final da linha

14

Pseudo-instruções

- No campo de operação
 - Instrução para o montador
- Diretivas do montador
- Relacionadas com o modo de operação da arquitetura
- Não correspondem a instruções de máquina
 - Alocação de variáveis
 - Criação de macros e subrotinas
 - Definição de escopo
 - Definição de segmentos
 - Constantes

15

Definição de Macro

- Solução para resolver o problema da necessidade de se repetir um bloco de instruções dentro de um programa:
 - Transformar este bloco em um procedimento.
 - Chamar este procedimento ao invés de reescrever as instruções.
- Necessita ser declarada previamente.
- É uma pseudo-instrução.

16

Exemplo

```
MOV EAX,P          SWAP MACRO
MOV EBX,Q          MOV EAX,P
MOV Q,EAX          MOV EBX,Q
MOV P,EBX          MOV Q,EAX
                  MOV P,EBX
                  ENDM

MOV EAX,P          SWAP
MOV EBX,Q          SWAP
MOV Q,EAX
MOV P,EBX

(a)                (b)
```

17

Chamada e Expansão

- É necessário:
 - Cabeçalho com o nome da macro;
 - Bloco de instruções;
 - Pseudo-instrução indicando o término.
- O montador salva a macro em uma tabela e quando esta é chamada, *substitui* a chamada pelo código.

18

Chamada e Expansão

- A expansão ocorre durante o processo de montagem e não durante a execução do programa.
- O código gerado pelo exemplo (a) é o mesmo do gerado pelo exemplo (b).
- Assembler efetuado em 2 passos:
 - As definições de macro são salvas e todas as operações são realizadas
 - Código é processado como se fosse o original.

19

Macros com parâmetros

```
MOV EAX,P          CHANGE MACRO P1, P2
MOV EBX,Q          MOV EAX,P1
MOV Q,EAX          MOV EBX,P2
MOV P,EBX          MOV P2,EAX
                  MOV P1,EBX
                  ENDM

MOV EAX,R          CHANGE P, Q
MOV EBX,S          CHANGE R, S
MOV S,EAX
MOV R,EBX

(a)                (b)
```

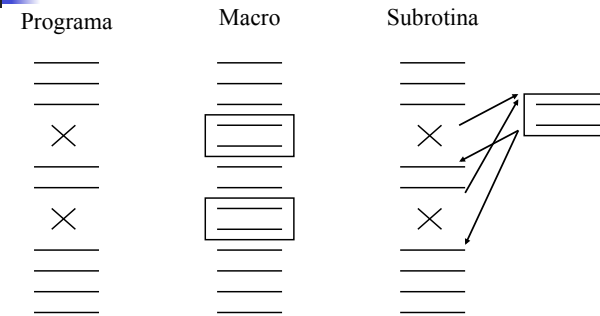
20

Subrotina

- É uma chamada a procedimento.
- Onde houver uma chamada, o procedimento é invocado separadamente e ao seu término, retorna para o programa que originou esta chamada.
- Ocorre um *desvio* no código.

21

Macro x Subrotina



22

O Processo de Montagem

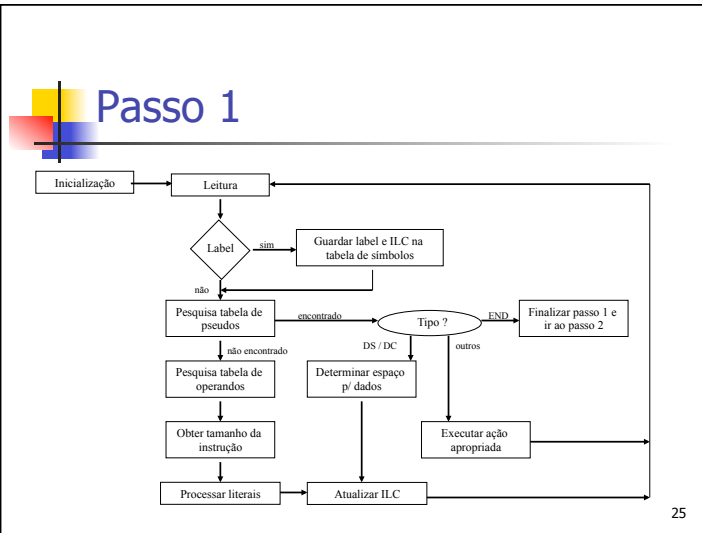
- Tradução imediata nem sempre é possível
 - Problema de referência posterior
- Tradução em dois passos
 - Alternativa 1:
 - armazenar as referências em uma tabela
 - traduzir o programa
 - solução simples
 - Alternativa 2:
 - gerar código intermediário
 - editar referências desconhecidas
 - pode ser mais eficiente

23

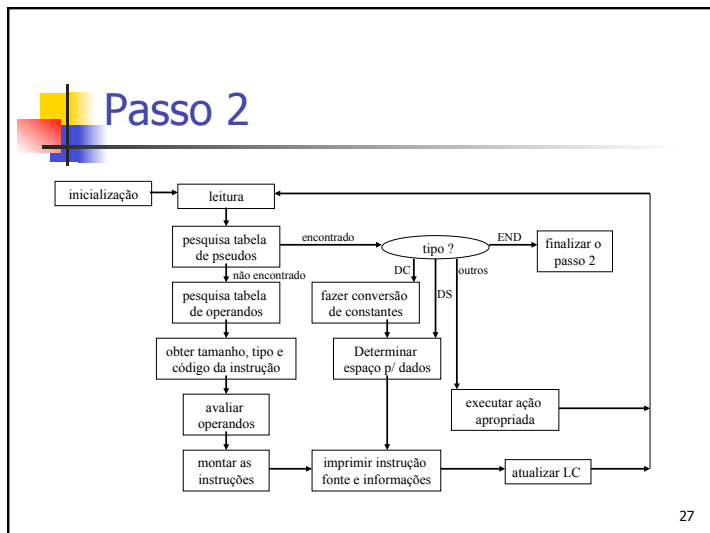
Passo 1

- Construção da tabela de símbolos
 - Labels
 - Valores produzidos por pseudo-instruções
- ILC – Instruction Location Counter
 - Usado para determinar endereço em tempo de execução dos símbolos
 - Inicializado com 0
 - Incrementado do tamanho de cada instrução traduzida

24





- ## Passo 2
- Executar a tradução do programa
 - Informação extra para o carregador
 - Leitura sequencial do código
 - Acessos às tabelas
 - de instruções
 - de símbolos
 - Tratamento e erros
 - Símbolos desconhecidos
- 26



- ## Tabela de símbolos
- Criada no passo 1 para utilização no passo 2
 - Informações coletadas sobre símbolos
 - tamanho do campo
 - informação de relocação
 - regras de escopo
 - Várias maneiras de organizar a tabela
 - Memória associativa: par (símbolo,valor):
 - Forma mais simples: vetor de registros
 - Forma mais esperta: tabela hash
- 28


Ligação e carga



- A maioria dos programas é constituída por mais de um procedimento.
- Os montadores traduzem um procedimento por vez, gerando um programa-objeto.
- Para um programa rodar, todos os procedimentos devem estar ligados.
- Como resolver isto?

29


Linker



- Funções:
 - Alocação: aloca espaço na memória para o programa.
 - Ligação: une os procedimentos e resolve as referências entre os módulos.
 - Relocação: ajusta os endereços que dependem da posição do programa na memória.

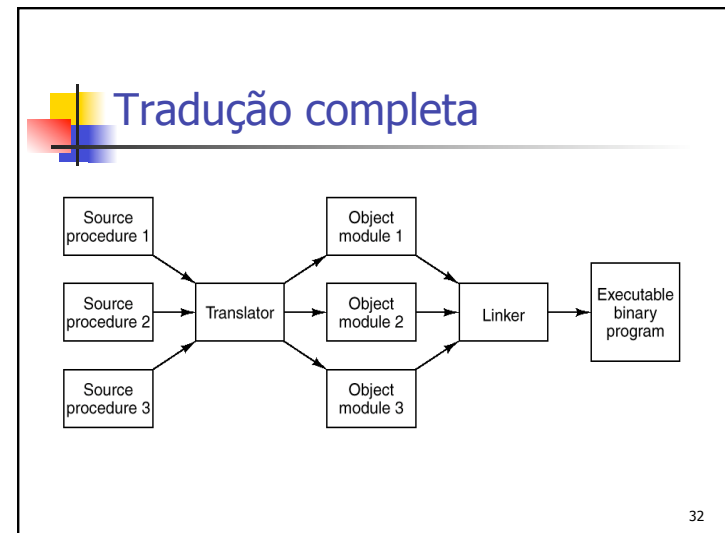
30

Tradução completa

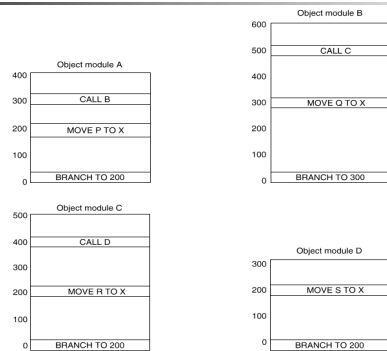


- Constituída de duas fases:
 - 1. Montagem dos procedimentos.
 - 2. Ligação dos módulos-objetos.
- Ao final, teremos um programa binário executável.

31

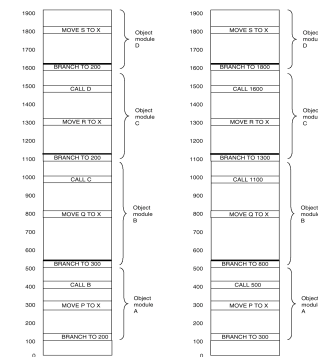


Exemplo de ligação



33

Exemplo de ligação



34

Problemas para o linker

- Problema da relocação:
 - Após a ligação dos módulos, os endereços devem ser recalculados.
- Problema da referência externa:
 - Definir o endereço dos módulos-objetos.

35