

Linguagem de Montagem (Assembly)

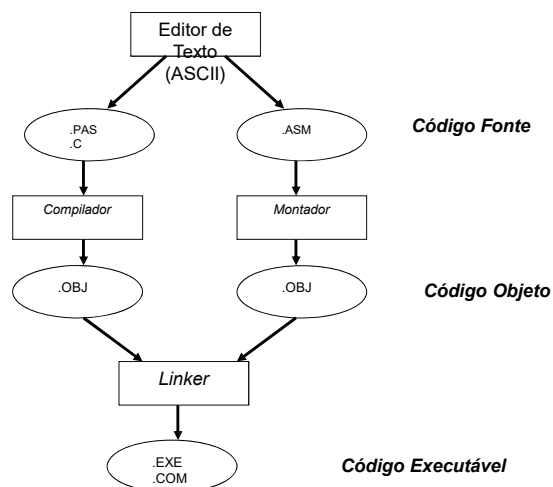
Prof. Edson Pedro Ferlin

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

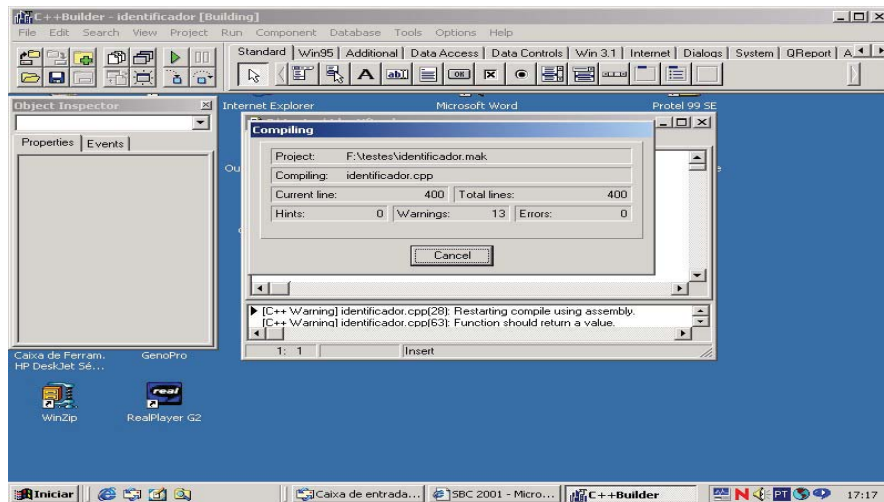
Linguagens de Programação

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

Processo de Compilação



Compilação

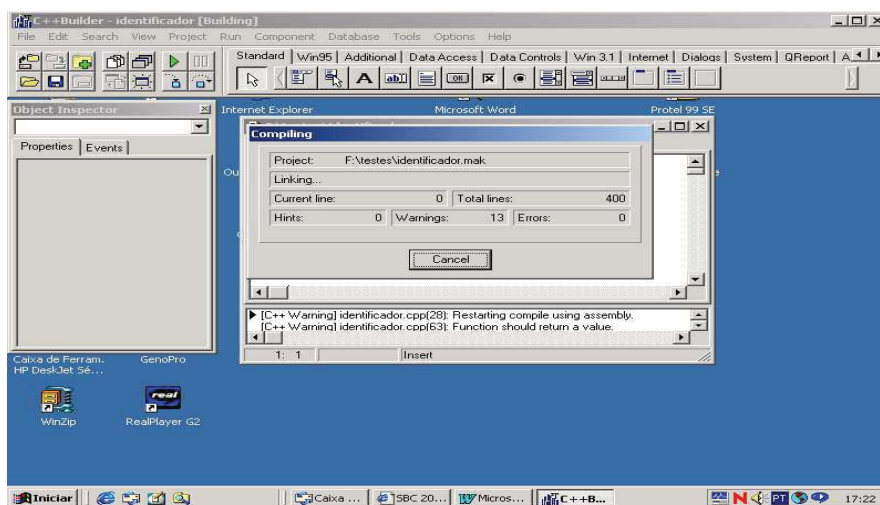


5

Linguagem de Montagem

Prof. Edson Pedro Ferlin

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

```

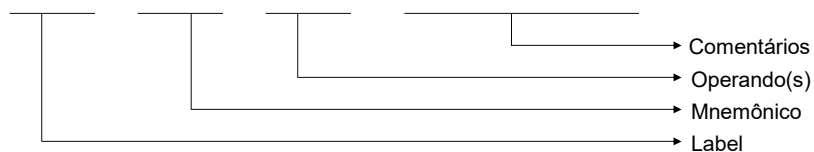


Diagram illustrating the format of an assembly instruction:

- Label (e.g., **Loop:**)
- Mnemônico (e.g., **add**)
- Operando(s) (e.g., **a,b**)
- Comentários (e.g., **; a := a + b**)

Processo de Montagem

Montadores de Dois Passos:

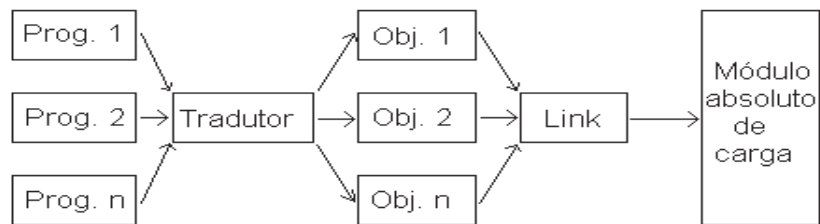
- Passo 1 – *Criação das tabelas (símbolos e de instruções)*
- Passo 2 – *Gerar o código objeto e as informações para o linker*

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> <i>e Constantes</i>
<i>Dicionário de Relocação</i>
<i>Fim do Módulo</i>

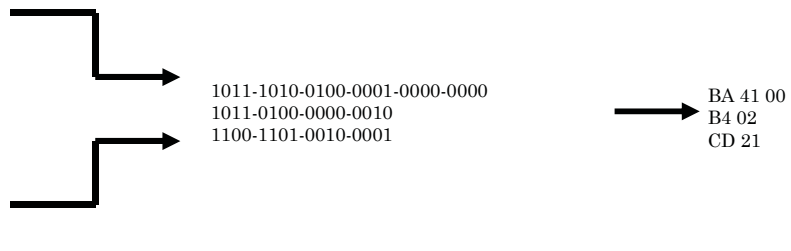
Ligação e Carga



Exemplo de Compilação

```
PRINTF("a");
```

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





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
```



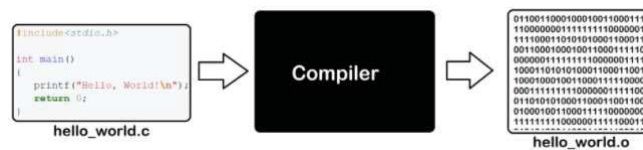
Afinação de Programas

```
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
            db     0x0f,0xa2           // CPUID
            db     0x0f,0xc3           // RDTSC
            mov     [StartTicks], eax  // TICK COUNTER STARTS HERE
        }
        break;
    }
}
```

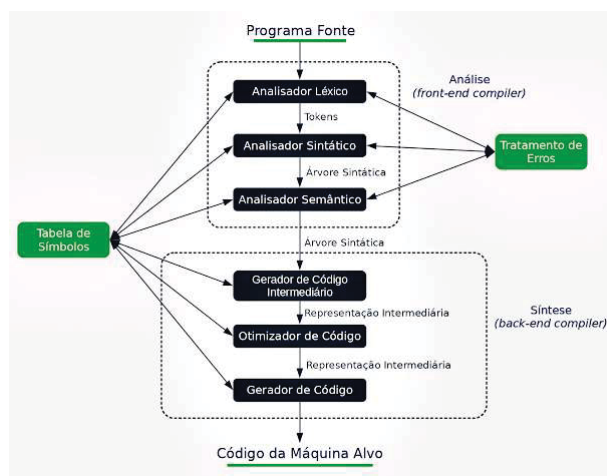
Compiladores

Elemento-chave para a obtenção do desempenho

Fonte  Executável



Compiladores



Otimização - Eliminação de Dependências

- Renomeação

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



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

- Substituição a frente

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



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

- Expansão Escalar

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

- Distribuição do Laço

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*

Atividade

- Resolver os exercícios.

