

## Solução - Exercícios Processadores

### 1- Qual as funções da **Unidade Aritmética e Lógica (ULA)**?

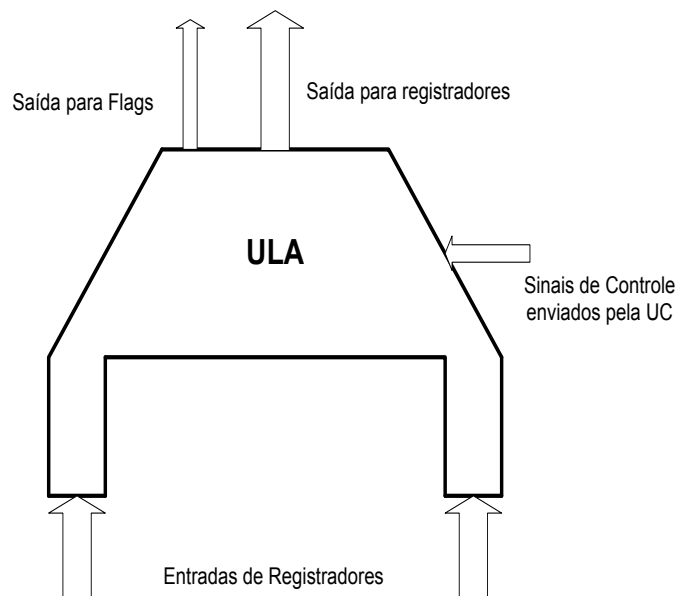
A ULA é o dispositivo da CPU que executa operações tais como:

- Adição
- Subtração
- Multiplicação
- Divisão
- Incremento
- Decremento
- Operação lógica AND
- Operação lógica OR
- Operação lógica XOR
- Operação complemento
- Deslocamento à direita
- Deslocamento à esquerda

Tais operações podem utilizar dois operandos (adição, and, etc..) ou apenas um valor (como, por exemplo, as operações de complemento ou deslocamento).

A ULA é um aglomerado de circuitos lógicos e componentes eletrônicos simples que, integrados, realizam as operações mencionadas acima. Ela pode ser uma parte pequena da pastilha do processador, usada em pequenos sistemas, ou pode compreender um considerável conjunto de componentes lógicos de alta velocidade.

A ULA é responsável por processar as operações aritméticas e lógicas requeridas nas instruções de máquina (ver Figura 1)



**Figura 1**

As operações matemáticas mais comuns são adição, deslocamento, rotação e operações lógicas realizadas sobre dois operandos e a de complemento sobre um operando. Outras operações, como multiplicação, divisão, etc, podem ser executadas por meio de microprogramas, ou ainda, podem ser realizadas por outro processador interno, ou externo (co-processador). A ULA recebe um ou dois valores de entrada. Estes valores passam pelos

circuitos internos da ULA, de acordo com o sinal enviado pela UC e apresentam um resultado na saída da própria ULA. De acordo com o resultado, a ULA envia sinais para o registrador de FLAG, informando o que ocorreu, como por exemplo, se o resultado é igual a zero, se houve vai-um, etc. Na ULA, encontramos os circuitos somadores, subtratores, comparadores, registradores de deslocamento, etc., que variam de processador para processador.

2- Descreva as funções básicas da **Unidade Central de Processamento (UCP)** indicando os seus componentes principais

O processador é o componente principal de um sistema de computação. É responsável pela realização das operações de **processamento** e de **controle**, durante a execução de um programa.

Um programa para ser executado pelo processador deve ser formado por uma série de *instruções de máquina*. Para iniciar a execução de um programa, as instruções devem ser armazenadas na memória principal.

A função da UCP consiste em:

- a) Buscar uma instrução da memória (operação de leitura), uma de cada vez;
- b) Decodificar a instrução, para determinar o tipo de operação a ser realizada: soma, multiplicação, leitura ou escrita na memória, mover dados de um lugar para o outro, etc;
- c) Se as instruções utilizarem dados, buscá-los onde estiverem armazenados para que a UCP possa processá-los;
- d) Executar a operação com os dados e guardar o resultado, se houver, no local especificado;
- e) Reiniciar o processo, buscando a próxima instrução.

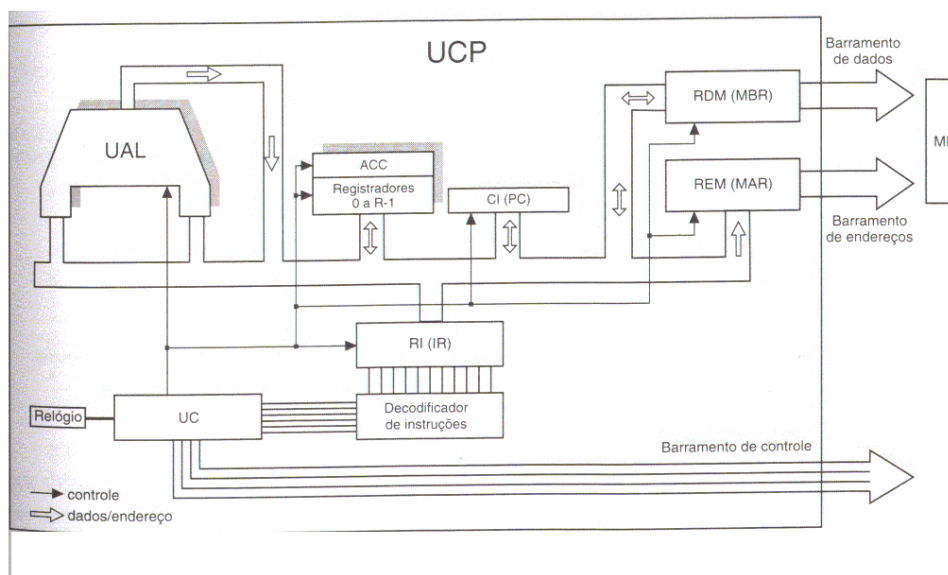
Esta etapa consiste no *ciclo de instrução*. Este ciclo se repete até que seja encontrada alguma instrução de parada, ou o sistema seja desligado ou ocorra algum tipo de erro.

A UCP é projetada para executar sucessivamente, operações básicas, na sequência definida pelo programa.

As atividades realizadas pela CPU podem ser divididas em duas categorias:

- 1) Processamento
- 2) Controle

A Figura 1 mostra o diagrama em blocos de uma UCP, contendo os componentes principais



**Figura 1**

Esta figura é baseada em uma UCP simplificada, para facilitar a compreensão sobre o assunto. Um processador real como os microprocessadores Intel, possuem vários elementos adicionais.

O processamento é responsável pela realização de atividades relacionadas com a execução de uma operação. O controle diz respeito às atividades de busca, interpretação e controle da execução das instruções, bem como o controle dos demais componentes do sistema de computação (memória, E/S).

3- Quais são as funções da **Unidade de Controle** de um processador?

A área de controle de uma UCP é a parte que realiza as seguintes atividades:

- a) Busca de instrução que será executada, armazenando-a em um registrador especialmente projetado para esta finalidade.
- b) Interpretação das ações que serão processadas durante a execução da instrução. Por exemplo, se é uma soma, um complemento, etc.
- c) Geração dos sinais de controle apropriados para a execução da instrução identificada. Sinais estes que são enviados aos diversos componentes do sistema, sejam eles internos ou externos.

A parte de controle é projetada para identificar o que deve ser feito, como deve ser feito e enviar os sinais apropriados para os dispositivos que serão utilizados, no tempo adequado.

Unidade de Controle. Os dispositivos da UCP responsáveis pelo controle são:

- Unidade de Controle (UC);
- Decodificador;
- Registrador de Instrução (IR)
- Contador de Instruções (PC)
- Relógio
- Registradores de Endereço (REM) e de dados (RDM).

A Unidade de Controle possui a circuitaria necessária para realizar a movimentação de dados e instruções de e para a UCP, enviando sinais de controle em instantes de tempo programado. Este dispositivo controla, por exemplo, qual operação será executada pela ULA. Os sinais de controle enviados pela UC ocorrem em vários instantes durante a realização de um ciclo de instrução.

4- Quando se fala que um determinado computador é de 8, 16, 32 ou 64 bits, a que está se referindo?

Ao tamanho da palavra do processador, ou seja, o número de bits que o processador é capaz de manipular. Isto diz respeito ao número de bits (tamanho) dos registradores internos, o tamanho em bits que a ULA é capaz de processar, o tamanho do barramento de dados.

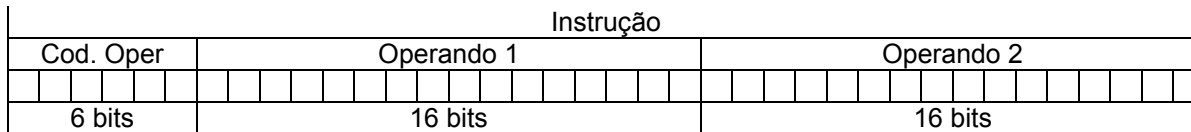
5- Considere um computador que possua uma UCP com o PC (ou CI) de 16 bits e IR (ou RI) de 38 bits. Suas instruções possuem dois operandos do mesmo tamanho (16 bits cada), além do código de operação.

- 5.1. Qual o tamanho da instrução?
- 5.2. Qual o tamanho do campo do código de operação?
- 5.3. Considerando que a configuração básica desta máquina é de 16 K bytes de memória, até que tamanho pode esta memória ser expandida?

5.1. O tamanho da instrução é igual ao tamanho do registrador de instrução, uma vez que este armazena a instrução que está sendo executada. Portanto, o tamanho da instrução é de 38 bits.

5.2. Como a instrução tem tamanho de 38 bits e possuem dois operandos de 16 bits cada, o tamanho do código de operação é:

$$38 - 16 - 16 = 6 \text{ bits}$$



5.3. O tamanho do registrador PC, que é usado para apontar para a próxima instrução a ser executada tem tamanho de 16 bits. Com 16 bits, temos um total de  $2^{16} = 64K$  combinações possíveis que vai de 0 até  $2^{16} - 1$ , ou seja, de 0 a 64K - 1. Portanto, podemos expandir a memória até 64K palavras.

6- Um computador tem uma REM de 16 bits e um barramento de dados de 20 bits. Possui instruções de 1 operando, todas do tamanho de uma *palavra de memória* e de mesmo tamanho da *palavra do processador*. Ele foi adquirido com uma placa de 4K de memória.

6.1. Qual o tamanho, em **bits**, do RDM e PC?

6.2. É possível aumentar a capacidade de armazenamento desta memória? Até quanto? Por quê?

6.3. Qual a quantidade máxima de instruções de máquina que poderia existir neste computador?

6.1. Como o barramento de dados tem tamanho de 20 bits, o registrador RDM terá o mesmo tamanho, uma vez que este registrador faz a interface entre o processador e o barramento de dados. O registrador REM, é usado para interconectar o processador ao barramento de endereço. Como REM possui 16 bits, o barramento de endereço também terá 16 bits.

6.2. Como o barramento de endereço tem 16 bits, podemos ter uma memória com  $2^{16} = 64K$  palavras. Portanto é possível aumentar a quantidade de palavras de memória até 64K.

6.3. Supondo que o operando faça referência à endereços de memória, este deve ter tamanho de 16 bits. Como a instrução possui o mesmo tamanho da palavra de memória e palavra do processador, chegamos à conclusão que o tamanho da instrução é de 20 bits. O tamanho do código de operação é dado por  $20 - 16 = 4$  bits. Com 4 bits, temos um total de 16 combinações possíveis e portanto, podemos ter no máximo um total de 16 instruções de máquina.

7- Um computador possui um conjunto de 128 instruções de um operando; supondo que sua memória tenha capacidade de armazenar 512 palavras e que cada instrução tem o tamanho de uma palavra do processador e palavra de memória, responda as perguntas a seguir:

7.1 Qual o tamanho em bits do REM, RDM, RI, ACC e PC?

7.2 Qual a capacidade da memória em bytes?

7.3 Se alterarmos o tamanho das instruções para 17 bits, mantendo inalterado o tamanho do REM, quantas novas instruções poderiam ser criadas?

7.1. Como a memória tem 512 palavras, ou seja,  $2^9$  palavras chegamos à conclusão que serão necessárias 9 linhas para endereçar todas estas palavras de memória. Portanto o barramento de endereço e o REM são de 9 bits.

Ender (Binário)	Ender (Hexa)	Ender (Decimal)	Conteúdo
00000000	000	0	
00000001	001	1	
00000010	002	2	
...	...	...	
11111111	1FF	511	

Com 512 palavras de memória, chegamos à conclusão que o operando deverá ter um tamanho de 9 bits, para endereçar qualquer posição (ou palavra) de memória. Como temos um conjunto de 128 instruções, necessitamos de  $\log_2^{128} = 7$  bits para o código de operação. Portanto, o tamanho da instrução será de 16 bits: 7 (para o código de operação) mais 9 (para o operando).

Como o tamanho da instrução é igual ao tamanho da palavra do processador e tamanho da palavra de memória, chegamos à conclusão que o barramento de dados, o RI, o ACC e o RDM terão também 16 bits.

O tamanho do PC é igual ao tamanho do barramento de endereço, ou seja, 9 bits.

7.2. O tamanho da palavra de memória é de 16 bits, ou seja, 2 bytes. Temos, portanto, uma memória com 512 palavras de memória de 16 bits cada, ou 1024 palavras de memória de 8 bits (1 byte cada).

7.3. Neste caso, aumentaremos o tamanho do campo código de operação de 7 para 8 bits, pois o operando permanece com o mesmo tamanho (9 bits). Com 8 bits temos um total de  $2^8 = 256$  combinações possíveis, ou seja, um total de 256 instruções.

8- Seria possível realizar o projeto de um processador onde o tamanho em bits do PC fosse diferente do REM? Nesse caso, qual dos dois registradores deveria ter o maior tamanho? Por quê?

Sim. Observe a arquitetura do 8086. Temos o IP (ou PC) com tamanho de 16 bits, que nos dá no máximo 64K combinações possíveis. O REM e o Barramento de Dados são de 20 bits, que resulta em  $2^{20} = 1\text{M}$  palavras de memória. O 8086 utiliza dois registradores para compor o endereço de 20 bits: o registrador de segmento e o registrador *offset*, ambos com tamanho de 16 bits.

O registrador que deveria ter o menor tamanho é o PC, pois podemos combiná-lo com outro registrador, para compor o endereço de 20 bits.

9- Considere um computador com 64K palavras de memória, instruções com um operando, tendo possibilidade de ter um conjunto com 256 instruções de máquina. Considerando que cada instrução tem o tamanho de uma palavra de memória, que é do mesmo tamanho da palavra do processador, qual o tamanho, em bits, do ACC, PC e RDM? Qual é o total de bits da memória?

Com 64K palavras de memória, serão necessárias  $\log_2^{64K} = 16$  linhas de endereço. Como temos um total de 256 instruções, precisamos de 8 bits para compor o código de operação, lembrando que  $2^8 = 256$ . É dado que as instruções possuem um único operando e este deve ter o mesmo tamanho do barramento de endereço, para referenciar todas as 64K palavras de memória. Portanto, a instrução terá 24 bits; 8 para o código de operação e 16 para o operando.

A palavra de memória e a palavra do processador possuem o mesmo tamanho da instrução. Portanto o ACC e o RDM terão 24 bits, enquanto o PC terá 16 bits. O total de bits da memória é dado por: número de palavras x tamanho da palavra:  $64K \times 24 = 1.572.864$  bits

10- Considere as instruções definidas a seguir, todas elas com apenas um operando:

Instrução	Significado
LDA Op	$ACC \leftarrow Op$
STA Op	$Op \leftarrow ACC$
ADD Op	$ACC \leftarrow ACC + Op$
SUB Op	$ACC \leftarrow ACC - Op$
MUL Op	$ACC \leftarrow ACC * Op$
DIV Op	$ACC \leftarrow ACC / Op$

Obter a equação que resultou no programa descrito a seguir:

```
1:   LDA   A
2:   ADD   C
3:   STA   X
4:   LDA   B
5:   MUL   D
6:   SUB   E
7:   STA   Y
```

```

8:    LDA    X
9:    ADD    Y
10:   DIV    F
11:   STA    X

```

1:	ACC = A	
2:	ACC = ACC + C	ACC = A+C
3:	X = ACC	X = A + C
4:	ACC = B	
5:	ACC = ACC * D	ACC = B * D
6:	ACC = ACC - E	ACC = B * D - E
7:	Y = ACC	Y = B * D - E
8:	ACC = X	ACC = A + C
9:	ACC = ACC + Y	ACC = X + Y = (A + C) + Y = (A + C) + (B * D - E)
10:	ACC = ACC / F	ACC = ((A + C) + (B * D - E))/F
11:	X = ACC	X = ((A + C) + (B * D - E))/F

11- Utilizando as instruções descritas na tabela do exercício 12, escreva os programas para as equações a seguir:

11.1.  $X = A + (B * (C - A) + (D - E / B) * D)$

```

LDA    C
SUB     A
MUL     B
STA     T
LDA     E
DIV     B
STA     Y
LDA     D
SUB     Y
MUL     D
ADD     T
ADD     A
STA     X

```

11.2.  $Y = A + B * (C - D * (E / (B - F)) + B)$

```

LDA     B
SUB     F
STA     T
LDA     E
DIV     T
MUL     D
STA     T
LDA     C
SUB     T
ADD     B
MUL     B
ADD     A
STA     Y

```

12- Seja a memória a seguir:

Endereço (em Hexa)	Conteúdo
--------------------	----------

00	
01	1C6
02	4C7
03	2FE
⋮	
C6	1B7
C7	1B6
⋮	
FE	
FF	

Valores iniciais:

PC = 01H  
IR = 317H  
ACC = 20BH  
REM = B5H  
RDM = 65FH

A instrução 1C6 = (0001 1100 0110)<sub>2</sub>. O código de operação 0001 representa a instrução LDA OP. Portanto teremos a instrução: LDA C6, ou seja, ACC = [C6].

A próxima instrução: 4C7 = (0100 1100 0111)<sub>2</sub>. O código de operação 0100 representa a instrução SUB OP. Portanto teremos a instrução: SUB C7, ou seja, ACC = ACC – [C7].

A instrução 2FE = (0010 1111 1110)<sub>2</sub>. O código de operação 0010 representa a instrução STR OP. Portanto temos a instrução STR FE: [FE] = ACC.

O programa em Assembly é dado por:

LDA C6  
SUB C7  
STR FE

O endereço da primeira instrução a ser executada está armazenado no registrador PC. Considerando os valores iniciais, a primeira instrução a ser executada está no endereço 01H, que é 1C6. O valor contido em PC é passado para o REM que em seguida, é enviado para a memória, através do Barramento de Endereço. A memória irá decodificar este endereço e transferir para o processador, através do Barramento de Dados, o valor contido no endereço 01H. Este valor será armazenado em RDM e em seguida transferido para o Registrador de Instrução (RI), para que possa ser decodificado e então executado. Esta instrução irá armazenar no registrador ACC, o valor contido no endereço C6. Após a decodificação desta instrução, será realizada mais uma operação de leitura, para buscar o conteúdo do endereço C6 e armazená-lo em ACC.

ACC = [C6] → ACC = 1B7.

De acordo com o ciclo de instrução, o valor do registrador PC é incrementado, passando para 02H.

O valor contido em PC (02H) será transferido para o registrador REM para então ser enviado para a memória. A memória irá decodificar este endereço e enviar para o processador o valor 4C7, que será temporariamente armazenado em RDM e em seguida, transferido para o registrador RI. Esta instrução será decodificada e uma nova operação de leitura será realizada no endereço C7. O valor contido em C7 será subtraído do valor do ACC. Portanto teremos: ACC = ACC – [C7] → ACC = 1B7H – 1B6H → ACC = 001H.

O valor de PC é incrementado, passando para 03H.

Finalmente, a instrução STR FE, irá armazenar o valor contido em ACC no endereço FE. Para isso, o valor de PC é novamente transferido para REM, que será enviado para a memória. O valor contido no endereço 03H será

enviado para o processador, sendo então armazenado em RDM e em seguida transferido para o registrador RI. Será feita uma operação de escrita no endereço FE.

Valores recebidos pelos registradores durante a execução do programa:

PC = 01H, 02H, 03H, 04H  
IR = 317H, 1C6H, 4C7H, 2FEH,  
ACC = 20BH, 1B7H, (1B7H – 1B6H = 001H),  
REM = B5H, 01H, C6H, 02H, C7H, 03H, FEH  
RDM = 65FH, 1C6H, 1B7H, 4C7H, 1B6H, 2FEH, 001H

Memória após a execução das instruções:

Endereço (em Hexa)	Conteúdo
00	
01	1C6
02	4C7
03	2FE
⋮	
C6	1B7
C7	1B6
⋮	
FE	<b>001</b>
FF	