Interpretador x Compilador

Mário Leite

A primeira coisa a ser esclarecida a respeito desses dois termos: Interpretador e Compilador: eles NÃO SÃO HARDWARE: são *softwares*. Você não pode tocar em nenhum deles; são apenas códigos internos da linguagem usados para fazer a tradução para a linguagem de máquina, pois a máquina (sistema computacional) não entende comandos em linguagem; por exemplo, o seu computador não entende a instrução (em Fortran 77): **WRITE(6,*) 'Ola, Mundo!'** ==> imprime a frase tradicional "Ola, Mundo!", no dispositivo **6** (impressora). Mas, o computador não entende nada disso; ele necessita de um mecanismo que faça a tradução para a sua própria linguagem: a linguagem de máquina (dos **0**'s e **1**'s); e esse tradutor pode ser: Interpretador, Compilador; ou ainda, uma cominação desses dois (tradução híbrida). Assim, baseando no Assemblyx86 (linguagem de baixo nível para a primeira etapa da tradução), o resultado final da tradução que é entregue à máquina, seria:

01001111 01001100 01000001 00101100 00100000

01001101 01010101 01001110 01000100 01001111 00100001

Onde o primeiro *byte* **01001111 -** conjunto de oito bits) representa a letra maiúscula "**O**" da frase desejada **Ola, Mundo**!

O esquema abaixo ilustra como é feita a tradução de um programa escrito em linguagem de alto nível para a Linguagem de Máquina - LM.

A diferença entre Interpretador e Compilador pode ser resumida no seguinte:

O **Interpretador** é um programa que faz a *leitura/tradução/execução* (nesta ordem) de cada linha do programa-fonte, permitindo ao programador saber de imediato se a instrução é válida ou não; e em alguns casos, permite corrigir a instrução e prosseguir com o processo.

O Compilador é um programa que gera um código em Linguagem de Máquina a partir da tradução integral do programa-fonte (programa original escrito na Linguagem de Alto Nível - LN), dando como resultado final um outro código (programa executável) que pode ser executado diretamente pelo sistema operacional. Após o processo de compilação, normalmente um outro processo se faz necessário: é a *linkedição* (ligação). Nessa etapa intermediária (que em algumas ferramentas é transparente ao usuário) são agregadas algumas funções que estão em bibliotecas, produzindo finalmente o arquivo-executável do programa cuja extensão normalmente é .EXE.

O esquema da f**igura 1** mostra o esquema de tradução de LN para LM com *compilador*, *interpretador* e uma *tradução híbrida*. A primeira parte mostra que, antes de ser obtido o código-executável, o processo de compilação cria um arquivo com um código denominado **código-objeto**, que ainda contém alguns símbolos legíveis que posteriormente convertidos totalmente em LM. Observe, ainda nesta figura (terceira parte) uma nova maneira de traduzir sob compilação está sendo muito empregada hoje em dia para facilitar a portabilidade dos programas; o resultado é um código binário (intermediário), denominado *bytecode*, que posteriormente é traduzido para a LM pura. A vantagem é que o *bytecode* pode ser executado por qualquer processador; é o caso da linguagem Java. A t**abela 1** mostra as vantagens e desvantagens no uso de cada um desses tipos de tradutores. A **figura 2** mostra o código-fonte, codificado em **C**, do programa "Ola, Mundo!", e a **figura 3** parte (tem muito mais) de seu código correspondente em linguagem de máquina (baixo nível) "imprimível".

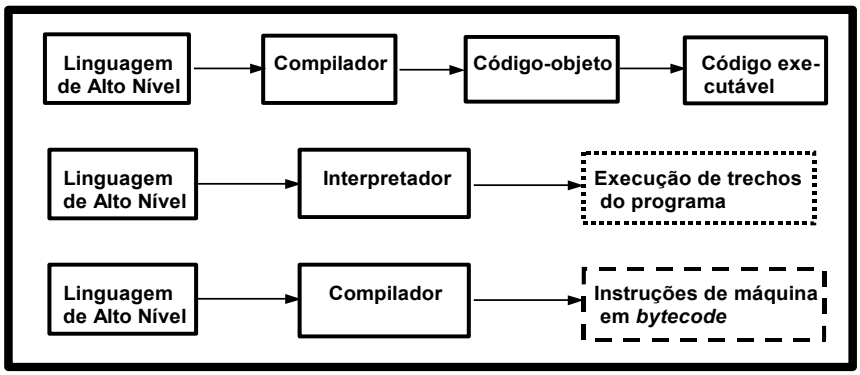

Figura 1 - Tipos de tradução de um programa em Linguagem de Alto Nível.

Tradutor	Vantagens	Desvantagens
Compilador	• Permite estruturas de programação mais complexas, otimizando o código. • Gera arquivo-executável, permitindo maior autonomia e segurança do código-fonte. • Execução mais rápida.	• Correção de erros mais difícil. • Não permite correções dinamicamente. • Necessita de várias etapas de tradução do código-fonte. • Consome muita memória.
Interpretador	• Consome pouca memória. • Permite estruturas dinâmicas de programação. • Tradução em uma única etapa.	• Execução lenta. • Não gera arquivo-executável, o que diminui a segurança do código-fonte.

Tabela 1 - Comparação entre Compilação e Interpretação

OlaMundo.C | Cronometro3.C | Fatorial.C

```c
#include <stdio.h>
#include <conio.h>

int main(void)
{
    printf("Ola, Mundo!\n");
    getch();   //apenas para o usuário poder ver a saída na tela
    return 0;
}
```

Figura 2 -Código-fonte do programa "OlaMundo" na linguagem C.

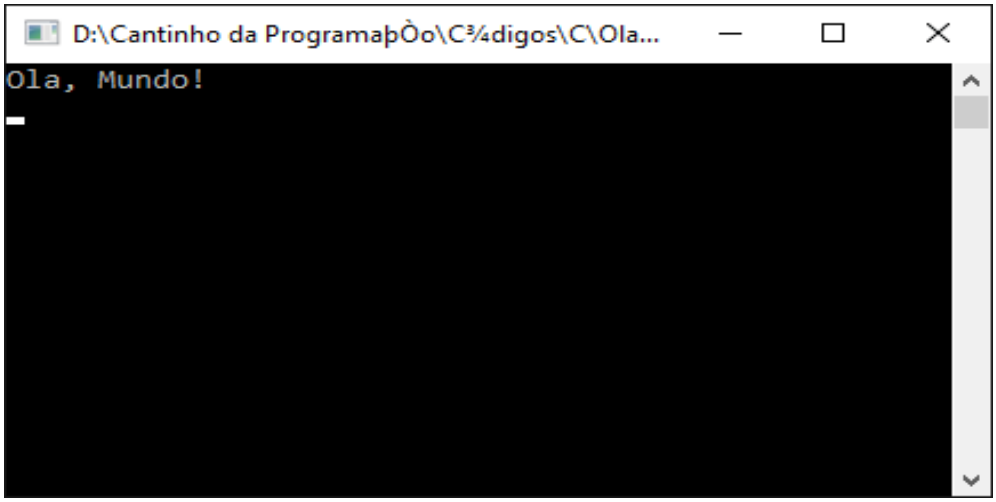

Figura 2 – Saída do programa "OlaMundo"

Figura 3 - Código do programa “OlaMundo” em linguagem de máquina (baixo nível)