Aula 3a – MSC – A Linguagem de Máquina

Vejamos as instruções do nosso MSC:

1) Entrada de Dados: `leia x`
2) Saída de Dados: `imprima x`
3) Movimento com ou sem cálculo de expressões aritméticas:
```
x ← c
y ← x+z
z ← x*x+y-1
d ← b*b-4*a*c
```
4) Condicionais:
```
se x>y então ...
       senão ...
se x+1<y*z então ...
           senão ...
se x=y então ...
```

Observe também que em todos os problemas resolvidos usamos apenas esses comandos usando rigorosamente a mesma forma.

Considere por exemplo o problema abaixo, já resolvido anteriormente:

Dados 2 números calcular a sua soma, diferença, produto e quociente.

```
leia x, y
a ← x + y
imprima a
a ← x - y
imprima a
a ← x * y
imprima a
a ← x / y
imprima a
```

Um computador real não "entende" diretamente os comandos acima.
Além disso, o programa deve ser armazenado na memória onde só podem ser colocados números.
A solução é codificar o programa acima. É o que chamamos de linguagem de máquina.

O computador "entende" a linguagem de máquina que é simplesmente uma codificação numérica de instruções elementares como aquelas que vimos em nosso MSC.

Vamos definir uma linguagem de máquina para o nosso MSC e verificar como a sequência de instruções acima pode ser codificada.

A Linguagem de Máquina do MSC

Vamos supor que a memória do nosso MSC possua 1000 posições (000 a 999).
Em cada posição vamos armazenar números de 5 dígitos com sinal.

Agora vamos às instruções. Cada uma delas terá um código numérico escolhido.

a) Entrada de dados – código 31
b) Saída de dados – código 41
c) Movimento com ou sem expressões aritméticas.

Aqui são necessárias algumas observações:
O MSC possui um registrador especial chamado de Acumulador. Os dados são movimentados da memória para o Acumulador e deste para a memória.
No Acumulador podemos:

- Somar valor que está em alguma posição de memória;
- Subtrair um valor que está em alguma posição de memória;
- Multiplicar por um valor que está em alguma posição de memória;
- Dividir por um valor que está em alguma posição de memória.

O resultado é sempre deixado no Acumulador.

Vamos então às instruções:
Movimenta um dado da memória para o Acumulador – código 11
Movimenta um dado do Acumulador para a memória – código 12
Soma um dado da memória ao Acumulador – código 21
Subtrai um dado da memória do Acumulador – código 22
Multiplica um dado da memória pelo Acumulador – código 23
Divide o Acumulador por um dado da memória – código 24

Com essas instruções, vamos codificar a seqüência acima.
Supondo que x, y e a ocupem respectivamente as posições de memória 900, 901 e 902 e que o programa comece na posição 000, vamos montar as instruções:

	Memória	Conteúdo
`leia x, y`	000	+31 900
	001	+31 901
`a ← x + y`	002	+11 900
	003	+21 901
	004	+12 902
`imprima a`	005	+41 902
`a ← x - y`	006	+11 900
	007	+22 901
	008	+12 902
`imprima a`	009	+41 902
`a ← x * y`	010	+11 900
	011	+23 901
	012	+12 902
`imprima a`	013	+41 902
`a ← x / y`	014	+11 900
	015	+24 901
	016	+12 902
`imprima a`	017	+41 902

Instruções de Desvio

Quando fazemos uma comparação (usando a instrução **`se`**), basta comparar o resultado com zero.
Por exemplo, **`x>y`** é o mesmo que **`x-y>0`**.

As instruções de desvio baseiam-se no valor do Acumulador e são as seguintes:

Desvia incondicionalmente – código 51
Desvia se o valor do Acumulador for maior que zero – código 52
Desvia se o valor do Acumulador for maior ou igual a zero – código 53

Desvia se o valor do Acumulador for diferente de zero – código 54
Desvia se o valor do Acumulador for igual a zero – código 55
Desvia se o valor do Acumulador for menor que zero – código 56
Desvia se o valor do Acumulador for menor ou igual a zero – código 57

Vamos introduzir também a instrução de parada – código 70

Considere agora o problema:

P5) Dados 2 números (supor distintos) calcular o maior.

```
leia a, b
se a > b
   então imprima a
   senão imprima b
```

Vamos traduzi-lo para linguagem de máquina, supondo que a e b ocupam respectivamente os endereços 500 e 501 e que o programa comece na posição 100:

	Memória	Conteúdo
leia a, b	100	+31 500
	101	+31 501
se a > b	102	+11 500
	103	+22 501
	104	+57 107
então imprima a	105	+41 500
	106	+51 108
senão imprima b	107	+41 501
	108	+70 000

Algoritmos

Vamos falar agora um pouco de algoritmos.

É um conceito matemático. Não é um conceito da computação.

Existem muitos exemplos conhecidos. Desde os mais simples:

a) Algoritmo da Soma de 2 números;
b) Algoritmo da Divisão entre 2 números;
c) Algoritmo para o cálculo de determinante de matriz;
d) Etc.

Até outros mais elaborados:

e) Algoritmo de Báscara para o cálculo das raízes de uma equação do segundo grau;
f) Algoritmo de Euclides para o cálculo do MDC;
g) Algoritmo de Briot-Ruffini para divisão de polinômio por (x-a);
h) Algoritmo de Newton para o cálculo da solução de uma equação;
i) Etc.

Uma caracterização bem simplificada:

Algoritmo é uma seqüência **bem definida** e **finita** de passos que levam a um **único resultado**.

Bem definida – Não pode haver dúvida ou mesmo interpretações distintas sobre cada passo do algoritmo.

Finita – Um algoritmo sempre termina.

Único resultado – O resultado do algoritmo é sempre o mesmo para uma mesma entrada.

Algoritmos e programação

Um programa em um computador é a expressão de um algoritmo. Esse algoritmo é expresso através de uma linguagem de programação.

Um algoritmo também pode ser expresso de outras maneiras:

1) Através de linguagem coloquial desde que bem descrito sem dar margem a interpretações diferentes;
2) Através de expressões matemáticas;
3) Qualquer outra maneira que não dê margem a dúvida sobre cada ação;

Neste curso, veremos como expressar ou escrever algoritmos através de uma linguagem de programação. Usaremos a linguagem C.