Unidade 2: Fundamentos da Programação de Computadores

Prof. Daniel Caetano

Objetivo: Compreender os diferentes tipos de linguagem de programação, como as linguagens de programação de alto nível são traduzidas para o computador e, finalmente, conhecer as diferentes unidades de daods/informações usadas no computador.

Bibliografia:

- MONTEIRO, M.A. **Introdução à Organização de Computadores**. 5ª. Ed. Rio de Janeiro: LTC, 2008.
- MURDOCCA, M. J; HEURING, V.P. **Introdução à Arquitetura de Computadores**. S.I.: Ed. Campus, 2000.
- TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 2ª.Ed. São Paulo: Prentice Hall, 2003.

INTRODUÇÃO

Vimos que todo o processamento de dados realizado por um computador segue uma "receita" chamada "**programa**". Um programa é uma implementação, na linguagem do computador, de um "**algoritmo**".

Poderíamos escrever nossos programas diretamente na linguagem do computador mas essa tarefa costuma ser extremamente tediosa, porque tudo teria que ser escrito em "0s" e "1s", que é a linguagem **binária** que o computador entende.

Sendo assim, costumamos programar em linguagens chamadas de "alto nível", muito parecidas com a nossa própria língua, que posteriormente é convertida para a linguagem binária dos computadores com a ajuda de alguns programas, chamados **compiladores**.

Todos esses tópicos serão abordados com mais detalhes nas próximas seções.

1. PROGRAMAÇÃO E OS ALGORITMOS

O ser humano sempre se deparou com "problemas", isto é, situações indesejadas que exigem intervenção para que uma situação desejável seja atingida. A intervenção a ser realizada depende do problema, assim como do que se considera um resultado desejável.

Frequentemente essa intervenção é uma sequência de tarefas que transformam a situação. À uma sequência de passos que visa atingir a um objetivo definido dá-se o nome de **algoritmo**.

Um exemplo clássico são as receitas de culinária:

Algoritmo 1 - Fazer um Omelete:
Passo 1: Em um prato fundo, bata 3 ovos.
Passo 2: Acrescente sal.
Passo 3: Acrescente cheiro-verde.
Passo 4: Bata mais um pouco.
Passo 5: Leve ao fogo médio em frigideira untada com manteiga.
Passo 6: Depois de dourar um lado, vire e deixe dourar o outro lado.

Algoritmo 2 - Fazer um Misto Quente
Passo 1: Pegar o presunto
Passo 2: Grelhar o presunto
Passo 3: Colocar o queijo sobre o presunto
Passo 4: Pegar duas fatias de pão de forma
Passo 5: Colocar uma fatia de pão sobre o queijo.
Passo 6: Virar e colocar a outra fatia de pão.
Passo 7: Deixe dourar ambos os lados.

Entretanto, um algoritmo precisa ser escrito em um formato, uma linguagem e com instruções que seu executor conheça e seja capaz de executar. Por exemplo: de nada adianta dar uma ordem para que uma geladeira faça café, se ela não for equipada para isso; ou pedir a um advogado que projete uma usina hidrelétrica. Da mesma forma, dar ordens em português para uma pessoa que só compreende russo não trará bons resultados.

Sendo assim, quando escrevemos um algoritmo em passos que um computador seja capaz de executar e em uma linguagem que ele seja capaz de compreender, esse algoritmo recebe o nome de **programa**.

Mas quais tarefas o computador sabe executar? E qual linguagem ele compreende?

A primeira pergunta é fácil de responder: de maneira geral, ele sabe fazer cálculos, decidir se executa ou não uma tarefa com base em algum valor, receber e armazenar dados do usuário e, finalmente, informar resultados ao usuário.

Quanto a linguagem, será necessária uma discussão mais aprofundada.

2. LINGUAGEM DE MÁQUINA E DE MONTAGEM

Assim como os seres humanos possuem sua linguagem - palavras e comandos que compreendem -, cada computador tem também a sua própria linguagem. Nós somos capazes de nos comunicar pelo som e pela escrita, gerando os mais diversos ruídos e símbolos para nos comunicarmos.

Os computadores, entretanto, são dispositivos eletrônicos, que se comunicam por meio de **fios**. Os fios, grosso modo, só permitem dois tipos de interações entre dispositivos eletrônicos: ou existe uma corrente elétrica entre eles... ou não existe essa corrente elétrica. Convencionou-se chamar o estado de passagem de corrente elétrica como “1” e o estado de não-passagem de corrente elétrica como “0”. Em outras palavras, um fio pode representar um valor “0” ou “1”, de acordo com seu estado de corrente. À quantidade de informação que

pode ser representada por um fio desse tipo dá-se o nome de **bit**, a menor unidade de informação em um computador.

Como um único fio acaba propiciando uma comunicação muito limitada, é comum usar vários fios para comunicação, cada um deles representando um bit, compondo grupos de vários bits chamados **números binários**.

A linguagem do computador - ou a linguagem de máquina - é uma linguagem em que esses números binários representam instruções; por exemplo, para armazenar o valor 10 na "variável" A, em um determinado processador chamado Z80, usamos a seguinte sequência de bits:

00111110 00001010

Obviamante escrever programas assim não é nem um pouco prático e não demorou até que alguém criasse um "nome" e uma "grafia" mais natural para os humanos para um comando desse tipo:

LOAD A,10

Muito mais fácil de ler, correto?

Sem dúvida, mas um programa escrito assim precisava antes ser "traduzido": uma enorme tabela continua todos os comandos nessa linguagem "mnemônica" (isso é, mais fácil de guardar) e sua versão em binário. O programa era contruído (ou montado) nessa linguagem mnemônica e depois era traduzido manualmente no momento de digitar no computador. Assim, essa linguagem mnemônica ficou conhecida como "lingaugem de montagem" (linguagem **assembly**).

Não demorou muito para que alguém construísse um programa que fosse capaz de ler a linguagem de montagem e gerasse o código binário para o computador; esses programas passaram a ser conhecidos como "montadores" ou **assemblers**. Esse primeiro passo foi muito importante e praticamente foi o início da atividade que se conhece hoje como "programação".

O assembler, entretanto, tinha um problema: apesar das instruções serem mais facilmente compreendidas pelo ser humano, elas eram simplesmente um "símbolo" diferente para especificar as limitadas instruções do computador. E, para piorar, cada computador/CPU possuia uma linguagem assembly própria, o que significava que um programa escrito para um computador não serviria em outro, de outro fabricante.

E com a "popularização" inicial da programação, os programadores passaram a desejar poder escrever programas de maneira mais parecida com que faziam nas deduções matemáticas: LOAD A,10 deveria ser escrito como **A = 10** e coisas mais complicadas deviam ser possíveis: **A = 10 * 17 / X**. Além disso, o ideal seria que um programa escrito deveria poder ser executado em vários computadores, de modelos diferentes.

Foi essa a motivação para a criação das **linguagens de programação de alto nível**.

3. LINGUAGENS DE PROGRAMAÇÃO DE ALTO NÍVEL

A linguagem assembly é denominada linguagem de baixo nível porque, para programar nela, o programador precisa "descer ao nível do computador", ou seja, construir um programa com ordens muito simples, que a CPU consiga compreender diretamente.

A ideia das linguagens de alto nível é que o programador possa construir programas em uma linguagem mais parecida com a usada por ele em seu dia-a-dia, sem se preocupar com as instruções que o computador compreende ou não.

Sendo assim, foram criadas linguagens como FORTRAN, COBOL, LISP, C/C++, BASIC, PASCAL, Java... E tantas outras. Cada uma delas com um propósito específico de facilitar a criação de um tipo específico de programa.

Essas linguagens são, na verdade, um "meio termo" entre a forma como os humanos se expressam e o que os computadores entendem. Elas exigem um pouco menos de esforço do ser humano quando comparado ao esforço necessário para escrever em assembly, mas ainda assim **são absolutamente incompreensíveis para o computador**.

Para que o computador seja capaz de compreender tais linguagens é necessária uma **tradução**, ou seja, é necessário que "alguém" leia o código na linguagem de alto nível e escreva um novo código em linguagem de máquina, para que o computador saiba executá-lo. Esse "alguém" que faz a tradução é um programa de computador denominado **compilador**.

Assim, o compilador é um programa que lê um código escrito em "linguagem de alto nível" e, com base nele, cria um programa em linguagem de máquina, chamado código objeto.

Em sistemas mais complexos, os programas tendem a ser compostos por várias partes menores que precisam ser conectadas para que possam funcionar. Esse trabalho final, que transforma o código objeto em um **arquivo executável** é chamado de **linker**.

4. UNIDADES DA INFORMAÇÃO

A quantidade de dados gerada e processada por um computador é, em geral, muito maior que aquela que podemos armazenar em 1 bit. Como vimos, mesmo as instruções do computador são organizadas em conjuntos de bits, sendo que alguns deles recebem nomes especiais. Os grupos mais comumente usados são:

Byte: um grupo de 8 bits, suficiente para armazenar os símbolos mais comuns da comunicação ocidental baseada em caracteres romanos.
Word: um grupo de 2 bytes (ou 16 bits). Em geral usado para armazenar números inteiros.

Os múltiplos são o kilo (K), o mega (M), o giga (G), o tera (T) e o peta (P) mas, diferentemente do sistema decimal, em que cada unidade é uma potência de 10, no sistema binário usamos potências de 2 para indicar essas magnitudes:

1 KB = 2^{10} bytes = 1024 bytes
1 MB = 2^{20} bytes = 1024 KB = 1048576 bytes
1 GB = 2^{30} bytes = 1024 MB = 1073741824 bytes
1 TB = 2^{40} bytes = 1024 GB
1 PB = 2^{50} bytes = 1024 TB

5. BIBLIOGRAFIA

MONTEIRO, M.A. **Introdução à Organização de Computadores**. 5ª. Ed. Rio de Janeiro: LTC, 2008.

MURDOCCA, M. J; HEURING, V.P. **Introdução à Arquitetura de Computadores**. S.I.: Ed. Campus, 2000.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 2ª.Ed. São Paulo: Prentice Hall, 2003.