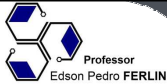
Professor
Edson Pedro FERLIN

Linguagem de Montagem (Assembly)

Prof. Edson Pedro Ferlin

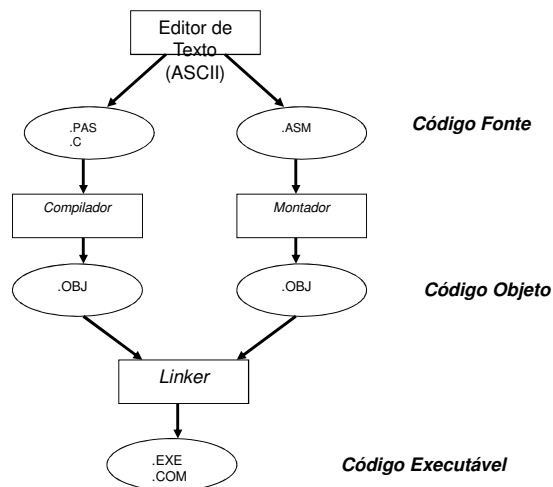
Professor
Edson Pedro FERLIN

- **Objetivos**
 - Apresentar a Linguagem de Montagem
- **Conteúdos**
 - Introdução à Linguagem de Montagem
 - Processo de Montagem

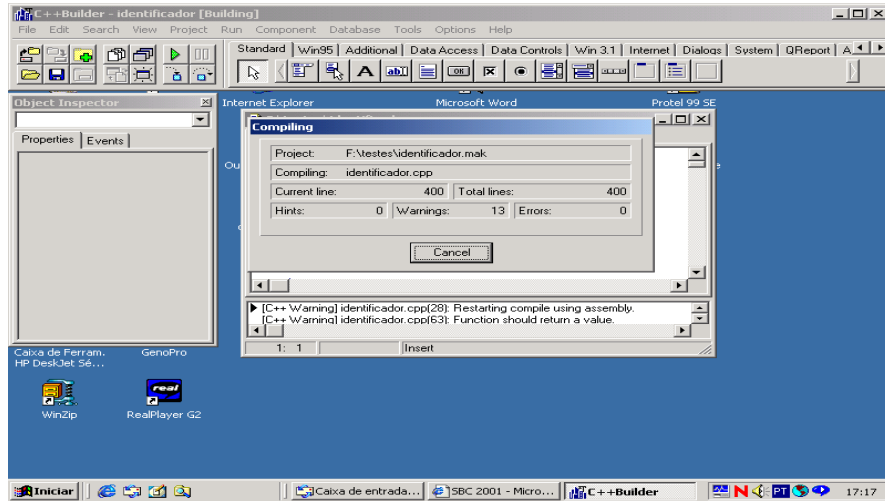
Linguagens de Programação

- Alto Nível – *Pascal, Lisp, Prolog, ...*
- Nível Médio – *C*
- Baixo Nível – *Assembler (Assembly)*

Processo de Compilação



Etapa de Compilação

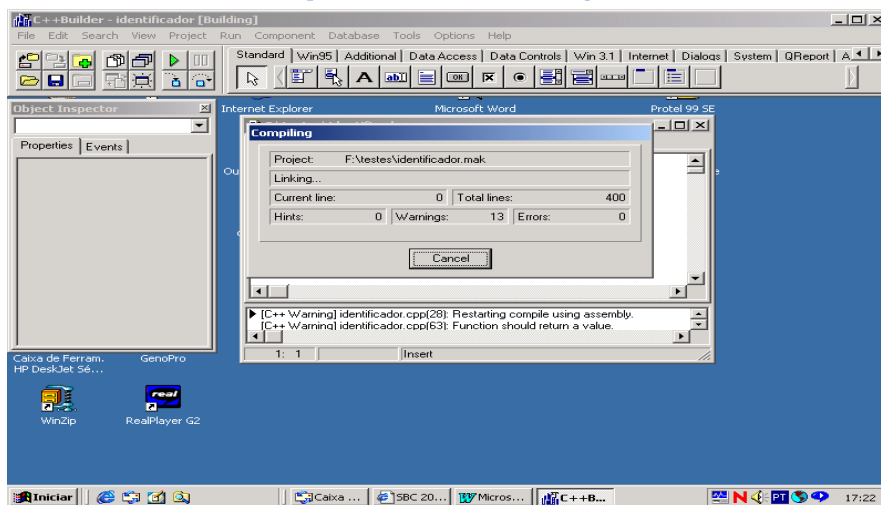


5

Linguagem de Montagem

Prof. Edson Pedro Ferlin

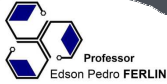
Etapa de Linkedição



6

Linguagem de Montagem

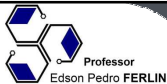
Prof. Edson Pedro Ferlin



Linguagen de Montagem

É uma representação simbólica para uma linguagem numérica de máquina.

- Cada comando (mnemônico) produz exatamente uma instrução de máquina.
- É fácil de programar (linguagem de máquina).
- O programador tem acesso a todos os recursos e instruções disponíveis na máquina.



Formato da Linguagen de Montagem

```

mov      a,#1d      ; a recebe 1 decimal
mov      b,#2d      ; b recebe 2 decimal
Loop:   add      a,b      ; a := a + b
        inc      b      ; incrementa b
        sjmp     Loop    ; Loop infinito

```

The diagram below illustrates the structure of the assembly code shown above. It uses brackets and arrows to identify the components of each instruction line:

- Label:** The text 'Loop:' at the start of the third line.
- Mnemonic:** The operation code, such as 'mov', 'add', 'inc', and 'sjmp'.
- Operand(s):** The data or registers involved, such as 'a,#1d', 'b,#2d', 'a,b', and 'Loop'.
- Comentários:** The semicolon-separated text explaining the instruction, such as '; a recebe 1 decimal'.

Processo de Montagem

Montadores de Dois Passos:

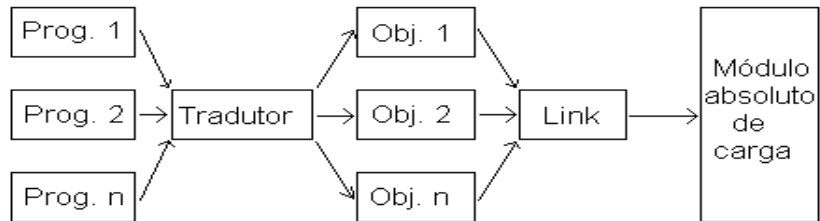
- Passo 1 – *Criação das tabelas (símbolos e de instruções)*
- Passo 2 – *Gerar o código objeto e as informações para o linker*

Erros – *Símbolo indefinido, duplicação de símbolos, operação não válida*

Módulo Objeto

Identificação
Tabela de <i>Entry Points</i> (<i>símbolos internos</i>)
Tabela de <i>Referência Externas</i> (<i>Símbolos externos</i>)
<i>Instruções de Máquina</i> e <i>Constantes</i>
<i>Dicionário de Relocação</i>
<i>Fim do Módulo</i>

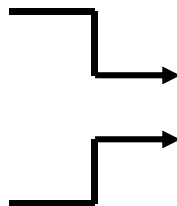
Ligação e Carga



Exemplo de Compilação

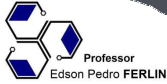
PRINTF ("a");

MOV DX, #41h
MOV AH, #2h
INT 21h



1011-1010-0100-0001-0000-0000
1011-0100-0000-0010
1100-1101-0010-0001

→ BA 41 00
B4 02
CD 21



Exemplo de Código em Assembly

$(x=x+3)+4$

```
; TINY
;
```

```
LDA    x
LOD    x
LDC    3
ADI
STN
LDC    4
ADI
```

```
; Intel 80x86
; Borland C 3.0
```

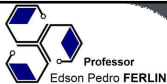
```
MOV    AX,word ptr [bp-2]
ADD    AX,3
MOV    word ptr[bp-2],AX
ADD    AX,4
```

```
; SparcStations
; Sun C 2.0
```

```
LD     [%fp+-0x4],%o1
ADD    %o1,0x3,%o1
ST     %o1,[%fp+-0x4]
LD     [%fp+-0x4],%o2
ADD    %o2,0x4,%o3
```

```
; 80x51/80x31
;
```

```
MOV    A,x
ADD    A,#3h
MOV    x,A
ADD    A,#4h
```




Afinação de Programas

```
timeStart = timeGetTime();           //GET TICK EDGE
for(;;)
{
    timeStop = timeGetTime();
    if ( (timeStop-timeStart) > 1 )   // ROLLOVER PAST 1
    {
        __asm{
            xor    eax, eax
            xor    ebx, ebx
            xor    ecx, ecx
            xor    edx, edx
            _emit 0x0f           // CPUID
            _emit 0xa2          // RDTSC
            _emit 0x0f           // RDTSC
            _emit 0x31          // RDTSC
            mov    [StartTicks], eax // TICK COUNTER STARTS HERE
        }
        break;
    }
}
```

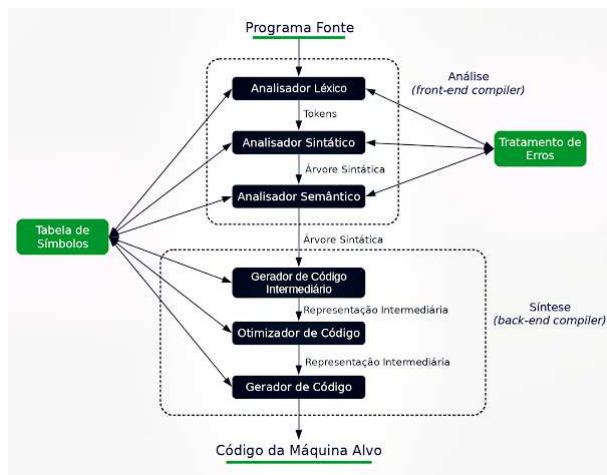
Compiladores

Elemento-chave para a obtenção do desempenho

Fonte  Executável



Módulos do Compilador



Otimização - Eliminação de Dependências

- Renomeação
- Substituição a frente
- Expansão Escalar
- Distribuição do Laço

A = B + C
D = A + E
A = A + D



A1 = B + C
D = A1 + E
A = A1 + D

A = B + C
D = A + E
A = A + D



D = B + C + E
A = B + C + B + C + E

DO I=1, N
S: X = C(I)
T: D(I) = X + 1
END_DO



DO I=1, N
S: X(I) = C(I)
T: D(I) = X(I) + 1
END_DO

Estrutura – Arquivo EXE

0	4D 5A 00 00 50 4E 00 00 4C 01 00 00 00 00 00 00	M Z P E L
10	00 00 00 00 00 00 00 00 00 00 03 01 0B 01 00 00	L 8
20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	i
30	00 00 00 00 00 00 00 00 00 00 40 00 04 00 00 00	8 J
40	04 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00	J J
50	00 00 00 00 00 00 20 00 00 00 00 00 00 00 00 00	
60	03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	L #
70	00 00 00 00 00 00 00 00 0E 00 00 00 00 00 00 00	
80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
90	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
E0	00 00 00 00 00 00 00 00 00 00 00 00 6A 2C 58 C3	J, X Å
F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

- DOS Header
- PE Header overlaps DOS
- Optional Header overlaps DOS
- Optional Header
- Data Directories
- Program Code

```
0x6A // PUSH
0x2C // value to push
0x58 // POP EAX
0xC3 // RETN
```

268 bytes is the absolute minimum size for a working executable under Windows 7 64-bit edition



Escolha do Compilador

Sugestões para escolher um Compilador

Fornecidas pela Intel

(Capítulo 6 – *Suggestions for Choosing a Compiler*)

- *Important Features for a Compiler*
- *Compiler Switches Recommendation*

Contato



eferlin@live.com



(BLOG) professorferlin.blogspot.com

(SITE) professorferlin.webnode.com.br

(YOUTUBE) [ProfEdsonPedroFerlin](https://www.youtube.com/ProfEdsonPedroFerlin)