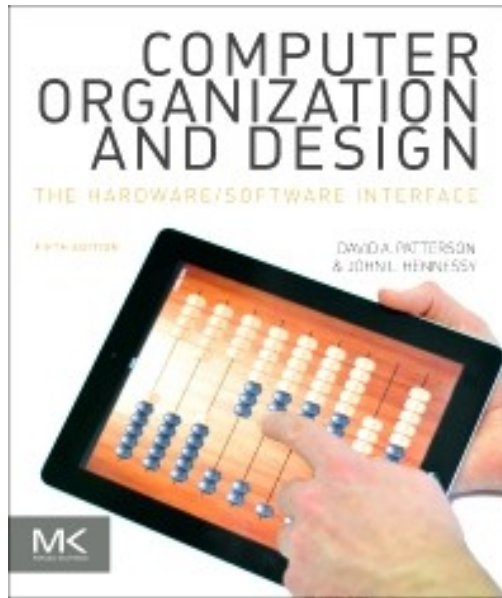


Livro-texto

Computer **Organization** and Design: The Hardware/Software Interface, **5th edition**

- Autores: David Patterson and John Hennessy
- **MIPS edition** (Não ARM edition! Nem RISC V edition!)
- Exemplares na BU: **4th edition**



Introdução à programação de sistemas

Introdução

- Codificação **binária** de instruções
 - Natural e eficiente para os computadores
 - » Números binários
- Codificação **simbólica** de instruções
 - Mais apropriada aos seres humanos
 - » Mnemônicos

Representações de código: linguagem de máquina

```
00100111101111011111111111100000
101011111011111110000000000010100
10101111101001000000000000100000
10101111101001010000000000100100
1010111110100000000000000011000
1010111110100000000000000011100
1000111110101110000000000011100
1000111110111000000000000011000
0000000111001110000000000011001
0010010111001000000000000000001
0010100100000001000000001100101
1010111110101000000000000011100
0000000000000000011110000010010
0000001100001111100100000100001
0001010000100000111111111110111
1010111110111001000000000011000
0011110000000100000100000000000
1000111110100101000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
1000111110111111000000000010100
00100111101111010000000000100000
000000111110000000000000001000
0000000000000000000100000100001
```

Calcula e imprime a soma dos quadrados dos inteiros de 0 a 100

Representações de código: linguagem de montagem

Operação

```
addiu    $29, $29, -32
sw       $31, 20($29)
sw       $4, 32($29)
sw       $5, 36($29)
sw       $0, 24($29)
sw       $0, 28($29)
lw       $14, 28($29)
lw       $24, 24($29)
multu    $14, $14
addiu    $8, $14, 1
slti     $1, $8, 101
sw       $8, 28($29)
mflo     $15
addu     $25, $24, $15
bne      $1, $0, -9
sw       $25, 24($29)
lui      $4, 4096
lw       $5, 24($29)
jal      1048812
addiu    $4, $4, 1072
lw       $31, 20($29)
addiu    $29, $29, 32
jr       $31
move     $2, $0
```

Registrador
Constante

Calcula e imprime a soma dos quadrados dos inteiros de 0 a 100

Representações de código: linguagem de montagem

```
.text
.align    2
.globl    main

main:
    subu    $sp, $sp, 32
    sw      $ra, 20($sp)
    sd      $a0, 32($sp)
    sw      $0, 24($sp)
    sw      $0, 28($sp)

loop:
    lw      $t6, 28($sp)
    mul     $t7, $t6, $t6
    lw      $t8, 24($sp)
    addu    $t9, $t8, $t7
    sw      $t9, 24($sp)
    addu    $t0, $t6, 1
    sw      $t0, 28($sp)
    ble     $t0, 100, loop
    la      $a0, str
    lw      $a1, 24($sp)
    jal     printf
    move    $v0, $0
    lw      $ra, 20($sp)
    addu    $sp, $sp, 32
    jr      $ra

.data
.align    0
str:
.asciiz    "The sum from 0 .. 100 is %d\n"
```

Diretivas

Registrador

Representações de código: linguagem de montagem

```
.text
.align 2
.globl main
main:
    subu    $sp, $sp, 32
    sw      $ra, 20($sp)
    sd      $a0, 32($sp)
    sw      $0, 24($sp)
    sw      $0, 28($sp)
loop:
    lw      $t6, 28($sp)
    mul     $t7, $t6, $t6
    lw      $t8, 24($sp)
    addu    $t9, $t8, $t7
    sw      $t9, 24($sp)
    addu    $t0, $t6, 1
    sw      $t0, 28($sp)
    ble     $t0, 100, loop
    la      $a0, str
    lw      $a1, 24($sp)
    jal     printf
    move    $v0, $0
    lw      $ra, 20($sp)
    addu    $sp, $sp, 32
    jr      $ra

.data
.align 0
str:
.asciiz "The sum from 0 .. 100 is %d\n"
```

Labels

Representações de código: linguagem de alto nível

```
#include <stdio.h>

int
main (int argc, char *argv[])
{
    int i;
    int sum = 0;
    for (i = 0; i <= 100; i = i + 1) sum = sum + i * i;
    printf ("The sum from 0 .. 100 is %d\n", sum);
}
```

Rotina de biblioteca

Variável

Constante

Operação

Calcula e imprime a soma dos quadrados dos inteiros de 0 a 100

A noção de montagem

- Linguagem de máquina (“**machine** language”)
 - Representação binária das instruções de um processador
- Linguagem de montagem (“**assembly** language”)
 - Representação simbólica da linguagem de máquina

A noção de montagem

- **Linguagem de máquina (“**machine** language”)**
 - Representação binária das instruções de um processador
 - Instruções divididas em campos codificados em binário
 - » Exemplo: 000000 10001 10010 01000 00000 100000
- **Linguagem de montagem (“**assembly** language”)**
 - Representação simbólica da linguagem de máquina

A noção de montagem

- **Linguagem de máquina (“**machine** language”)**
 - Representação binária das instruções de um processador
 - Instruções divididas em campos codificados em binário
 - » Exemplo: 000000 10001 10010 01000 00000 100000
- **Linguagem de montagem (“**assembly** language”)**
 - Representação simbólica da linguagem de máquina
 - Mnemônicos associados a campos da instrução binária
 - » Exemplo: add \$t0, \$s1, \$s2

Elementos do “assembly”

- **Mnemônicos de operações**
 - Exemplos: add, sub, jal
- **Mnemônicos de registradores**
 - Exemplo: \$s1
- **Mnemônicos de endereços de memória**
 - Rótulos (“**Labels**”)
 - Exemplo:
main: add \$t0, \$s1, \$s2

Elementos do “assembly”

- **Pseudo-instruções**
 - São representadas como se fossem instruções
 - Não são nativas do processador
 - Representação alternativa ou condensada
- **Exemplo:**
 - `move $t0, $t1` `# $t0 ← $t1`
 - `add $t0, $zero, $t1` `# $t0 ← $t1 + 0`

O programa montador

- **Tradução operações e registradores**

- Mnemônico → número(s) binário(s)

`add $t0, $s1, $s2` → 000000 10001 10010 01000 00000 100000

- **Tradução de rótulos (“labels”)**

- Endereço simbólico → endereço binário

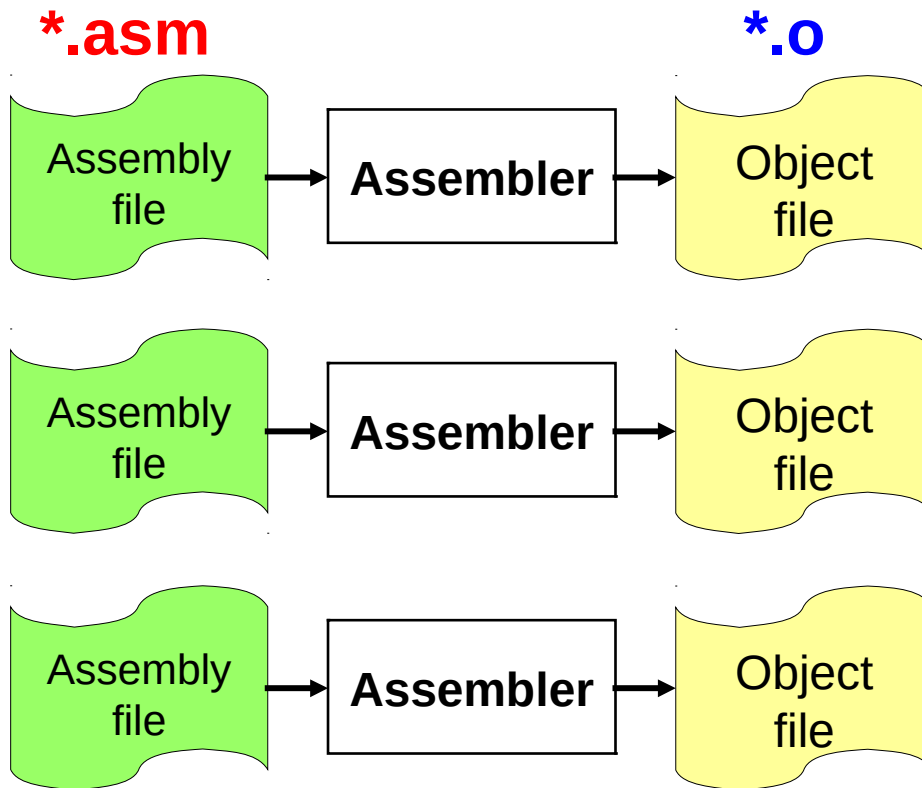
`jal main` → 000011 1010101010101010101010 101010

- **Tradução de pseudo-instruções**

- Pseudo-instrução → instrução nativa

`move $t0, $t1` → `add $t0, $zero, $t1`

O programa montador



Gera um arquivo **objeto para cada módulo**

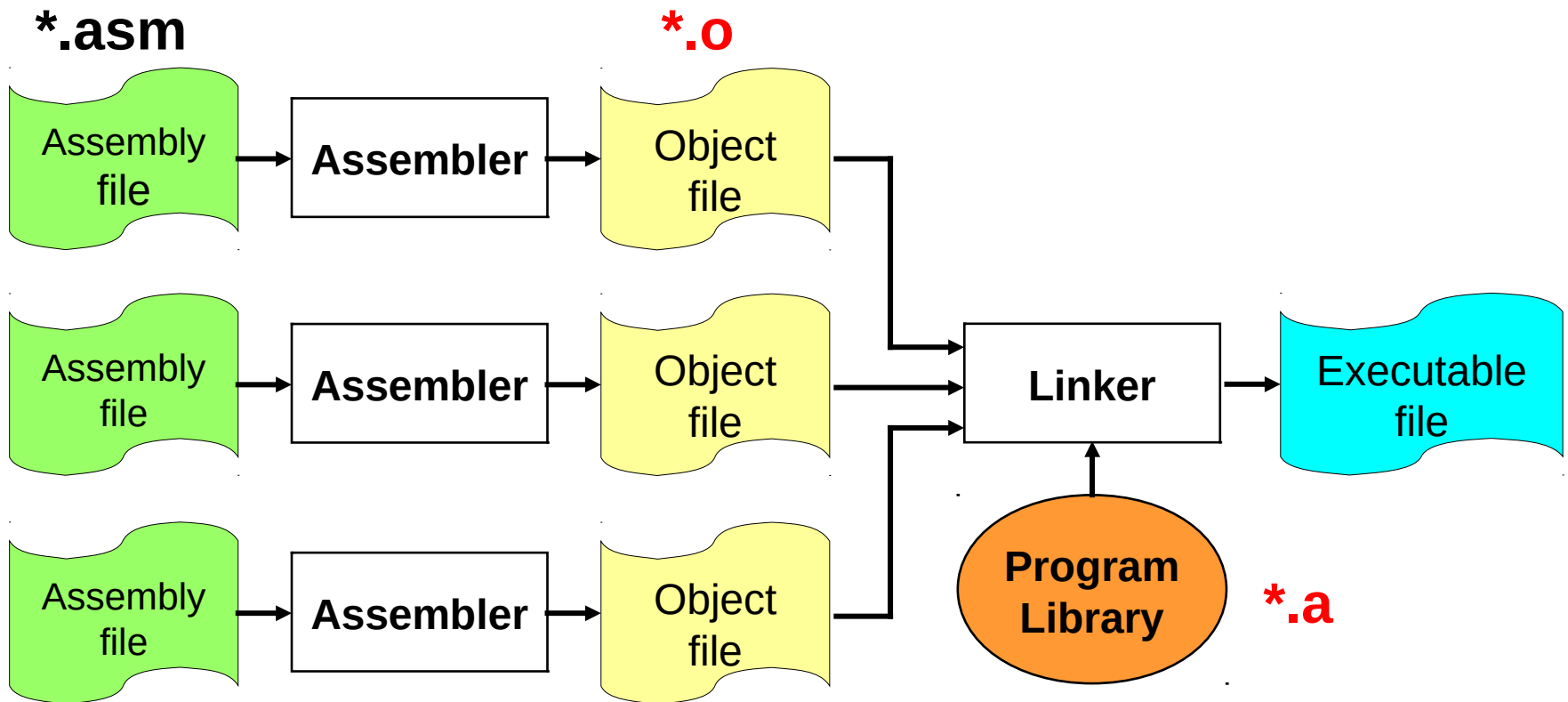
A noção de ligação

- **Programas são divididos em módulos**
 - Desenvolvidos independentemente
 - Compilados e montados independentemente
- **Programas usam bibliotecas**
 - Rotinas pré-desenvolvidas
 - Exemplo: “standard C library”

A noção de ligação

- **Módulo contém referências**
 - Labels de subrotinas e dados
 - Definidos em outro módulo ou biblioteca
- **Referências não resolvidas**
 - Código do módulo não pode ser executado
- **É preciso editar as referências**
 - Para combinar arquivos-objeto e biblioteca
 - Gerando um único **arquivo executável**

O programa ligador



Gera o arquivo **executável**

Uso do “assembly”

- **Tamanho/Velocidade do programa é crítico**
 - Computador embarcado (“embedded computer”)
 - “High end”:
 - » Exemplos: controle de freios (ABS) e estabilidade (ESC)
 - » Objetivo: Viabilização de restrição temporal
 - Resposta rápida e previsível
 - Garantia de resposta em tempo determinado
 - “Low-end”
 - » Exemplo: controle de eletrodomésticos (temporizador programável)
 - » Objetivo: redução de custo de produto
 - Menos instruções, menor quantidade de memória

Uso do “assembly”

- **Trechos críticos de um programa**
 - Detecção de trechos críticos
 - » Maior parte do tempo em pequenos trechos de código
 - » Ferramentas de “profiling”
 - Recodificação em alto nível
 - » Melhores algoritmos e estruturas de dados
 - Recodificação em assembly
 - » Permite maior desempenho
 - Enfoque híbrido:
 - » Maior parte em linguagem de alto nível: C, C++
 - » Trecho crítico em “assembly”: **asm("move \$t0, \$t1");**

Uso do “assembly”

- **Não disponibilidade de um compilador**
 - “Low-end”:
 - » Microcontroladores de baixíssimo custo
 - Exemplo: alguns usados em controle de telefones sem fio
 - “High-end”:
 - » Processadores dedicados à aplicação específica
 - » ASIPs : “Application-specific instruction-set processor”
 - Exemplo: áudio digital

Vantagens do “assembly”

- **Programador controla melhor os recursos do hardware**
 - Pode gastar mais tempo e criatividade otimizando
 - Compiladores produzem código mais uniforme
 - » Mas não enxergam todas as otimizações possíveis

Vantagens do “assembly”

- **Exploração de instruções especializadas**
 - Exemplos:
 - » Cópia de strings
 - » Instruções multimedia
 - » Laços em instrução única
 - » **Instruções recentemente criadas para estender uma ISA, mas ainda não exploradas em compiladores distribuídos em domínio público (ex. [instruções AVX](#))**

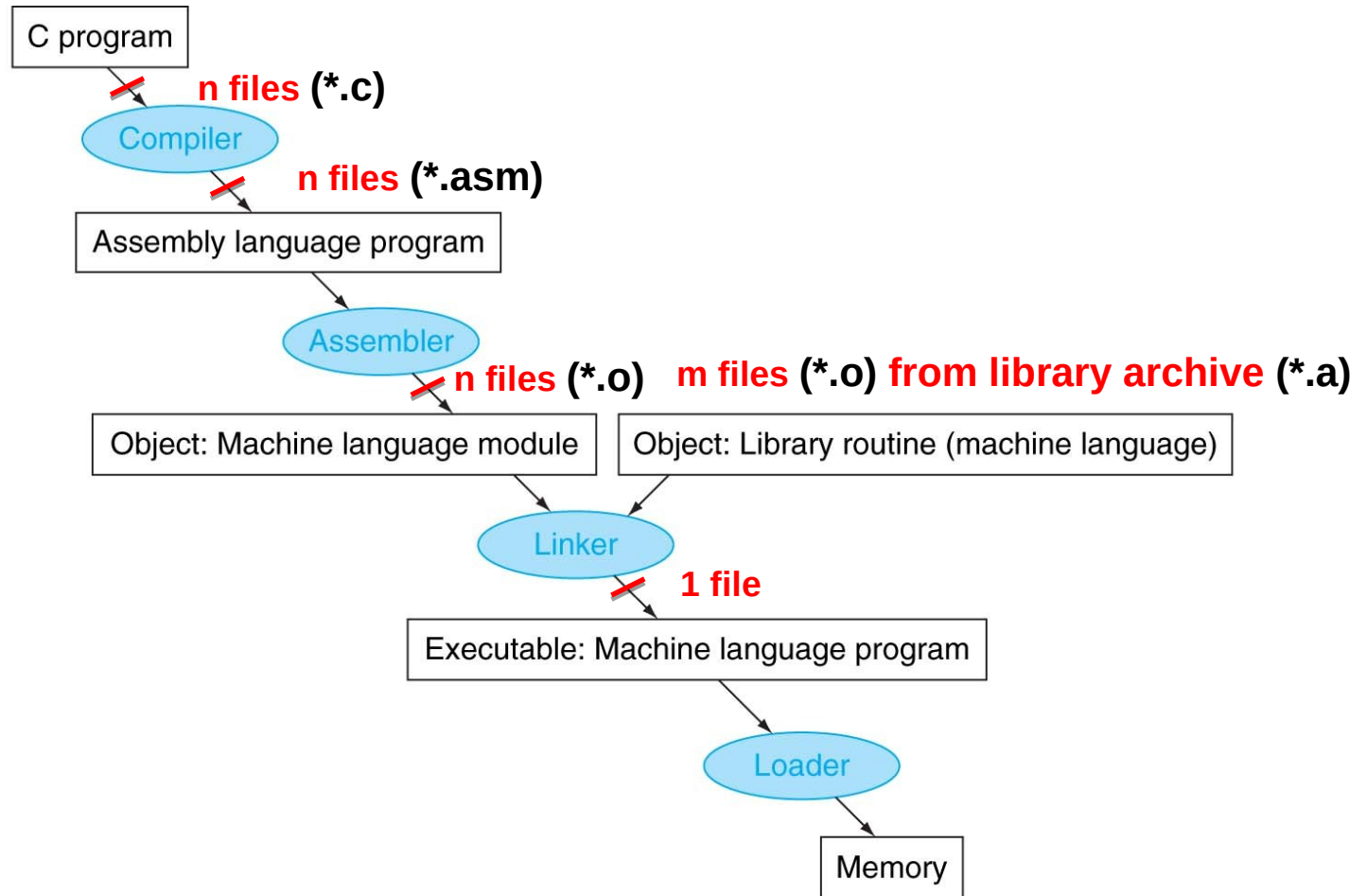
Desvantagens do “assembly”

- **Não portabilidade**
 - Código reescrito para cada novo processador
 - » Só pode ser usado na mesma família (ISA)
- **Programas mais longos**
 - Exemplo desta aula:
 - » C (15 linhas) x assembly (31 linhas)
 - Programas mais complexos: maior expansão

Desvantagens do “assembly”

- **Menor produtividade**
 - Programadores escrevem por dia o mesmo número de linhas de código
 - » Em alto nível ou em assembly
 - Se programa expande n vezes
 - » Produtividade em alto nível é n vezes maior
- **Menor legibilidade**
 - Mais difícil de ler e depurar
 - Falta de estrutura (ex. “if-then-else”)
 - Análise requer “reconstrução” do algoritmo

Cadeia de ferramentas













Introdução à programação de sistemas

Top Languages


















- **IEEE Spectrum 2019 ranking**
 - <https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019>
 - 11 metrics from 8 sources:
 - » CareerBuilder, Google, GitHub, Hacker News, Reddit, Stack Overflow, Twitter
- **Sectors**
 - Web
 - Enterprise
 - Mobile
 - Embedded

Web

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	JavaScript		79.4
4	C#	   	74.5
5	Go	 	68.0
6	HTML,CSS		66.8


















Fonte: IEEE Spectrum, The Top Programming Languages 2019

Enterprise

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	C	  	94.4
4	C++	  	87.5
5	R		81.5
6	C#	   	74.5
















Fonte: IEEE Spectrum, The Top Programming Languages 2019

Mobile

Rank	Language	Type	Score
1	Java	  	96.3
2	C	  	94.4
3	C++	  	87.5
4	C#	   	74.5
5	Swift	 	69.1
6	Dart	 	57.4















Fonte: IEEE Spectrum, The Top Programming Languages 2019

Embedded

Rank	Language	Type	Score
1	Python	  	100.0
2	C	  	94.4
3	C++	  	87.5
4	C#	   	74.5
5	Arduino		67.2
6	Assembly		63.7

Fonte: IEEE Spectrum, The Top Programming Languages 2019

All

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	C	  	94.4
4	C++	  	87.5
5	R		81.5
6	JavaScript		79.4

Fonte: IEEE Spectrum, The Top Programming Languages 2019

Exemplo de programação híbrida

```
long my_mult(long a, long b, long* c) {  
    long overflow;  
    asm("mult $4,$5");  
    asm("mflo $4");  
    asm("sw $4, 0($6)"); // store product  
    asm("mfhi $11");  
    asm("sra $12,$4, 31");  
    asm("sne $12,$12,$11");  
    asm("sw $12, 0(%0)", &overflow); // store overflow  
    return(overflow); }
```