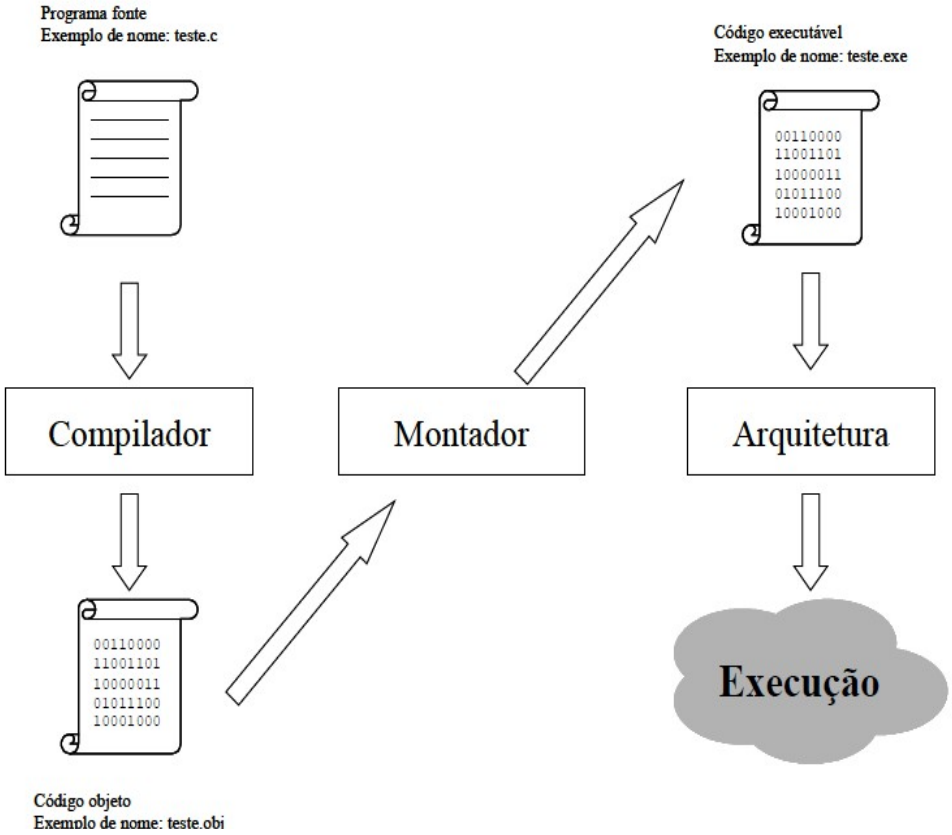
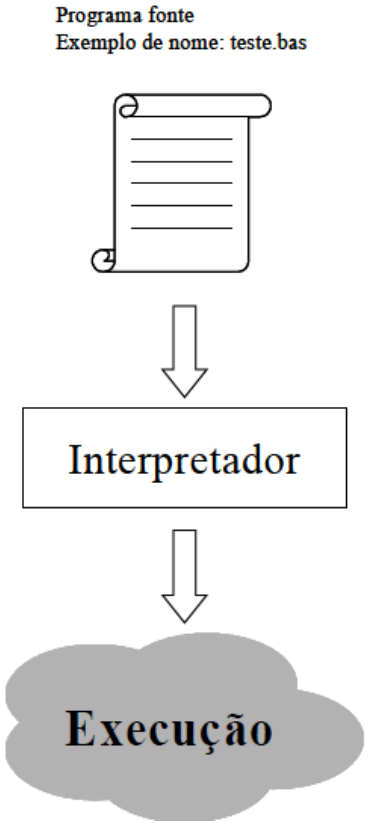
Fundamentos de Programação

Prof. Márcio Miguel Gomes

Processo de Compilação



Processo de Interpretação



Compilar

- Conforme dicionário Aurélio:
- Reunir metodicamente escritos diversos sobre o mesmo assunto
- Reunir numa mesma obra trabalhos de várias origens
- Juntar um conjunto de informações
- Converter linguagem de programação em linguagem ou código que possa ser lido ou executado por um computador

Compilador

- Um compilador é um programa que interpreta um programa em código fonte para gerar um programa em outra linguagem (*assembly*)
- O código fonte deve ser compatível com o compilador utilizado
- O programa gerado em *assembly* deve ser compatível com o *hardware* e sistema operacional que irá executar o programa

Etapas da Compilação

- Análise léxica
- Análise sintática
- Análise semântica
- Geração de código intermediário
- Otimização de código
- Geração de código de objeto

Código Fonte C++

```
int main()
{
    int meses;
    float salario;
    cout << "Informe seu salário mensal: ";
    cin >> salario;
    cout << "Quantidade de meses trabalhados: ";
    cin >> meses;
    cout << "Total recebido: " << salario * meses << endl;
    return 0;
}
```

Código Fonte Python

```
salario = float(input('Informe seu salário mensal: '))
meses = int(input('Quantidade de meses trabalhados: '))
print('Total recebido:', salario * meses)
```

Código Assembly

```
section   .text
   global_start

_start:
   mov    edx,len
   mov    ecx,msg
   mov    ebx,1
   mov    eax,4
   int    0x80

   mov    eax,1
   int    0x80

section   .data
     msg db 'Hello, world!',0xa
     len equ $ - msg
```

Montador

- Um montador é um processo que converte o código *assembly* em um objeto de máquina
- O objeto de máquina é o programa ou biblioteca
- É o código executado pela máquina

Código de Máquina

Address	Machine Language				Assembly Language
0000 0000	0000	0000	0000	0000	TOTAL .BLOCK 1
0000 0001	0000	0000	0000	0010	ABC .WORD 2
0000 0010	0000	0000	0000	0011	XYZ .WORD 3
0000 0011	0001	1101	0000	0001	LOAD REGD, ABC
0000 0100	0001	1110	0000	0010	LOAD REGE, XYZ
0000 0101	0101	1111	1101	1110	ADD REGF, REGD, REGE
0000 0110	0010	1111	0000	0000	STORE REGF, TOTAL
0000 0111	1111	0000	0000	0000	HALT