INE5607 – Organização e Arquitetura de Computadores

Linguagem de Montagem e de Máquina

Aula 13: Tradução e inicialização de um programa

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

Sumário

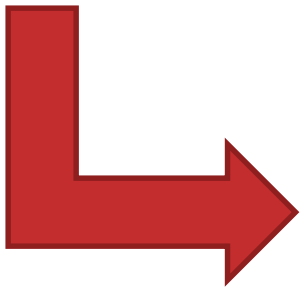
- Do alto nível à máquina
- Compilador
- Montador
- Ligador
- Carregador
- Considerações finais

DO ALTO NÍVEL À MÁQUINA

Do alto nível à máquina

- Exemplo: programa que soma os inteiros de 0 a 100 elevados ao quadrado

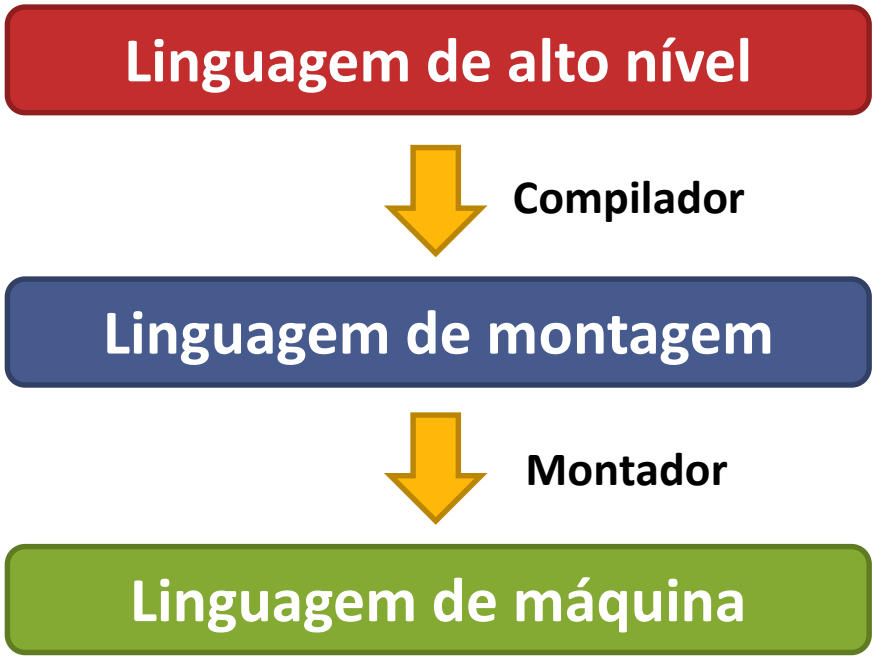
```
int
main (int argc, char *argv[])
{
    int i;
    int sum = 0;

    for (i = 0; i <= 100; i = i + 1) sum = sum + i * i;
    printf ("The sum from 0 .. 100 is %d\n", sum);
}
```



```
00100111101111011111111111100000
10101111101111110000000000010100
10101111101001000000000000100000
10101111101001010000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101110000000000000011000
00000001110011100000000000011001
00100101110010000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
00000000000000000111100000010010
00000011000011111100100000100001
00010100001000001111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
00000011111000000000000000001000
00000000000000000001000000100001
```

Do alto nível à máquina

- Traduzindo o programa:
  (forma simplificada)



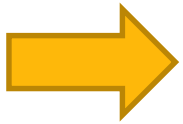
Do alto nível à máquina

- **Compilador**
  - Transforma um programa escrito em linguagem de alto nível (C por exemplo) para linguagem de montagem
  - Compiladores produzem códigos de montagem quase tão bons quanto programadores (ou até melhores)
    - Uso de otimizações

Do alto nível à máquina

- **Montador**
  - Traduz linguagem de montagem em linguagem de máquina

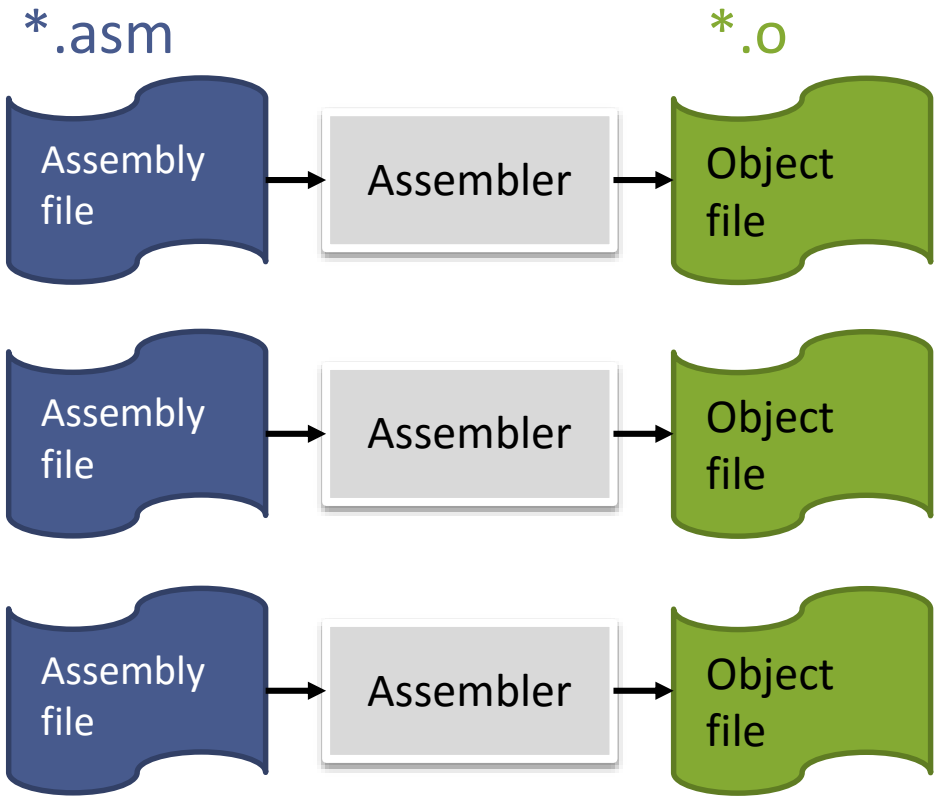
```
addiu   $29, $29, -32
sw      $31, 20($29)
sw      $4,  32($29)
sw      $5,  36($29)
sw      $0,  24($29)
sw      $0,  28($29)
lw      $14, 28($29)
lw      $24, 24($29)
multu   $14, $14
addiu   $8,  $14, 1
slti    $1,  $8, 101
sw      $8,  28($29)
mflo    $15
addu    $25, $24, $15
bne     $1,  $0, -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
```



```
00100111101111011111111111100000
10101111101111110000000000010100
10101111101001000000000000100000
10101111101001010000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101110000000000000011000
00000001110011100000000000011001
00100101110010000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
00000000000000000111100000010010
00000011000011111100100000100001
00010100001000001111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
00000011111000000000000000001000
00000000000000000001000000100001
```

Do alto nível à máquina

- **Montador**
  - Gera um arquivo objeto para cada módulo



Do alto nível à máquina

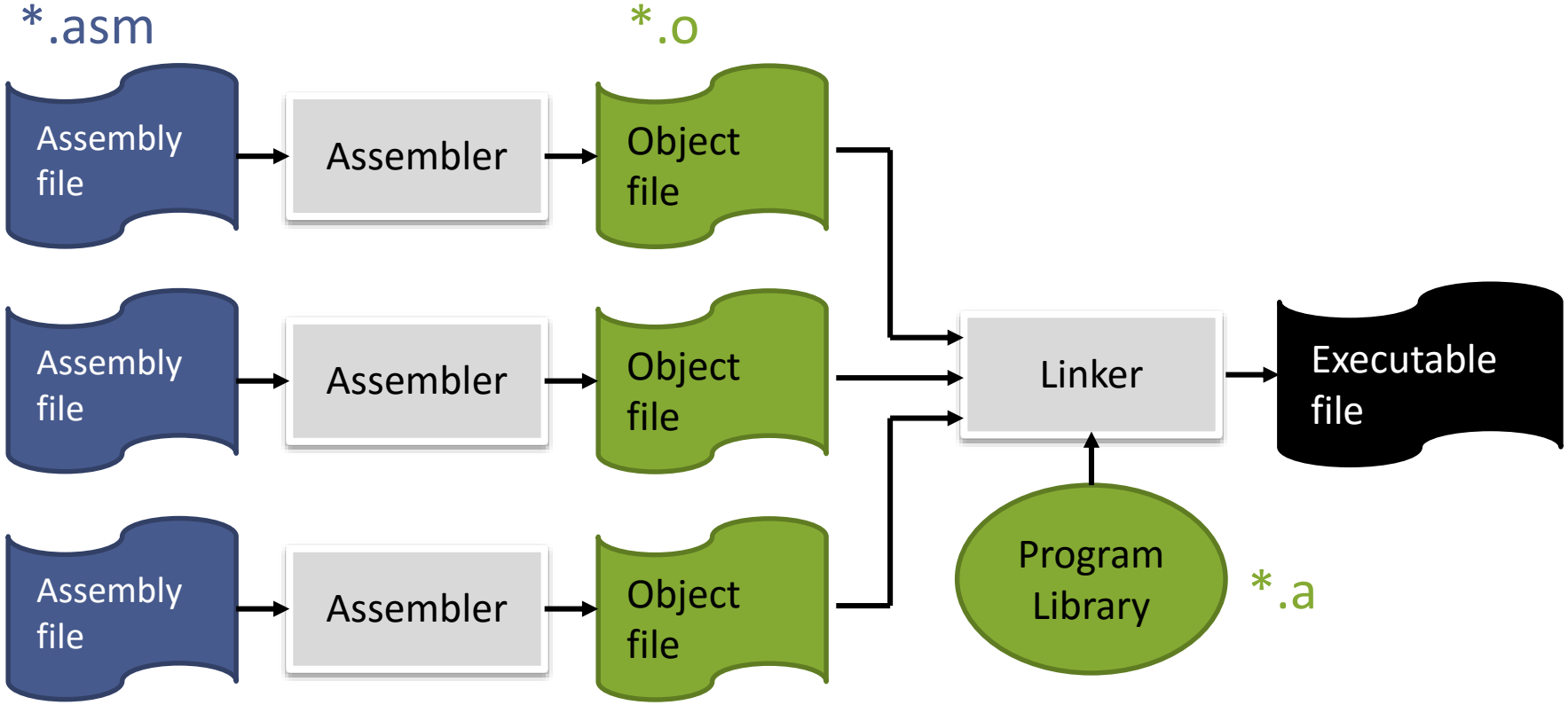
- **Programas**
  - São divididos em **módulos?**
    - Desenvolvidos de forma independente
    - Compilados e montados de forma independente
  - Usam **bibliotecas**
    - Rotinas pré-desenvolvidas
    - Exemplo: “standard C library”

Do alto nível à máquina

- **Módulos**
  - Contém **referências**
    - *Labels* de subrotinas e dados
    - Definidas em outras bibliotecas
  - Referências não resolvidas impedem execução do módulo?
  - É preciso resolver referências, combinar módulos e bibliotecas e gerar um arquivo executável

Do alto nível à máquina

- **Ligador (*linker* ou link-editor)**
  - Gera arquivo executável a partir de arquivos objetos dos módulos e bibliotecas



Do alto nível à máquina

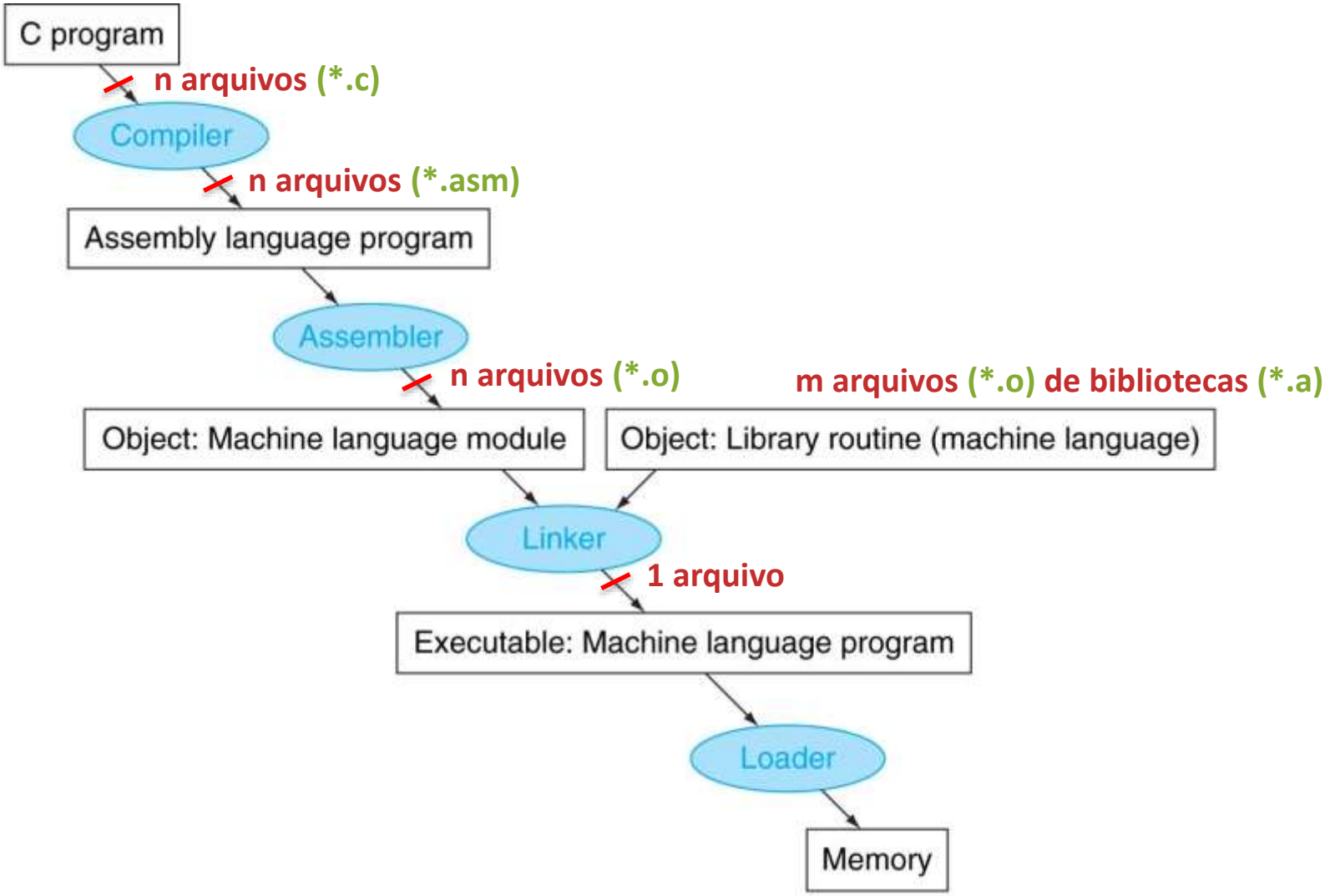
- Processo de **carregamento**
  - Após a ligação, o programa está apto para rodar
  - Antes de rodar, o programa fica armazenado em um dispositivo de memória secundária
  - Para ser executado, o programa deve ser **carregado** na memória e sua execução deve ser iniciada

Do alto nível à máquina

- Processo de **carregamento**
  - Em geral, para iniciar o programa, o sistema operacional realiza diversas tarefas como:
    - Leitura do cabeçalho do arquivo para determinar o tamanho da memória necessária
    - Criação de um espaço de endereçamento para o programa
    - Copia instruções e dados para o espaço de endereçamento
    - Copia os argumentos passados ao programa para a sua **pilha**
    - Inicializa os registradores
    - Pula para o início do programa (**main**)

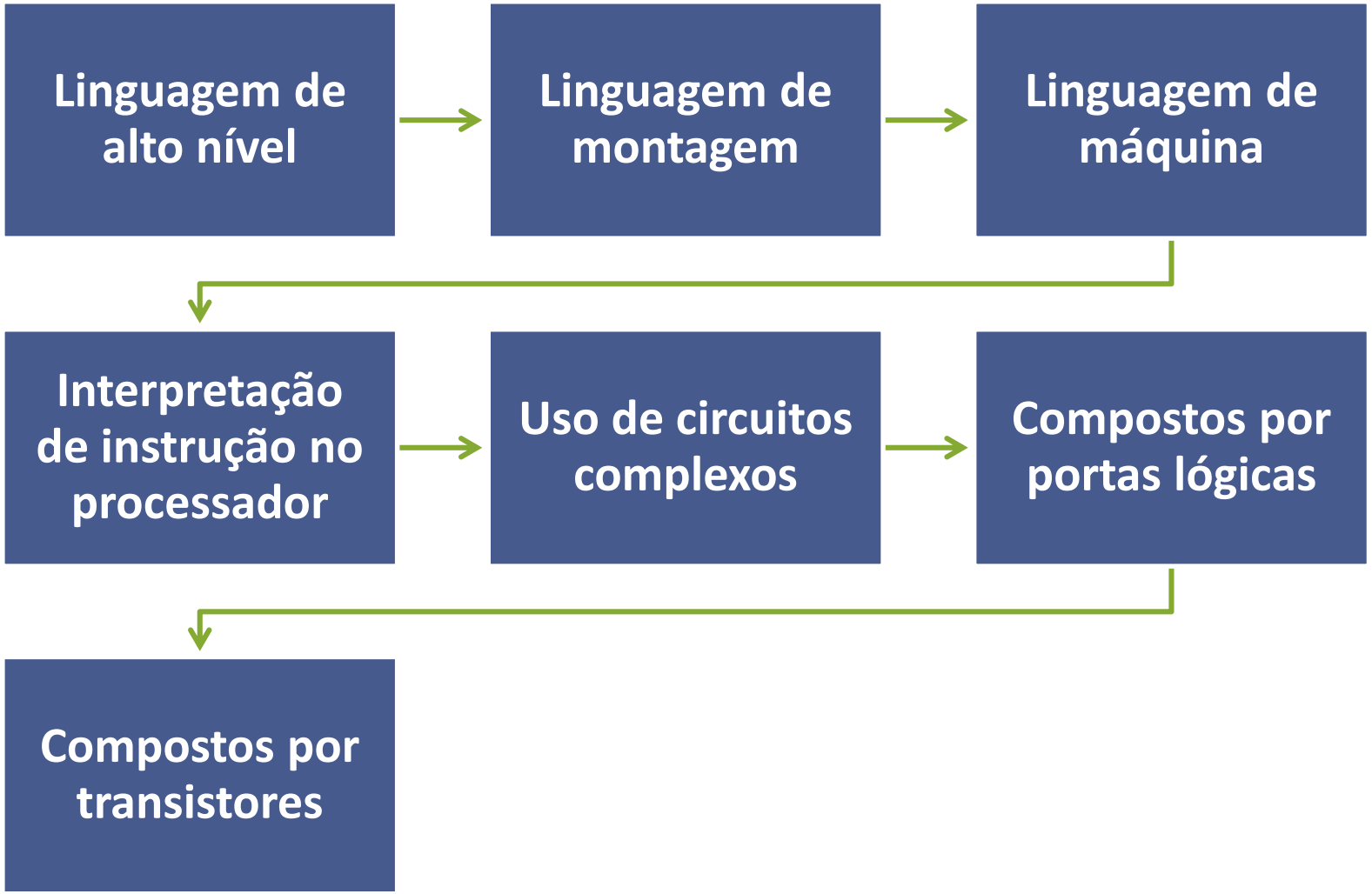
Do alto nível à máquina

- **Com isso temos:**



Do alto nível à máquina

- **Com isso temos:**



COMPILADOR

Compilador

- O que faz
  - Transforma um programa escrito em linguagem de alto nível (C por exemplo) para linguagem de montagem
  - Compiladores produzem códigos de montagem quase tão bons quanto programadores (ou até melhores)
    - Uso de otimizações
  - Basicamente o que nós passamos fazendo nas últimas aulas

Compilador

- Otimizações: exemplo
  - x[i] = x[i] + 42
    - Considerando x em $s0 e i em $t2

```
mult $t3, $t2, 4          #i*4
add $t4, $t3, $s0         #posição x[i] em mem
lw $t5, 0($t4)            #lê x[i]
addi $t6, $t5, 42         #x[i]+42
mult $t7, $t2, 4          #i*4
add $t8, $t7, $s0         #posição x[i] em mem
sw $t6, 0($t8)            #armazena x[i]+42
```

Compilador

- **Redução de força**
  - Operações complexas por operações mais simples

~~mult $t3, $t2, 4~~ -> sll $t3, $t2, 2

add $t4, $t3, $s0

lw $t5, 0($t4)

addi $t6, $t5, 42

~~mult $t7, $t2, 4~~ -> sll $t3, $t2, 2

add $t8, $t7, $s0

sw $t6, 0($t8)

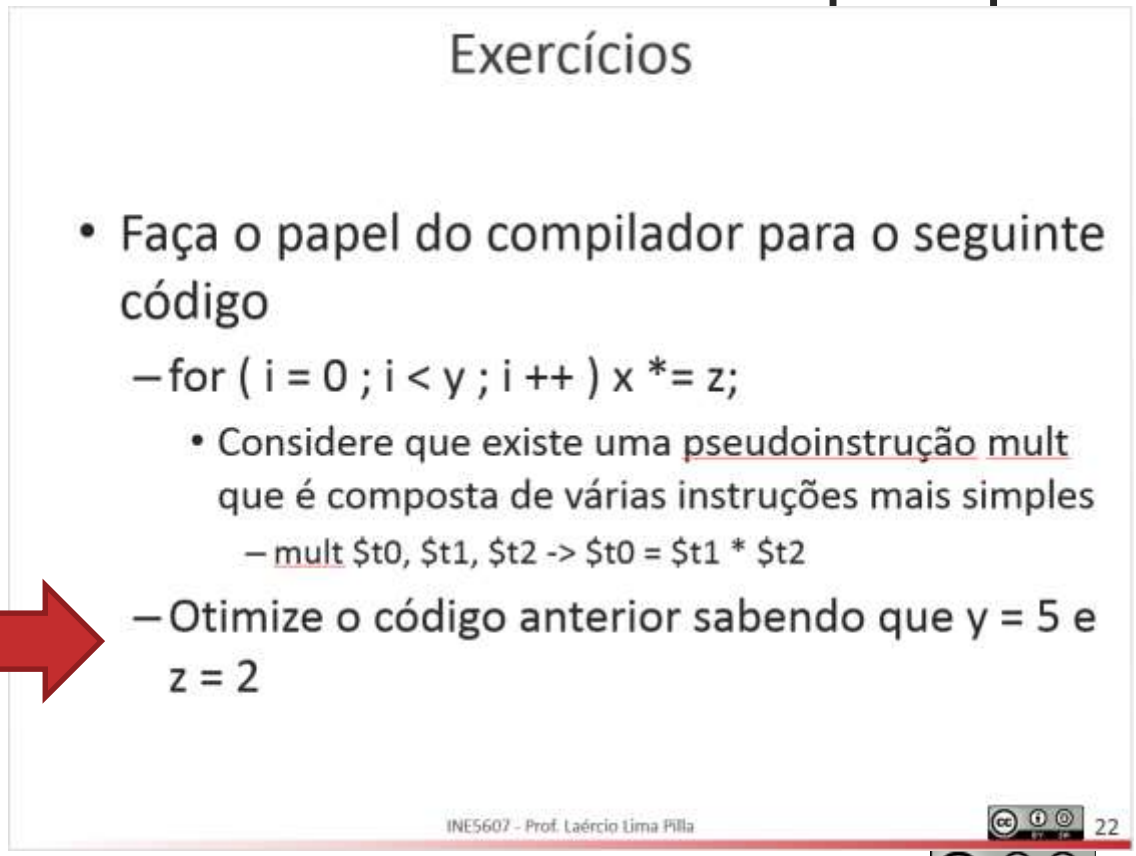
Compilador

- **Propagação de cópia e eliminação de subexpressões comuns**
  - Elimina replicação

```
sll $t3, $t2, 2
add $t4, $t3, $s0
lw $t5, 0($t4)
addi $t6, $t5, 42
~~sll $t3, $t2, 2~~
~~add $t8, $t3, $s0~~
sw $t6, 0($t4)
```

Compilador

- **Propagação de constante e junção de constante**
  - Encontram constantes e as usam sempre que possível



Compilador

- Eliminação de local de armazenamento morto
  - Remoção de variáveis não usadas
    - *Warning: blabla is defined but never used*
- Eliminação de código morto
  - Remoção de operações cujas saídas não são utilizadas
    - Aquele bla = a + 1 esquecido no código

MONTADOR

Montador

- O que faz
  - Traduz linguagem simbólica em linguagem de máquina
    - Resolve pseudoinstruções
    - Organiza arquivo objeto

Montador

- Resolução de pseudoinstruções
  - move $t0, $t1
    - **add $t0, $t1, $zero**
  - bge $t0, $t1, Label
    - *Branch greater than or equal*
    - **slt $t9, $t0, $t1**
    - **beq $t9, $zero, label**

Montador

- Organização arquivo objeto
  - Exemplo UNIX: seis partes
  1. Cabeçalho do arquivo
  2. Segmento de texto
  3. Segmento de dados estáticos
  4. Informações de relocação
  5. Tabela de símbolos
  6. Informações de depuração

Montador

- Organização arquivo objeto
  - Exemplo UNIX: seis partes
  1. Cabeçalho do arquivo
     - Tamanho e posição das partes do arquivo objeto
  2. Segmento de texto
     - Instruções em linguagem de máquina
  3. Segmento de dados estáticos
     - Dados alocados por toda a vida do programa
     - Variáveis globais

Montador

- Organização arquivo objeto
  - Exemplo UNIX: seis partes
  4. Informações de relocação
     - Identifica instruções e words que dependem de endereços absolutos
  5. Tabela de símbolos
     - Rótulos que ainda não estão definidos
       - Referências externas
  6. Informações de depuração
     - Descrição de como o módulo foi compilado

Montador

Cabeçalho do arquivo objeto

	Nome	Procedimento A	
	Tamanho do texto	100hexa	
	Tamanho dos dados	20hexa	
Segmento de texto	Endereço	Instrução	
	0	lw $a0, 0($gp)	
	4	jal 0	
	...	...	
Segmento de dados	0	(X)	
	...	...	
Informações de relocação	Endereço	Tipo de instrução	Dependência
	0	lw	X
	4	jal	B
Tabela de símbolos	Rótulo	Endereço	
	X	-	
	B	-	

Montador

Cabeçalho do arquivo objeto

	Nome	Procedimento B	
	Tamanho do texto	200hexa	
	Tamanho dos dados	30hexa	
Segmento de texto	Endereço	Instrução	
	0	sw $a1, 0($gp)	
	4	jal 0	
	...	...	
Segmento de dados	0	(Y)	
	...	...	
Informações de relocação	Endereço	Tipo de instrução	Dependência
	0	sw	Y
	4	jal	A
Tabela de símbolos	Rótulo	Endereço	
	Y	-	
	A	-	

LIGADOR

Ligador

- O que faz
  - Gera arquivo executável a partir de arquivos objetos dos módulos e bibliotecas
    - Organiza os módulos de código e dados simbolicamente na memória
    - Determina os endereços dos rótulos de dados e instruções
    - Remenda as referências internas e externas

Ligador

Cabeçalho do arquivo objeto A

	Nome	Proc. A	
	Tam. Texto	100hexa	
	Tam. Dados	20hexa	
Seg. de texto	End.	Inst.	
	0	lw $a0, 0($gp)	
	4	jal 0	
	...	...	
Seg. de dados	0	(X)	
	...	...	
Info. de relocação	End.	Tipo	Dep.
	0	lw	X
	4	jal	B
Tabela de símbolos	Rótulo	Endereço	
	X	-	
	B	-	

Cabeçalho do arquivo objeto B

	Nome	Proc. B	
	Tam. Texto	200hexa	
	Tam. Dados	30hexa	
Seg. de texto	End.	Inst.	
	0	sw $a1, 0($gp)	
	4	jal 0	
	...	...	
Seg. de dados	0	(Y)	
	...	...	
Info. de relocação	End.	Tipo	Dep.
	0	sw	Y
	4	jal	A
Tabela de símbolos	Rótulo	Endereço	
	Y	-	
	A	-	

Ligador

- Organização dos módulos
  - Ordem: Primeiro procedimento A e depois o procedimento B, por exemplo

Seg. de texto	End.	Inst.
	0	lw $a0, 0($gp)
	4	jal 0
	...	...

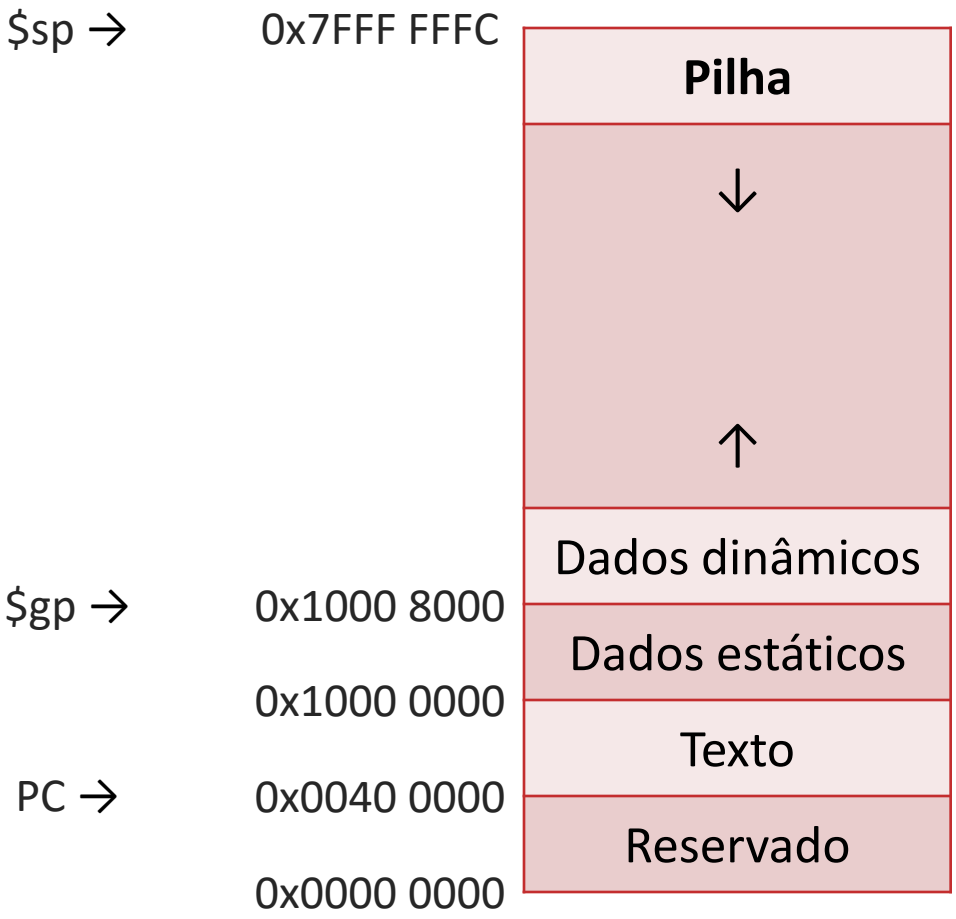
End.	Inst.
0->100	sw $a1, 0($gp)
4->104	jal 0
...	...

Cabeçalho do arquivo objeto A	
Nome	Proc. A
Tam. Texto	100hexa
Tam. Dados	20hexa

Cabeçalho do arquivo objeto B	
Nome	Proc. A
Tam. Texto	200hexa
Tam. Dados	30hexa

Ligador

- Alocação de espaço padrão MIPS 32

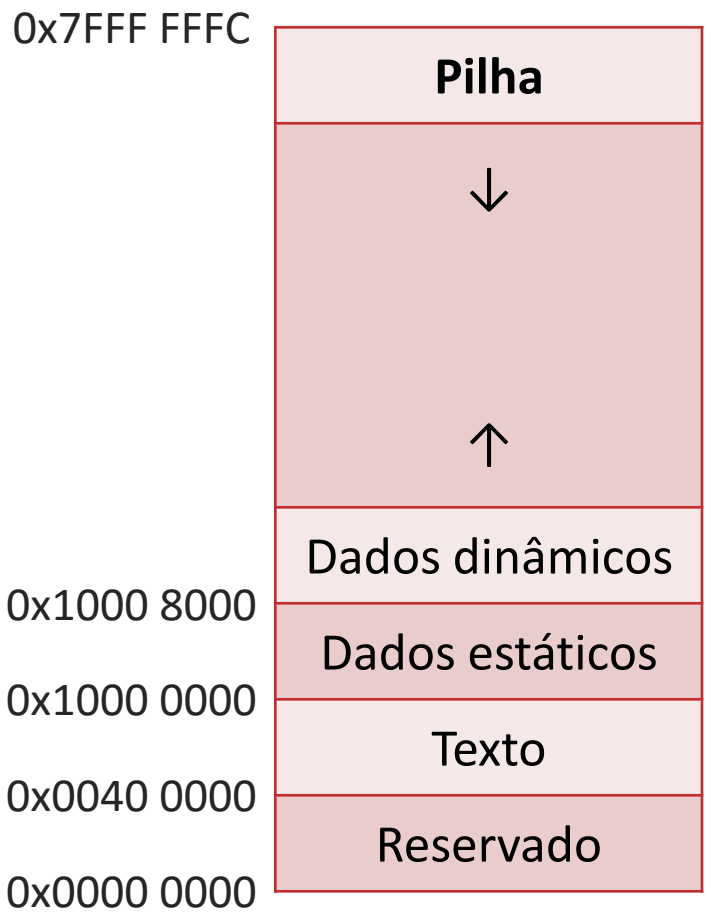


Ligador

- Organização dos módulos
  - Determina os endereços dos rótulos de dados e instruções

Seg. de texto	End.	Inst.
	0040 0000	lw $a0, 0($gp)
	0040 0004	jal 0
	...	...

End.	Inst.
0040 0100	sw $a1, 0($gp)
0040 0104	jal 0
...	...



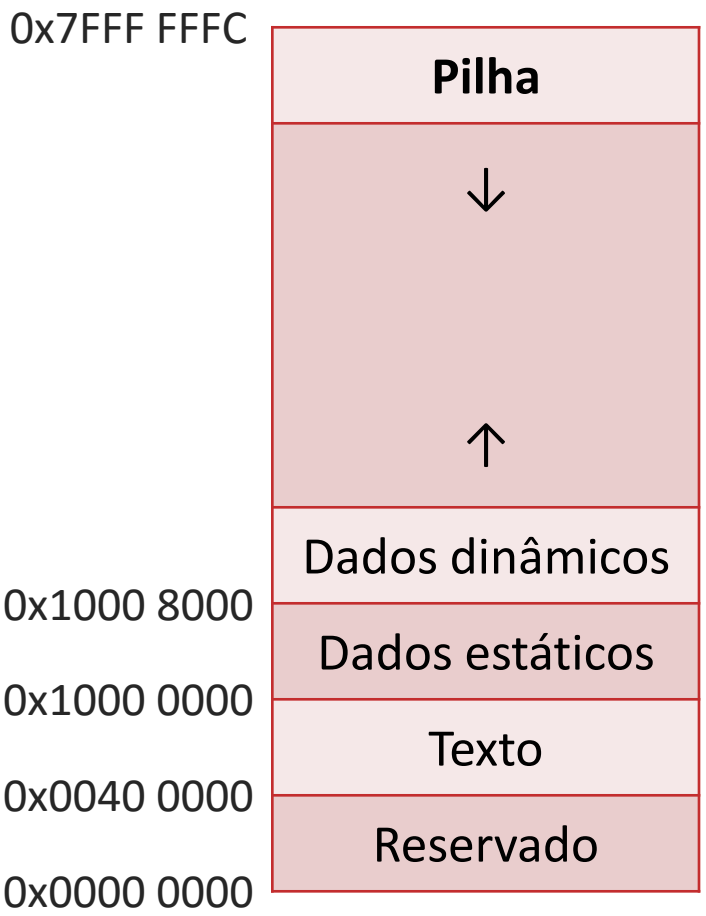
Ligador

- Organização dos módulos
  - Determina os endereços dos rótulos de dados e instruções

Seg. de dados	1000 0000	(X)
	...	...
	1000 0020	(Y)
	...	...

Cabeçalho do arquivo objeto A	
Tam. Dados	20hexa

Cabeçalho do arquivo objeto B	
Tam. Dados	30hexa



Ligador

- Organização dos módulos
  - Remenda as referências internas e externas

Segmento de texto	End.	Inst.
	0040 0000	lw $a0, 0x8000($gp)
	0040 0004	jal 0x40 0100
	...	...

End.	Inst.
0040 0100	sw $a1, 0x8020($gp)
0040 0104	jal 0x40 0000
...	...

Seg. de dados		
	1000 0000	(X)
	...	...
	1000 0020	(Y)
	...	...

Info. de relocação	End.	Tipo	Dep.
	0	lw	X
	4	jal	B

Tabela de símbolos	Rótulo	Endereço
	X	-
	B	-

End.	Tipo	Dep.
0	sw	Y
4	jal	A

Rótulo	Endereço
Y	-
A	-

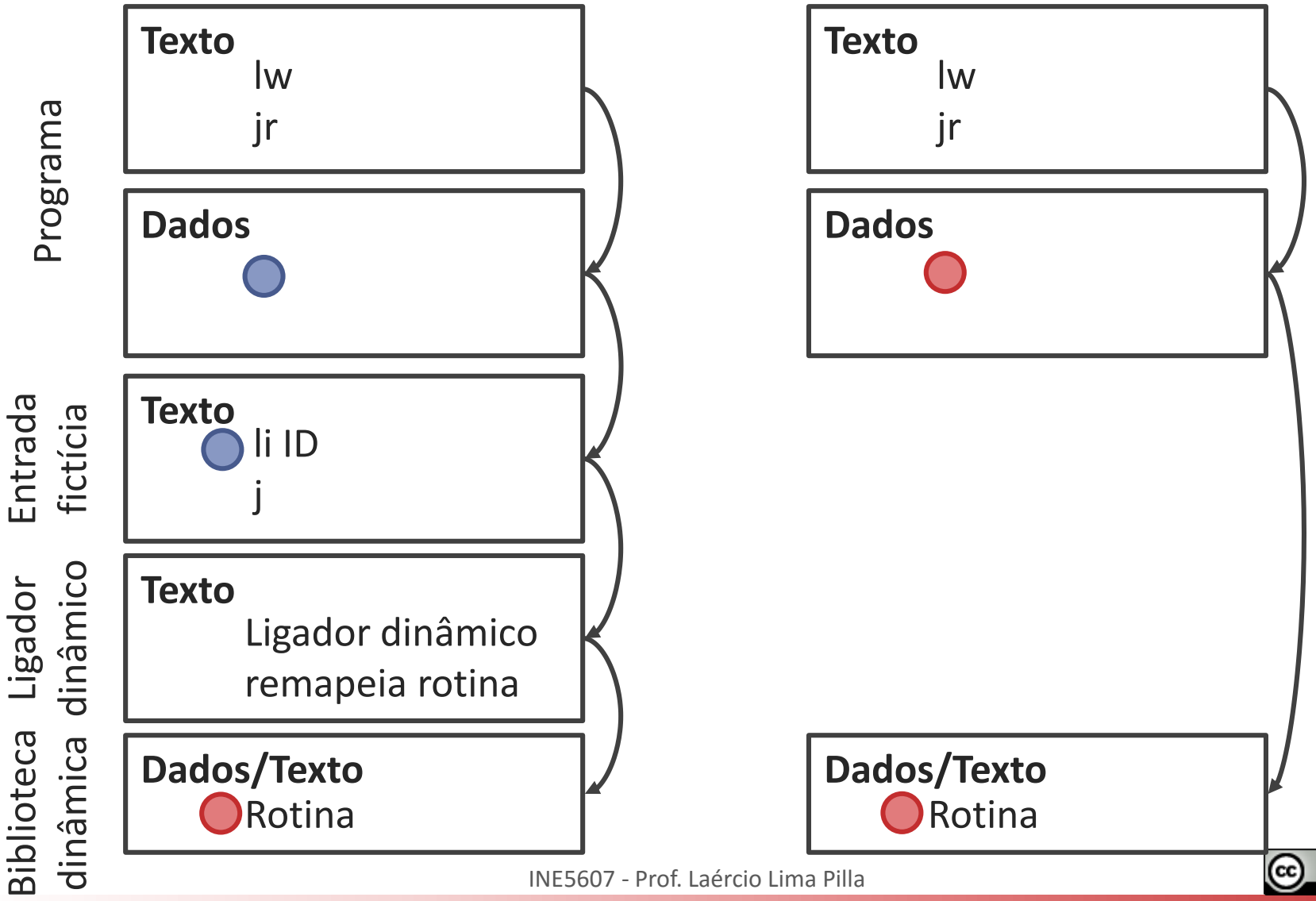
CARREGADOR

Carregador

- O que faz
  - Leitura do cabeçalho do arquivo para determinar o tamanho da memória necessária
  - Criação de um espaço de endereçamento para o programa
  - Copia instruções e dados para o espaço de endereçamento
  - Copia os argumentos passados ao programa para a sua **pilha**
  - Inicializa os registradores
  - Pula para o início do programa (**main**)

Carregador

- Tratamento de bibliotecas dinâmicas



CONSIDERAÇÕES FINAIS

Considerações finais

- Software de sistema
  - Compilador
  - Montador
  - Ligador
  - Carregador

Considerações finais

- Próxima aula
  - Laboratório 2

INE5607 – Organização e Arquitetura de Computadores

Linguagem de Montagem e de Máquina

Aula 13: Tradução e inicialização de um programa

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br