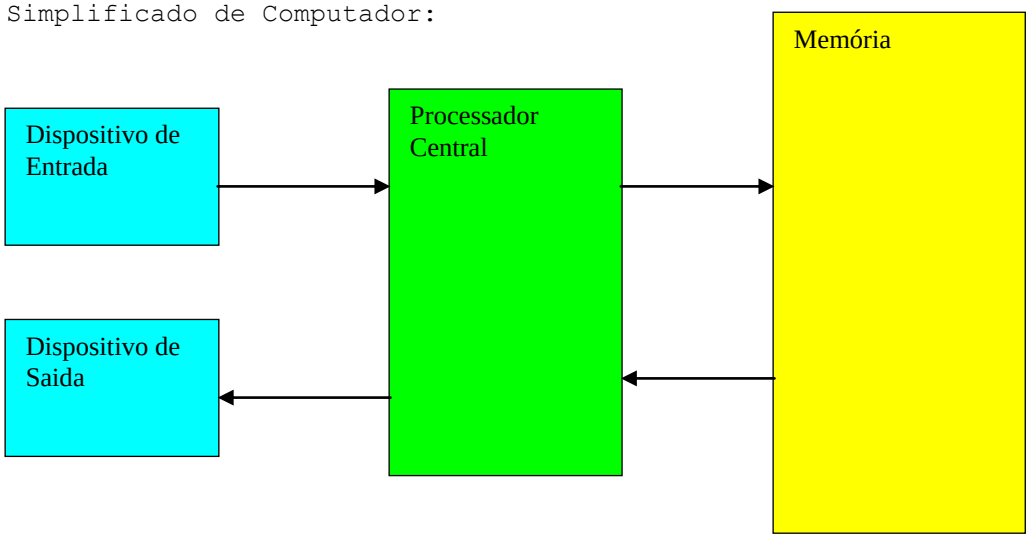
Aula 2 – Modelo Simplificado de Computador

Um computador pode ser esquematizado de maneira bastante simplificada da seguinte forma:

Modelo Simplificado de Computador:



As setas representam o fluxo de dados ou informações, entre os 4 elementos do MSC.

Dispositivos de Entrada:

Seu objetivo é obter dados que serão colocados na memória para que sejam posteriormente usados pelo processador em cálculos aritméticos ou lógicos.

Exemplos de dispositivos de entrada nos computadores reais:

- Teclado – cada caractere digitado é transferido para a memória do computador.
- Mouse – usado para marcar ou indicar elementos na tela do computador. A cada click no mouse a sua posição é transferida para a memória.
- Microfone – é possível, gravar arquivos com voz, ou mesmo falar no microfone quando se usa telefonia pelo computador.
- Scanner – imagens são digitalizadas e gravadas em arquivos.

Dispositivos de Saída:

Seu objetivo é obter dados da memória do computador e mostrá-los ou enviá-los para outros computadores ou dispositivos.

Exemplos de dispositivos de saída nos computadores reais:

- Impressora
- Vídeo
- Alto-falante
- Plotter

Dispositivos de Entrada e Saída:

Alguns dispositivos são de entrada e saída ao mesmo tempo, isto é, podem ser usados para obter ou para enviar dados do exterior para o computador.

Exemplos de dispositivos de entrada e saída nos computadores reais:

- Disquete
- Disco rígido
- CD ou DVD – em alguns computadores a unidade de CD ou DVD é apenas de leitura. Linha telefônica
- Rede Local
- Linhas de comunicação (dispositivos elétricos ou ópticos para transmissão de dados, voz e imagem)
- Vídeos especiais onde se pode teclar com os dedos (touch screen)
- Pen-Drive de memória (simula um disco)

Memória

Onde são armazenados os dados. Internamente somente se armazenam números. Para se armazenar textos há uma codificação em que cada caractere é representado por um número.

A memória pode ser entendida como dividida em elementos denominados de palavras, cada uma associada a um endereço.

0000	
0001	
0002	
0003	

: :
: :
: :

9996	
9997	
9998	
9999	

Bits, Bytes e Palavras

Nos computadores os dados são armazenados na forma binária, isto é como zeros e uns. A memória, portanto contém uma sequencia de zeros e uns.

Bit (Binary digIT)é a menor unidade de armazenamento e assume os valores 0 ou 1. A memória é na verdade uma seqüência de bits. Esses bits estão divididos em grupos de 8, denominados bytes. Assim a memória é também uma seqüência de bytes. Os bytes são divididos em grupos de 1, 2, 3, 4, ..., denominados de palavras. O tamanho da palavra depende do particular computador. Em geral quanto maior a palavra maior a potência e velocidade do computador, pois o computador movimenta sempre uma palavra por vez. Quando maior o seu tamanho, mais dados são movimentados ao mesmo tempo pelo computador.

Existem computadores com palavras de 1, 2, 3, 4 ou 8 bytes de palavra, ou seja, de 8 bits, 16 bits, 24 bits, 32 bits e de 64 bits de palavra. Hoje em dia os de 8 ou 16 bits são computadores usados para propósitos específicos, ou seja, computadores usados em controle de processos (processos fabris, controle de semáforo, terminais caixa, etc.).
Os computadores pessoais e notebooks mais comuns possuem palavras de 32 bits ou 64 bits.

Para exemplificar, suponhamos um computador de 8 bits. Em cada palavra podemos armazenar um número de 0 a 255, ou seja, em binário:
00000000 a 11111111

Um pouco de números na base 2 (binária)

Lembre-se da notação decimal, isto é como um número é escrito na base 10.

1346 significa $6 + 10*4 + 3*100 + 1*1000$, ou $6 + 10^1 + 3*10^2 + 1*10^3$

Na base 2 é a mesma coisa:

10101100 significa $0 + 0*2 + 1*4 + 1*8 + 0*16 + 1*32 + 0*64 + 1*128 = 172$

O programa também está na memória

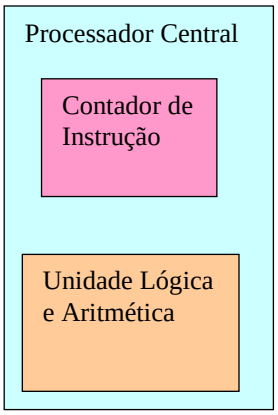
Além de armazenar os dados, o computador também armazena na memória, o que chamamos de programa, que nada mais é que uma seqüência de instruções que são entendidas pelo Processador Central. Tal seqüência é também um conjunto de números armazenados na memória.

Processador Central

É o que realmente realiza as operações e controla todo o fluxo de dados entre os dispositivos de entrada e saída de e para a memória.

Dentro do processador está a unidade lógica e aritmética que realiza as operações. Há também um contador que indica qual a próxima instrução que será executada, para que o processador ao executar uma instrução saiba qual a próxima, etc.

É no processador que realmente está a parte mais importante e mais complexa do computador, pois ele é responsável por executar cada uma das instruções elementares que constituem juntas um programa.



Instruções, Operações ou Comandos do Modelo Simplificado de Computador

Um programa de computador é dividido em instruções, operações ou comandos elementares. Vamos descrever algumas destas instruções para o nosso modelo simplificado de computador e com elas montar alguns programas. Usaremos principalmente a denominação instrução, porém são também operações ou comandos.

Já vimos que a memória é dividida em palavras. Por simplificação vamos dar nomes simbólicos a essas palavras, em vez de chamá-las pelo seu endereço numérico. Por exemplo, vamos dizer palavra x, y z de memória, ficando entendido que estamos nos referindo a palavras com endereços, por exemplo, 0101, 5436 e 2456.

Outra simplificação que faremos é que usaremos apenas números inteiros, isto é, o nosso MSC só trabalha com números inteiros. Assim podemos armazenar 0, 12, -567, etc. em sua memória, mas não podemos armazenar 1.234, -104.23, 1.2, etc.

1. Entrada

Ler um número e colocar na posição de memória x. Vamos indicar essa instrução por **leia x**. Assim:

leia z – lê um número e coloca na posição z de memória
leia a, b, c – lê 3 números colocando-os nas posições a, b e c de memória.

Dizemos que a instrução de leitura é destrutiva, pois irá destruir o conteúdo anterior da posição de memória colocando o novo valor. Assim, se x contem 25, e o valor lido for 32, x conterá 32 após a leitura.

Antes: x | 25 |

Comando: leia x (vamos supor que foi digitado o valor 32)

Depois: x | 32 |

2. Saida

Imprimir um número que está na posição x de memória. Vamos indicar esta instrução por **imprima x**. Assim:

imprima z – imprime um número que está na posição z de memória
imprima a, b, c, d – imprima 4 números seguidos que estão respectivamente nas posições a, b, c e d de memória

A instrução de saída, não é destrutiva, pois o conteúdo da posição de memória que foi impressa, permanece inalterado. Ou seja, é tirada uma cópia do valor da posição de memória para ser impresso.

Se x contem 25, após imprima x, continuará com o valor 25.

| 25 |

antes: x

comando: imprima x (o valor 0025 é mostrado no vídeo)

depois: x | 25 |

3. Movimento de dados de e para a memória

Uma instrução bastante comum é o movimento de um valor que está numa posição de memória para outra. Vamos indicar essa instrução da seguinte forma:

x ← y movimenta ou atribui o conteúdo da posição de memória y para a posição de memória x, ou simplesmente atribui y a x.

antes:	x	25	y	32
comando:	x ← y			
depois:	x	32	y	32

i ← j atribui j a i

antes:	i	100	j	- 202
comando:	i ← j			
depois:	i	- 202	j	- 202

Um caso particular é a atribuição de um valor constante a uma posição de memória:

x ← 0 atribui 0 a x
z ← 35 atribui 35 a z

Qual será o valor das posições de memória x, y e z depois da seguinte seqüência de instruções?

x ← -1
z ← x
y ← z
x ← y

Todas ficarão com –1

E após a seguinte seqüência?

x ← 0
z ← y
y ← x
x ← z

y fica com 0, mas x e z ficam com o valor anterior de y.

Agora, considere o seguinte problema: Trocar os valores das posições x e y, isto é, atribuir o valor que está em x para y e vice-versa.

Se fizermos simplesmente:

x ← y
y ← x

Antes:	x	100	y	200
Comando:	x ← y			
Depois:	x	200	y	200
Comando:	y ← x			
Depois:	x	200	y	200

Não adianta, pois ambos ficarão com o valor anterior de y. A solução é usar uma posição de memória auxiliar, por exemplo, z:

z ← x guarda o valor original de x em z
x ← y atribui y a x
y ← z atribui z a y, isto é, atribui o valor original de x a y

antes:	x	33	y	82	z	??
comando:	z ← x					
depois:	x	33	y	82	z	33
comando:	x ← y					
depois:	x	82	y	82	z	33
comando:	y ← z					
depois:	x	82	y	33	z	33

4. Cálculo de expressões aritméticas

O Processador Central possui internamente a Unidade Lógica e Aritmética que consegue fazer cálculos com números armazenados na memória do nosso MSC. O cálculo depois de realizado, deve ser armazenado em alguma outra posição de memória. Portanto o cálculo de uma expressão aritmética pressupõe também o movimento do resultado para uma nova posição de memória. Vamos indicar essa instrução da seguinte forma:

x ← expressão aritmética

Isto é, calcula a expressão aritmética e atribui o resultado a x.

A expressão aritmética pode ter em sua formação as operações elementares +, -, * e /, bem como constantes e outras posições de memória.

Exemplos:

x ← y + z

Soma o conteúdo da posição y com o conteúdo da posição z e coloca o resultado na posição de memória x, ou simplesmente, atribui a x o valor de y mais z.

Antes:	x	33	y	82	z	21
Comando:	x ← y + z					
Depois:	x	103	y	82	z	21

x ← y – 1

Atribui a x o valor de y menos 1.

Antes:	x	33	y	82
Comando:	x ← y - 1			
Depois:	x	81	y	82

x ← 3 * a + y

Atribui a x o valor de 3 vezes a mais y

Antes:	x	33	a	- 27	y	2
Comando:	x ← 3 * a + y					
Depois:	x	- 79	a	- 27	y	2

x ← y * y – 2 * y + 1

a ← (y – 1) * (y + 1) / (z – 1) * (z + 1)

x ← x + 1

Este exemplo é interessante, pois estamos atribuindo a x o seu valor anterior mais 1.

Antes:	x	25
Comando:	x ← x + 1	
		26

Depois: x

x ← x - 1

Análogo ao anterior está atribuindo a x o seu valor anterior menos 1.

Antes: x | 25 |

Comando: x ← x - 1

Depois: x | 24 |

x ← x + y

Análogo ao anterior, estamos atribuindo a x o seu valor anterior mais y.

Antes: x | 33 | y | 82 |

Comando: x ← x + y

Depois: x | 115 | y | 82 |

A atribuição simples do tipo:

x ← y

É um caso particular da atribuição de expressões aritméticas, ou seja, y sozinho é uma expressão aritmética também.

Portanto temos na verdade 3 tipos de instruções, operações ou comandos, ou seja:

a) Entrada
b) Saída
c) Atribuição

Programas elementares usando o MSC

Com esses 3 tipos de instruções, vamos fazer alguns programas usando o nosso MSC.

Convenções de enunciado

Em primeiro lugar algumas convenções para o enunciado dos nossos problemas:

1) Quando falamos no enunciado, que são **dados alguns números**, tais números devem ser lidos pelo programa, isto é, deve haver um comando **leia** para cada um desses dados.
2) Os resultados calculados pelo programa devem ser impressos, isto é, deve haver um comando **imprima** para cada um desses resultados, embora não se diga explicitamente para imprimir o resultado.

Isso posto vamos aos problemas.

P1) Dados 2 números, calcular a sua soma.

```
leia x
leia y
z ← x + y
imprima z
```

ou ainda

```
leia x, y
z ← x + y
imprima z
```

P2) Dados 2 números, calcular a sua soma, subtração, multiplicação e divisão.

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

Outra forma:

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

Outra forma:

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

Neste último caso, como a posição a é usada só para armazenar o cálculo, não há problema em armazenar todos os cálculos na mesma posição desde que se imprima o resultado após cada um dos cálculos.

P3) Dados 2 números calcular a média

```
leia x, y
z ← (x + y) / 2
```

```
imprima z
```

P4) Dados 2 triplas de números a1,b1,c1 e a2, b2 e c2, determinar x e y onde
a1x+b1y=c1
a2x+b2y=c2
Isto é, a solução do sistema de 2 equações a 2 incógnitas. Supor que existe solução.

Lembre que x=(c1.b2-c2.b1)/(a1.b2-a2.b1) e y=(a1.c2-a2.c1)/(a1.b2-a2.b1).

```
leia a1, b1, c1, a2, b2, c2
x ← (c1*b2-c2*b1)/(a1*b2-a2*b1)
y ← (a1*c2-a2*c1)/(a1*b2-a2*b1)
imprima x, y
```