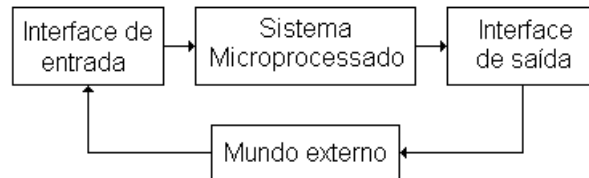


1. Sistemas Microprocessados

Um sistema microprocessado é aquele baseado em um microprocessador, dispositivo capaz de executar instruções e tomar decisões de acordo com sua programação.

Tipicamente, um sistema microprocessado precisa receber informações do mundo externo, processá-las e retornar alguma resposta ou sinal, conforme mostra a figura abaixo.



Um sistema microprocessado é muitas vezes também chamado de **sistema computacional**, devido à sua capacidade de processar informações.

1.1. Microprocessadores

Os microprocessadores são o “cérebro” de um sistema computacional. São circuitos integrados passíveis de serem programados para executar uma tarefa predefinida, basicamente manipulando e processando dados. Os outros componentes do sistema, como as memórias ROM (genericamente) e RAM e as interfaces de entrada e saída, existem somente para fazer uma ligação entre o mundo externo e o processador.

Seu funcionamento básico é buscar e executar instruções existentes na memória. São instruções simples, como operações aritméticas e lógicas, leituras e escritas na memória, comparações e movimentos de dados. Essas instruções, quando agrupadas, formam o que chamamos de programas.

Para que um processador seja capaz de executar um programa, ele deverá ser capaz de reconhecer as instruções presentes no programa. Dessa forma, o programa escrito para um microprocessador não é entendido por outro, a não ser que sejam de uma mesma família.

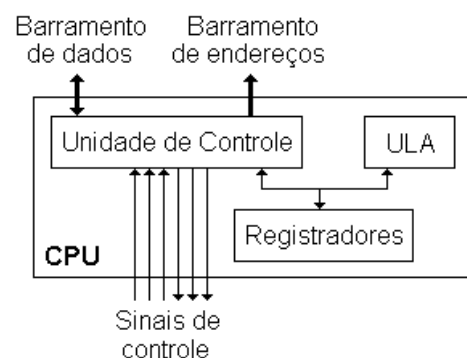
Os microprocessadores podem ser classificados pelo tamanho da palavra (ou comprimento, em bits, da unidade de informação) que são capazes de processar de uma só vez. O primeiro microprocessador comercial foi lançado em 1971 pela Intel, o 4004, com apenas 4 bits, desenvolvido para uma calculadora. Em seguida, houve uma evolução para os de 8 bits, quando esses componentes se popularizaram e passaram a ser altamente utilizados pela indústria. Na sequência, vieram os microprocessadores de 16 bits e 32 bits e, mais recentemente, os de 64 bits.

Regra geral, a velocidade de operação e o tamanho da palavra indicam o desempenho de um determinado processador.

Muitas vezes, dentro de um sistema computacional, os microprocessadores recebem o nome de CPU (*Central Processing Unit*) ou, em português, UCP (Unidade Central de Processamento). Essa denominação é um legado histórico já que nos primeiros sistemas computacionais não existia a figura do microprocessador e suas funções eram realizadas por circuitos distintos. Para agrupar esses circuitos em uma unidade funcional, foi criado o nome de CPU. Com o passar dos anos e conseqüente evolução dos sistemas e da eletrônica, esses circuitos foram agrupados em um único componente, o microprocessador.

1.1.1. Componentes principais de um microprocessador

Simplificadamente, podemos dizer que um microprocessador é composto por três unidades básicas: a unidade de controle (UC), a unidade lógica e aritmética (ULA) e por um banco de registradores, que armazenam informações necessárias ao correto funcionamento do sistema. Alguns desses registradores podem estar dentro da unidade de controle, dentro da ULA, ou fora delas, dependendo de sua função.



1.1.1.1. Unidade de controle

Responsável pelas seguintes tarefas:

- busca de instruções na memória, gerando a sinalização adequada;
- decodificação e execução das instruções;
- acesso aos dispositivos externos;
- controle da pilha nas chamadas de subrotinas e tratamento de interrupções.

Dentre os sinais que a unidade de controle pode gerar para acessar a memória e dispositivos externos ao processador, podemos agrupá-los em três grupos básicos:

- **Barramento de dados:** usado para receber os dados provenientes da memória (sentido memória → processador) ou de algum dispositivo externo, como também na escrita desses dados (sentido processador → memória).
 - sinais bidirecionais
 - determinam o “tipo” de processador (8 bits, 16 bits, etc.)
- **Barramento de endereços:** fornecem o endereço da posição de memória, ou do dispositivo de entrada/saída (E/S), onde o dado será lido ou escrito.
 - sinais unidirecionais (sempre fornecidos pelo processador)
 - determinam a capacidade máxima de memória que o processador pode acessar (ou endereçar). Por exemplo, com 16 bits pode-se endereçar $2^{16} = 65.536$ posições, ou seja, uma memória de 64 Kbytes.
- **Sinais de controle:** indicam se a operação a ser realizada é de leitura ou escrita, se o acesso será na memória ROM ou RAM, se o acesso será para a memória ou para um dispositivo de E/S, se o processador deve aguardar um tempo antes de ler o dado porque o dispositivo é lento, etc.
 - sinais de entrada e saída
 - controlam a comunicação com dispositivos externos ao processador

Obs.: Barramento é uma “coleção” de “fios” paralelos (ou um conjunto de sinais) usados para transmitir informações de um mesmo tipo.

1.1.1.2. Unidade lógica e aritmética (ULA)

Como o próprio nome já diz, realiza as operações lógicas, como AND, OR, XOR, SHIFT, ROTATE, SWAP, e as aritméticas, como ADIÇÃO, SUBTRAÇÃO, INCREMENTO, DECREMENTO, MULTIPLICAÇÃO, DIVISÃO, dentro do microprocessador.

1.1.1.3. Conjunto de registradores

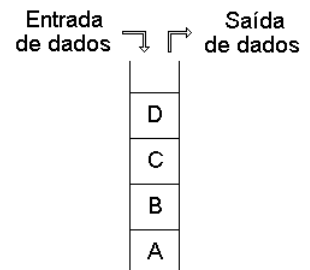
Memórias de altíssima velocidade usadas para armazenamento temporário dos resultados de diversas operações. Podem armazenar um ou mais bytes (registradores de 8 bits, registradores de 16 bits, etc.). Se um registrador for de **8 bits**, o valor **máximo** que ele pode armazenar é **255** (ou **FFH**). Os principais registradores são:

- *Accumulator* (**Acc**) ou acumulador: possui uma ligação direta com a ULA e é muito utilizado nas operações lógicas e aritméticas para servir como um dos valores nessas operações e também armazenar o resultado delas.
- *Status Register* (**STATUS**) ou registrador de estado: contém diversos **bits**, também conhecidos como **flags**, que indicam diversos estados ou a ocorrência de eventos dentro do processador. Dentre as principais **flags**, temos:
 - **Flag Z** (zero): é colocada em 1 quando o resultado de uma operação é igual a zero.
 - **Flag C** (*carry*): é colocada em 1 quando ocorre um “vai um” em uma soma. É afetada também em operações de subtração e por algumas funções lógicas (como rotate).

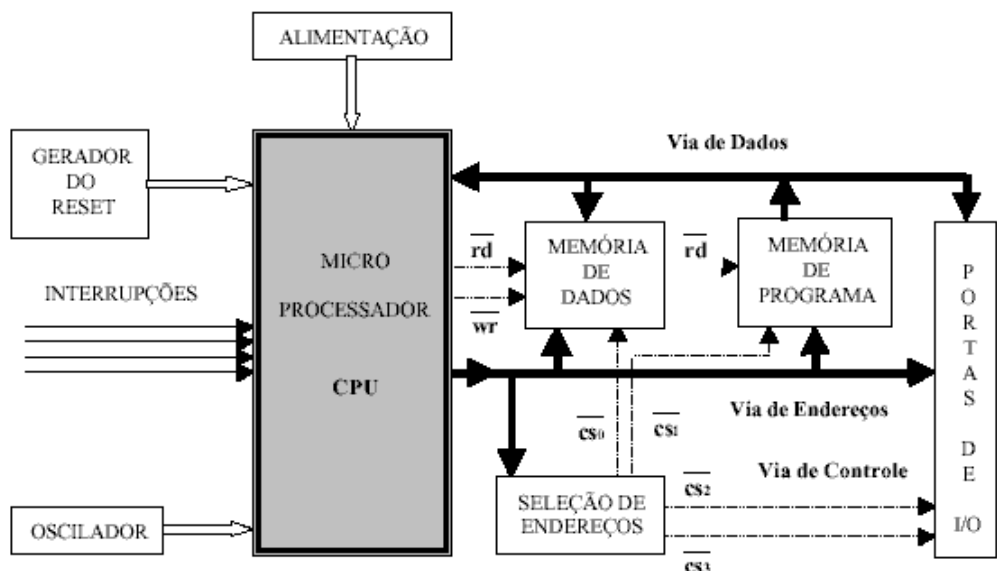
- **Flag DC** (*digit carry*): é colocada em 1 quando ocorre um “vai um” do nibble inferior para o nibble superior em uma soma. Usado nas operações com valores em BCD.
- **Flag N ou S** (*negative* ou *sign*): indica que o resultado de uma operação matemática foi negativo.
- **Flag P** (*parity*): indica se a quantidade de 1's do último resultado gerado na ULA é par ou ímpar.
- **Program Counter (PC)** ou ponteiro de programa: registrador especial que controla a execução de um programa dentro de um processador. Ele aponta para a posição na memória de programa onde será buscada a próxima instrução a ser executada pelo processador.
- **Instruction Register (IR)** ou registrador de instruções: recebe a instrução que foi lida da memória para que ela seja decodificada e executada. Normalmente se encontra na unidade de controle.
- **Stack Pointer (SP)** ou ponteiro de pilha: efetua o gerenciamento da pilha nas chamadas de subrotinas e no tratamento das interrupções.

1.1.1.4. Pilha:

Estrutura de dados do tipo LIFO (*last in first out*) utilizada para armazenamentos temporários. Sua principal função é armazenar o conteúdo do registrador PC nas chamadas de sub-rotinas ou nos atendimentos a interrupções, para que o programa retorne para a posição correta do programa na qual se encontrava antes de ser desviado ou interrompido. O controle de acesso é feito pelo registrador SP (*Stack Pointer*).




2. Sistema Computacional Genérico



Além do microprocessador, um sistema computacional típico necessita de diversos circuitos auxiliares para poder operar corretamente e se “comunicar” com o mundo externo. A figura acima ilustra um sistema microprocessado genérico e os diversos componentes do mesmo.

Conforme pode ser visto, além do microprocessador, o sistema necessita ainda de um circuito de *reset*, um circuito oscilador, controle de interrupções, fonte de alimentação, memória de programa (em geral, uma memória ROM), memória de dados (em geral, uma memória RAM), um circuito de seleção e portas de entrada / saída (também conhecidas como portas de *input / output* ou I/O).

2.1. Relógio (*clock*)

Onda quadrada com uma frequência determinada e fixa proveniente de um circuito oscilador e responsável por sincronizar todas as operações dentro e fora de um microprocessador, como leitura e escrita na memória, decodificação e execução de uma instrução. 

2.2. Oscilador

Circuito eletrônico responsável pela geração do sinal de relógio (*clock*). A frequência de oscilação do circuito (F_{osc}) pode ser determinada por diversos componentes, tais como: redes RC (resistor e capacitor), ressonadores cerâmicos e, principalmente, por cristais de quartzo.

2.3. Cristal

Componente eletrônico cujo elemento principal é um cristal de quartzo e que é colocado em um circuito oscilador para determinar a frequência de oscilação desse circuito. O cristal, quando alimentado, vibra em uma frequência fixa e essa vibração é captada e amplificada pelo circuito oscilador para gerar o sinal de relógio (*clock*).

2.4. Ciclos básicos de um processador

Para que o processador execute uma instrução de um programa, é necessário dividir essa tarefa em algumas etapas essenciais. Essas etapas são chamadas ciclos básicos do processador:

- **busca:** o processador acessa a memória de programa, cujo endereço está contido no registrador PC (*Program Counter*), e lê o código (*opcode*) da próxima instrução a ser executada;
- **decodificação:** a instrução lida é transferida para o registrador de instruções IR (*Instruction Register*) e é decodificada para que o processador saiba que operações deve realizar;
- **execução:** as operações indicadas no código da instrução são executadas pelo processador. A unidade de controle ativa os circuitos necessários, inclusive a ULA, no caso de operações lógicas e aritméticas;
- **escrita:** se a execução da instrução resultar em algum valor que deva ser escrito na memória, essa tarefa é realizada nesse ciclo.

Obs.: Esses ciclos podem variar de acordo com a arquitetura do processador. Por exemplo, em alguns processadores, o ciclo de decodificação e execução são feitos em um passo só.

2.5. Ciclo de máquina (CM)

É uma unidade de referência para se calcular o tempo gasto por um processador para executar uma instrução, ou seja, realizar seus ciclos básicos (busca, decodificação, execução e escrita). É um múltiplo do período do oscilador, $CM = n \times T_{osc}$. Como $T_{osc} = 1 / F_{osc}$, então, $CM = n / F_{osc}$. CM pode variar de acordo com a arquitetura do processador.

Por exemplo, nos microcontroladores da família PIC, $n = 4$, ou seja, se a frequência de oscilação (que é dada pelo cristal) for igual a 4 MHz, temos:

$$F_{osc} = 4 \text{ MHz} \Rightarrow CM = 4 / F_{osc} \Rightarrow CM = 4 / 4 \text{ MHz} \Rightarrow \mathbf{CM = 1 \mu seg}$$

Já para microcontroladores da família 8051, $n = 12$. No manual do processador (*datasheet*) estará indicado quantos ciclos de máquina cada instrução necessita para ser executada.

2.6. Pipeline

É uma técnica utilizada para aumentar a velocidade de um processador. As operações são decompostas em estágios processados em série por unidades separadas de hardware, de maneira idêntica a uma linha de montagem industrial. Assim, os ciclos básicos do processador são executados por unidades distintas de hardware. Em cada instante de tempo, enquanto uma unidade

está buscando uma instrução, a outra está decodificando a instrução buscada anteriormente e assim por diante.

T	T+1	T+2	T+3	T+4	T+5
Busca Inst. n	Decodificação Inst. n	Execução Inst. n			
	Busca Inst. n+1	Decodificação Inst. n+1	Execução Inst. n+1		
		Busca Inst. n+2	Decodificação Inst. n+2	Execução Inst. n+2	
			Busca Inst. n+3	Decodificação Inst. n+3	Execução Inst. n+3

2.7. Portas de E/S (entrada / saída) ou I/O (input / output)

É o canal de acesso ao mundo externo para o processador. É por intermédio delas que o processador lê sinais externos e atua em dispositivos nele conectados.

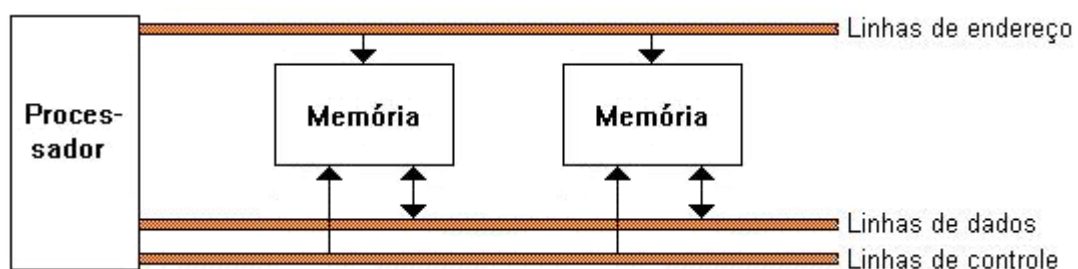
2.8. Memórias

Por definição, são dispositivos semicondutores que armazenam informações na forma binária. Os bits das informações armazenadas podem ser acessados quando no procedimento de leitura, ou gravados / substituídos quando no procedimento de escrita ou armazenamento.

Nos Sistemas Computacionais, entende-se como memória os dispositivos que armazenam os dados com os quais o processador trabalha. Há, essencialmente, duas categorias de memórias: **ROM** (*Read-Only Memory* ou Memória Somente de Leitura), que não perde sua informação na ausência de energia, ou seja, é uma memória não-volátil; e **RAM** (*Random-Access Memory* ou Memória de Acesso Aleatório), que permite ao processador tanto a leitura quanto a gravação de dados, mas perde sua informação quando não há alimentação elétrica, ou seja, é uma memória volátil.

As memórias do tipo ROM são muito utilizadas nos Sistemas Computacionais para armazenar o código (ou programa) que será executado pelo processador, enquanto as memórias RAM armazenam os dados e/ou informações que são manipulados pelo processador durante a execução do código. Além disso, as memórias RAM têm um processo de gravação de dados extremamente rápido, se comparadas aos vários tipos de memória ROM.

Armazenam ou acessam as informações digitais em lugares denominados localidades ou posições, mediante um endereçamento. Para o acesso a essas posições, possuem uma série de terminais de entradas de endereços que são ligados a um conjunto de fios denominado **barramento de endereços** (*address bus*). Para a entrada e saída dos dados possuem uma série de terminais (bidirecionais) ligados ao **barramento de dados** (*data bus*). Possuem ainda sinais de controle que indicam a operação desejada (leitura, escrita, habilitação).



2.8.1. Tipos de memória

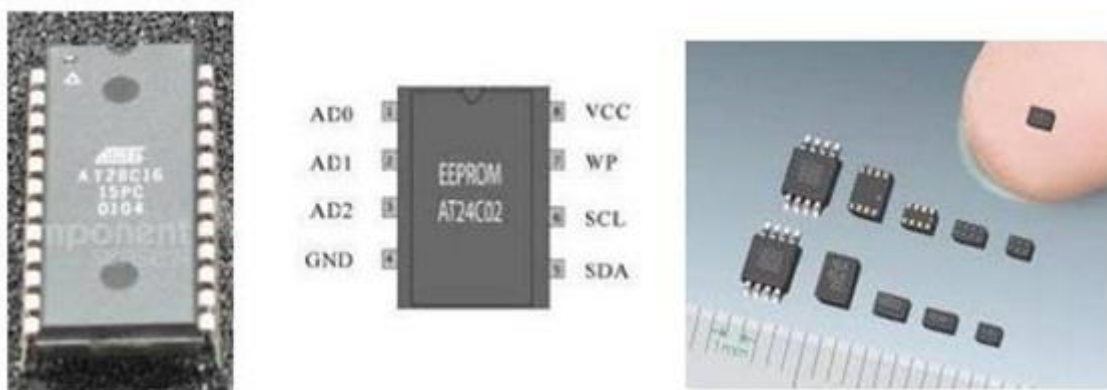
2.8.1.1. Tipos de memória ROM

- **ROM (*Read-Only Memory* ou Memória Somente de Leitura)**: recebem esse nome porque os dados são gravados nelas apenas uma vez, ainda no processo de fabricação da memória. Depois disso, essas informações não podem ser apagadas ou alteradas, apenas lidas pelo processador.

- **PROM (Programmable Read-Only Memory ou Memória Somente de Leitura Programável):** Esse tipo de memória vem de fábrica “vazia” e a gravação dos dados pode ser realizada por meio de equipamentos que geram uma sinalização apropriada para este fim. É como se ocorresse a “queima de fusíveis” dentro da memória para armazenar a informação desejada. Dessa forma, uma vez programadas, não é possível mais alterar o seu conteúdo, que agora só poderá ser lido pelo processador.
- **EPROM (Erasable Programmable Read-Only Memory ou Memória Somente de Leitura Programável e Apagável):** as memórias EPROM funcionam basicamente como as PROM, mas têm como principal característica a capacidade de permitir que seus dados sejam apagados e depois regravados no dispositivo. Para isso, é necessário um equipamento que emita uma luz ultravioleta sobre o dispositivo. Diferentemente das outras memórias, que possuem encapsulamento plástico, esse tipo de memória possui um encapsulamento cerâmico e uma “janela” no encapsulamento vedada com um cristal de quartzo, que permite a passagem da luz ultravioleta diretamente sobre o chip interno à memória. Inicialmente, por ser mais cara, por causa do encapsulamento cerâmico e do cristal de quartzo, esse tipo de memória era utilizada durante a fase de desenvolvimento do projeto, no qual o código do sistema precisava ser constantemente testado e regravado. Uma vez concluído o projeto, já com o código a ser gravado na memória todo testado, o processo de fabricação do sistema utilizava as memórias PROM, mais baratas.



- **EEPROM (Electrically Erasable Programmable Read-Only Memory ou Memória Somente de Leitura Programável e Apagável Eletricamente):** este tipo de memória ROM também permite a regravação de dados, no entanto, ao contrário do que acontece com as memórias EPROM, os processos para apagar e gravar dados são feitos eletricamente, fazendo com que não seja necessário remover o dispositivo de seu lugar para um aparelho especial para apagar o conteúdo da memória e depois regravar os dados por meio de outro equipamento. Existem dispositivos com alta capacidade de armazenamento, mas também dispositivos que armazenam apenas poucos bytes. O segundo tipo é muito utilizado nos Sistemas Computacionais para armazenar pequeno volume de dados, em geral, parâmetros de configuração de um sistema, que devem permanecer gravados mesmo que o equipamento perca a sua alimentação.



- **Flash:** as memórias Flash também podem ser vistas como um tipo de EEPROM, no entanto, o processo de gravação (e regravação) é muito mais rápido. Além disso, memórias Flash são mais duráveis e podem guardar um volume elevado de dados. É o tipo de ROM atualmente mais utilizada, tendo como aplicações os cartões de memória de câmeras e celulares, os pen-drives, para armazenar o BIOS dos computadores pessoais, os discos rígidos do tipo SSD, entre outras.

2.8.1.2. Tipos de memória RAM

Há dois tipos de tecnologia de memória RAM que são muito utilizados: estático e dinâmico, isto é, SRAM (*Static RAM*) e DRAM (*Dynamic RAM*), respectivamente

- **SRAM (Static Random-Access Memory):** sua célula básica de armazenamento da informação (bit) é um flip-flop. São muito mais rápidas que as memórias DRAM, porém, possuem uma densidade bem menor, ou seja, armazenam menos dados numa mesma área de silício, além de possuírem custo elevado se considerarmos o valor por megabyte. Memórias SRAM costumam ser utilizadas dentro dos processadores (registradores internos) e como cache, onde a quantidade de memória é pequena, mas se necessita de uma alta velocidade de comunicação.
- **DRAM (Dynamic Random-Access Memory):** sua célula básica é formada por um capacitor e um transistor. O capacitor armazena a informação e o transistor controla o fluxo dessa informação. Pelo fato de sua célula de armazenamento ser bem menor que a de uma memória estática, possui uma alta densidade, ou seja, pode comportar uma quantidade muito maior de informação em uma mesma área de silício. No entanto, o acesso a essas informações é mais lento que o acesso às memórias estáticas. Por causa da sua elevada densidade, a DRAM possui um preço bem menor quando comparado ao tipo estático, assim, são muito utilizadas como memória principal nos computadores pessoais, onde a capacidade da memória (volume de dados) é muito mais importante que a velocidade de acesso à memória.

2.8.1.2.1. Tipos de memória DRAM

Várias tecnologias de memórias são criadas com o passar do tempo, basicamente com o objetivo de aumentar a capacidade de armazenamento e também a velocidade de acesso à memória e, quando possível, diminuir o seu consumo de energia.

- **FPM (*Fast-Page Mode*):** com o FPM, a primeira leitura acaba sendo mais demorada, mas as três seguintes são mais rápidas. Isso porque são feitos, na verdade, quatro operações de leitura seguidas, ao invés de apenas uma, em um esquema do tipo x-y-y-y. Com esse esquema, o controlador de memória trabalha apenas uma vez com o endereço de uma linha (RAS) e, em seguida, trabalha com uma sequência de quatro colunas (CAS), ao invés de trabalhar com um sinal de RAS e um de CAS para cada bit. Memórias FPM utilizavam módulos SIMM, tanto de 30 quanto de 72 vias;
- **EDO (*Extended Data Output*):** a sucessora da tecnologia FPM é a EDO, que possui como destaque a capacidade de permitir que um endereço da memória seja acessado ao mesmo tempo em que uma solicitação anterior ainda está em andamento. Esse tipo foi aplicado principalmente em módulos SIMM, mas também chegou a ser encontrado em módulos DIMM de 168 vias. Houve também uma tecnologia semelhante, chamada BEDO (Burst EDO), que trabalhava mais rapidamente por ter tempo de acesso menor, mas quase não foi utilizada, pois tinha custo maior por ser de propriedade da empresa Micron. Além disso, foi "ofuscada" pela chegada da tecnologia SDRAM;
- **SDRAM (*Synchronous Dynamic Random Access Memory*):** as memórias FPM e EDO são assíncronas, o que significa que não trabalham de forma sincronizada com o processador. O problema é que, com processadores cada vez mais rápidos, isso começou a se tornar um problema, pois muitas vezes o processador tinha que esperar demais para ter acesso aos dados da memória. As memórias SDRAM, por sua vez, trabalham de forma sincronizada com o processador, evitando os problemas de atraso. A partir dessa tecnologia, passou-se a considerar a frequência com a qual a memória trabalha para medida de velocidade. Surgiram então as memórias SDR SDRAM (*Single Data Rate SDRAM*), que podiam trabalhar com 66 MHz, 100 MHz e 133 MHz (também chamadas de PC66, PC100 e PC133, respectivamente).

Muitas pessoas se referem a essa memória apenas como "memórias SDRAM" ou, ainda, como "memórias DIMM", por causa de seu módulo. No entanto, a denominação SDR é a mais adequada;

- **DDR SDRAM (*Double Data Rate SDRAM*):** as memórias DDR apresentam evolução significativa em relação ao padrão SDR, isso porque elas são capazes de lidar com o dobro de dados em cada ciclo de *clock* (memórias SDR trabalham apenas com uma operação por ciclo). Assim, uma memória DDR que trabalha à frequência de 100 MHz, por exemplo, acaba dobrando seu desempenho, como se trabalhasse à taxa de 200 MHz. Visualmente, é possível identificá-las facilmente em relação aos módulos SDR, porque este último contém duas divisões na parte inferior, onde estão seus contatos, enquanto que as memórias DDR2 possuem apenas uma divisão.
- **DDR2 SDRAM:** como o nome indica, as memórias DDR2 são uma evolução das memórias DDR. Sua principal característica é a capacidade de trabalhar com quatro operações por ciclo de *clock*, portanto, o dobro do padrão anterior. Os módulos DDR2 também contam com apenas uma divisão em sua parte inferior, no entanto, essa abertura é um pouco mais deslocada para o lado.
- **DDR3 SDRAM:** as memórias DDR3 são, obviamente, uma evolução das memórias DDR2. Novamente, aqui dobra-se a quantidade de operações por ciclo de *clock*, desta vez, com oito operações por ciclo.
- **Rambus (Rambus DRAM):** as memórias Rambus recebem esse nome por serem uma criação da empresa Rambus Inc. e chegaram ao mercado com o apoio da Intel. Elas são diferentes do padrão SDRAM, pois trabalham apenas com 16 bits por vez. Em compensação, memórias Rambus trabalham com frequência de 400 MHz e com duas operações por ciclo de *clock*. Tinham como desvantagens, no entanto, taxas de latência muito altas, aquecimento elevado e maior custo. Memórias Rambus nunca tiveram grande aceitação no mercado, mas também não foram um total fiasco: foram utilizadas, por exemplo, no console de jogos Nintendo 64. Curiosamente, as memórias Rambus trabalham em pares com "módulos vazios" ou "pentes cegos". Isso significa que, para cada módulo Rambus instalado, um "módulo vazio" tem que ser instalado em outro slot. Essa tecnologia acabou perdendo espaço para as memórias DDR.

2.8.2. Capacidade de uma memória

É muito importante especificar quantos bits podem ser armazenados em uma determinada memória. Tomemos, como exemplo, uma memória que possa armazenar 4K palavras de 20 bits. Isto representa uma capacidade total de armazenamento de 81.920 bits (4.096×20), onde 4.096 é o número de palavras ($4K = 4 \times 1K = 4 \times 1.024$) e 20 é a quantidade de bits por palavra.

$$\begin{array}{ccc} & \text{Quantidade de} & \\ & \text{bits por palavra} & \\ & \downarrow & \\ \text{4.096} & \times & 20 = 81.920 \\ \uparrow & & \uparrow \\ \text{Número de} & & \text{Quantidade total} \\ \text{palavras} & & \text{de bits} \end{array}$$

Utiliza-se comumente representar o número de palavras da memória como múltiplo de 1.024, sendo comum a designação 1K (1 kilo) para representar 1.024 bits, que é igual a 2^{10} . Por exemplo, uma memória que tenha uma capacidade de armazenamento de 8K x 20 é na verdade uma memória de 8.192×20 . Memórias de grande capacidade de armazenamento utilizam a designação 1M (1 mega), que representa 2^{20} (que é igual a 1.048.576 bits), e 1G (1 giga), que representa 2^{30} (que é igual a 1.073.741.824 bits). Dessa forma uma memória com capacidade de $2M \times 16$, possui uma capacidade de armazenar $2.097.152 \times 16 = 33.554.432$ bits, e uma memória

de 4G x 8, possui uma capacidade de armazenar $4.294.967.296 \times 8 = 34.359.738.368$ bits.

- Vejamos alguns exemplos:

1) Um chip de memória é especificado como tendo a capacidade de 4K x 8. Quantas palavras podem ser armazenadas nesse chip? Qual é o tamanho da palavra? Quantos bits no total esse chip pode armazenar?

Solução:

$4K = 4 \times 1.024 = 4.096$ palavras

Cada palavra tem 8 bits ou 1 byte

O número total de bits é 32.768 (4.096×8)

2) Qual das memórias armazena mais bits: 2M x 8 ou 1M x 16?

Solução:

a) $2 \times 1.048.576 \times 8 = 16.777.216$

b) $1 \times 1.048.576 \times 16 = 16.777.216$

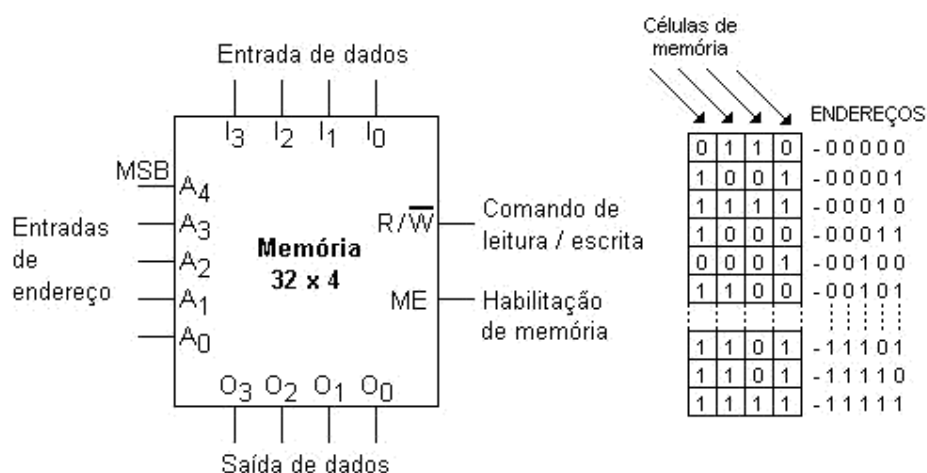
Portanto, as duas memórias têm a mesma capacidade de armazenamento de bits.

2.8.3. Operações básicas da memória

Apesar dos diversos tipos de implementação, as memórias, de uma forma geral, apresentam algumas operações básicas, que são:

- selecionar o endereço da posição que será acessada na memória;
- selecionar a operação a ser realizada, leitura ou escrita, e habilitar a memória com a qual ocorrerá a troca de dados (essa habilitação é feita por uma lógica de seleção desenvolvida a partir dos bits mais significativos do barramento de endereços);
- no caso de uma operação de escrita, fornecer os dados de entrada;
- manter estáveis as informações para a memória durante um tempo determinado, suficiente para a leitura do dado endereçado (operação de leitura) ou para que a memória armazene o dado (operação de escrita);
- desabilitar a memória.

A figura abaixo mostra, como exemplo, o diagrama simplificado de uma memória 32 x 4 (32 palavras de 4 bits), isto é, cada palavra tem o tamanho de 4 bits.



Como o tamanho da palavra é de 4 bits, essa memória deve possuir 4 sinais ou linhas de dados (barramento de dados). Normalmente, o barramento de dados é bidirecional, ou seja, os sinais de dados são utilizados tanto para fornecer um dado para a memória, quanto para ler um dado da memória. Nesse exemplo, para facilitar a compreensão, dividiremos o barramento de dados em 2, um de entrada e um de saída. Assim, existe nesta memória 4 linhas de entrada de dados e 4 linhas de saída de dados. Durante a operação de escrita, os dados a serem armazenados na

memória devem ser colocados nas linhas de entrada de dados (I0 a I3) e durante a operação de leitura, a palavra lida aparece nas linhas de saída de dados (O0 a O3).

A memória mostrada no exemplo possui 32 posições diferentes de armazenamento e, portanto, 32 endereços diferentes, começando por 00000 e terminando em 11111 (0 a 31 decimal). Dessa forma, essa memória deve ter 5 entradas de endereçamento ($2^5 = 32$). São necessárias N linhas de entrada de endereço (barramento de endereço) para uma memória com capacidade de 2^N palavras.

Observando a figura anterior, percebe-se que cada posição (ou endereço) possui 4 células de memória, onde cada célula armazena um bit (0 ou 1), formando assim uma palavra de 4 bits por endereço. Dessa forma, no endereço 00001 está armazenada a palavra 1001; e no endereço 11110 está armazenada a palavra 1101.

O comando de leitura/escrita determina qual das operações a memória deverá executar. Alguns sistemas utilizam linhas separadas para leitura e escrita. Quando se usa uma única linha temos as seguintes condições, onde R é a abreviação de *Read* (Leitura) e W é a abreviação de *Write* (Escrita):

$$\text{LEITURA: } R / \overline{W} = 1$$

$$\text{ESCRITA: } R / \overline{W} = 0$$

As memórias possuem ainda um sinal que desabilita sua operação, ou seja, ela não responderá a nenhum tipo de operação (leitura ou escrita), ignorando o endereço a ela fornecido. Na figura acima, a memória possui um sinal de entrada, ME, que permite a habilitação e desabilitação, que é ativo em nível alto (um sinal será ativo em nível baixo quando possuir uma barra horizontal, de negação, sobre seu nome). Esse sinal é usado quando diversos módulos de memória são combinados para formar uma memória maior. Embora ele tenha sido chamado de ME (*Memory Enable*) na figura de exemplo, nos circuitos integrados comerciais de memória, o nome desse sinal é CS (*Chip Select*) ou CE (*Chip Enable*) e é normalmente ativo em nível baixo.

2.8.4. Conexão da memória a um processador

Da mesma forma que a memória, o processador possui um barramento de endereços, que indica a quantidade de posições de memória que ele é capaz de ler e/ou escrever; um barramento de dados, que indica o tamanho dos seus registradores internos ou o tamanho da palavra que ele é capaz de manipular por vez; e os sinais de controle.

Se o barramento de endereços da memória for igual ao do processador, bem como o barramento de dados, basta ligar diretamente os sinais do processador na memória. Entretanto, regra geral, a capacidade de endereçamento do processador é maior do que a capacidade de armazenamento da memória, o que faz com que sejam necessários vários CIs de memória para conseguir suprir a capacidade de endereçamento do processador.

Nessa situação, as linhas de endereço correspondentes da memória e do processador são ligadas diretamente, e **as linhas de endereço que sobram do processador são utilizadas para criar uma lógica de seleção atuando no sinal CS das memórias**, que indica com qual memória o processador quer se comunicar.

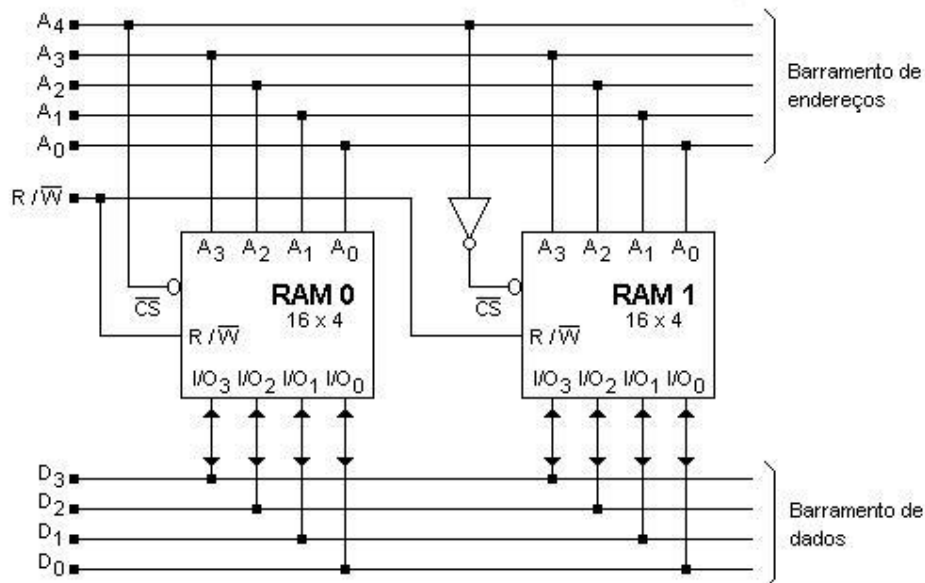
Como exemplo, podemos citar a necessidade de um processador com uma capacidade de endereçar 32 posições e com um barramento de dados de 4 bits (32 palavras de 4 bits ou 32×4). Para criar a memória desse sistema, dispomos de circuitos integrados de 16×4 . A figura a seguir mostra como ligar esses circuitos integrados das memórias nos barramentos de endereço e dados do processador. Note que o bit de endereço que “sobrou” do processador foi utilizado para criar uma lógica de seleção da memória.

O arranjo ilustrado acima abrange os seguintes endereços:

RAM 0 – Endereços de 00000 a 01111 (16 posições)

RAM 1 – Endereços de 10000 a 11111 (16 posições)

TOTAL - 00000 a 11111 (32 palavras = $2^5 \rightarrow 5$ linhas de endereços)



Observa-se que os sinais de endereço A0 a A3 da RAM foram ligados diretamente nos sinais de endereço A0 a A3 do processador. Dessa forma, a faixa de endereços para as RAMs 0 e 1 ficará por conta das entradas A3 A2 A1 A0, que irão fornecer os 4 bits necessários para endereçar as 16 posições de memória de cada RAM. Já o sinal de endereço restante, A4, foi utilizado para selecionar uma das duas RAMs através das suas entradas CS':

- quando $A_4 = 0 \rightarrow CS_0' = 0$ e $CS_1' = 1$, o que habilita apenas a RAM 0, desabilitando a RAM 1;
- quando $A_4 = 1 \rightarrow CS_0' = 1$ e $CS_1' = 0$, o que habilita apenas a RAM 1, desabilitando a RAM 0.

2.9. Reset

Um sistema microprocessado, ao ser ligado, leva um tempo para que todos os seus componentes possam estar em condições de operação, incluindo o circuito oscilador, que gera o sinal que sincroniza todas as suas operações. Dessa forma, o microprocessador deve aguardar um tempo (para que tudo estabilize) antes de começar a executar suas funções. O circuito responsável por manter o processador "parado", impedindo que ele opere em condições não confiáveis é chamado circuito de *reset*. Além de manter o processador em *stand-by* (estado de espera), ele também põe o processador em **uma condição conhecida**, colocando um valor inicial em vários registradores, entre eles o PC e o SP. O *reset* também pode ser utilizado se o processador estiver "travado", ou seja, operando em condições desconhecidas, permitindo que ele recupere desse estado e volte a operar normalmente.

- **Power-on Reset (POR):** é aquele que ocorre ao se alimentar o circuito. Para ser efetivo deve ocorrer depois que a alimentação e o oscilador estiverem estáveis.
- **Reset manual:** é o *reset* que ocorre com o circuito "a quente", ou seja, com alimentação presente. Em geral, é feito por uma chave colocada no circuito que, quando pressionada, muda o nível lógico da entrada de *reset* do processador.

3. Linguagem Assembly

Para executar uma tarefa qualquer, um microprocessador precisa receber as instruções precisas sobre o que fazer. Uma sequência adequada de instruções para realizar uma tarefa se constitui em um programa de microprocessador e é elaborado em uma linguagem de programação. Uma linguagem de programação é um conjunto de ferramentas, regras de sintaxe e símbolos ou códigos que nos permitem escrever programas.

A **linguagem de máquina** é aquela que mais se aproxima das características técnicas do hardware. É a linguagem própria do microprocessador, definida pelo fabricante, confusa para o ser humano, e caracterizada por códigos pequenos e rápidos chamados de *opcodes* (**operation codes** ou códigos da operação). Sua representação é feita por números binários e/ou hexadecimais, o que torna a programação nessa linguagem pouca prática.

Um programa em linguagem de máquina é uma longa série de 0's e 1's (ou em hexadecimal), ordenados de forma que alguns representem códigos de instruções e os outros representem os dados que serão processados ou indicam onde esses dados estão armazenados.

Já a **linguagem assembly**, também chamada de **linguagem simbólica**, foi a primeira evolução das linguagens de programação. Essa linguagem é constituída por um conjunto de instruções propostas em símbolos e números, onde os símbolos representam os mnemônicos e registradores e os números representam os dados e endereços. Como exemplo de um mnemônico podemos citar a instrução **ADD A,#35H** que representa a soma de dois dados. Essa mesma instrução escrita em linguagem de máquina de um processador hipotético seria 2435H (em hexadecimal) ou 0010010000110101 (em binário).

A linguagem *assembly* proporciona uma maior clareza, inexistente na linguagem de máquina, facilitando bastante a programação. É importante salientar que um microprocessador entende única e exclusivamente a sua linguagem de máquina.

Portanto, para que um programa escrito em *assembly* (ou outra linguagem diferente da de máquina) possa ser entendido e executado por um processador, é preciso haver um outro programa que leia o código escrito nessa linguagem alternativa e o traduza para a linguagem de máquina do processador em questão.

O programa chamado de **assembler** (montador) é quem realiza o processo de tradução da linguagem *assembly* para a linguagem de máquina específica de um processador. A criação dos programas montadores facilitou muito o trabalho dos programadores, além de permitir uma melhor depuração de erros de sintaxe e de programação.

O programa em *assembly* é chamado de **programa fonte**, e o programa em linguagem de máquina é chamado de **programa objeto**. A tabela abaixo nos mostra um exemplo bem simples de um trecho de programa em linguagem *assembly* e o mesmo escrito em linguagem de máquina.

Programa em Linguagem Assembly			Programa em Linguagem de Máquina		
End. na memória de programa	Mnemônico	Operando	End. na memória de programa	Código da operação (<i>opcode</i>)	Dado ou endereço
DESVIO:	CLR	A	0125H	3F	
	ADD	A,#05H	0126H	E7	05
	OUT	20H	0128H	F1	20
	JMP	TESTE	012AH	87	012F

➤ Vantagem dos montadores:

- Permitem atribuir nomes a localizações de memória, dispositivos de entrada e saída, e mesmo seqüências de instruções. No exemplo da tabela anterior, TESTE é um nome atribuído à localização de memória 012FH;
- Convertem dados e endereços de diversos sistemas de numeração (decimal, octal, hexadecimal) para binário e vice-versa, e ainda caracteres para seus correspondentes códigos, por exemplo, ASCII;
- Executam algumas operações aritméticas como parte do processo *assembly*, como na instrução ADD A, C+1, na qual o acumulador é somado com o conteúdo do registrador C mais 1.

➤ Desvantagens da linguagem *assembly*:

- Não é portátil, ou seja, um programa escrito para um determinado microprocessador não pode ser executado em outro microprocessador de outra família;
- Deve-se conhecer bastante o microprocessador que está sendo, para que o programa em *assembly* se possa ser escrito da maneira mais otimizada possível;
- Existe uma distância grande entre as instruções do microprocessador e as tarefas que o sistema microprocessado deve executar, ou seja, o programador deve raciocinar bem “próximo” do *hardware* do sistema.

As **linguagens de baixo nível** são compostas por símbolos e números, ou seja, códigos próximos aos interpretados pela máquina. Entre elas estão as linguagens de máquina e *assembly*. Suas vantagens são: código pequeno e alta velocidade de processamento. Suas desvantagens são: difícil aprendizado, difícil manutenção e depuração.

As **linguagens de alto nível** caracterizam-se por símbolos próximos à linguagem natural do homem, por exemplo, Pascal, C, Basic. Suas vantagens são: fácil aprendizado, modularização,

depuração e gerência automática da memória. Suas desvantagens são: necessidade de tradução para linguagem de máquina correspondente, código mais extenso quando traduzido do que um feito em uma linguagem de baixo nível.

Cada instrução de um programa em linguagem de alto nível corresponde a inúmeras instruções do microprocessador. Os compiladores são os programas que traduzem o programa fonte em linguagem de alto nível para linguagem de máquina, o programa objeto.

3.1. Tipos de instruções:

Embora o conjunto de instruções varie de processador para processador, essas instruções podem ser divididas em três grupos básicos:

- Lógicas e Aritméticas: utilizam a ULA (AND, OR, XOR, SHIFT, SWAP, ADD, SUB, etc.);
- Movimentação de dados: acesso à memória e registradores;
- Desvio (incondicional e condicional): alteração do PC.

3.2. Sub-rotinas:

O fluxo normal do programa é interrompido por uma instrução (CALL) e desviado para uma rotina que irá realizar uma tarefa determinada (chamada a uma sub-rotina). Ao encerrar a tarefa, o programa retorna ao ponto de onde foi desviado. Como essa rotina pode ser chamada de diversos pontos do programa principal, é necessário armazenar o endereço para onde o fluxo deve retornar, de acordo com o local da chamada. O endereço de retorno é armazenado na pilha do sistema.

3.3. Interrupções:

Semelhantemente a uma sub-rotina, o fluxo normal do programa é interrompido e desviado para uma rotina que irá realizar uma tarefa determinada (atendimento da interrupção), retornando depois ao ponto de onde o programa foi desviado. A diferença é que não se sabe o instante no qual esse desvio ocorrerá, sendo imprevisível. O controle do desvio é feito por hardware e não mais por software. Também é necessário que o endereço de retorno seja armazenado na pilha.

4. Microcontroladores

Com a evolução da microeletrônica e conseqüente barateamento dos circuitos integrados, além do surgimento de microprocessadores (μ P) mais poderosos, começou-se a utilizar os microprocessadores mais simples para implementar tarefas dedicadas, tais como controle de impressora, plotter, reguladores de velocidade, acionadores de motores de passo, controladores de elevadores, etc.

A utilização de um μ P para controle implica em um circuito relativamente grande, que muitas vezes encarece o custo do controlador. Conforme foi visto, um sistema microprocessado necessita de um série de circuito auxiliares (oscilador, memórias, portas de E/S, temporizadores, etc.) para poder operar. Além da quantidade de componentes, isso implica também em uma maior quantidade de conexões e tamanho da placa do sistema.

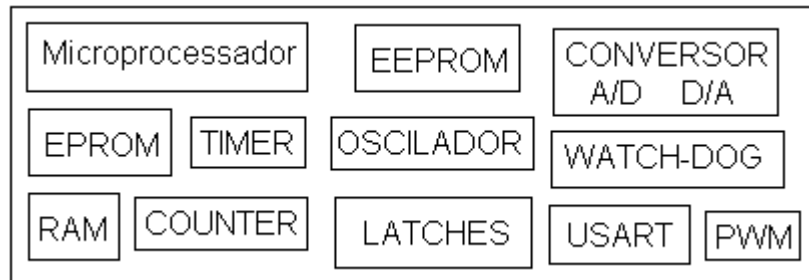
Regra geral, o μ P dedicado a um determinado controle não precisa ser muito rápido nem tampouco ter um conjunto de instruções extenso e poderoso. Dessa forma, surgiu a idéia de colocar os diversos componentes de um sistema microprocessado em um só *chip* com uma CPU simples. Isso baratearia e diminuiria o tamanho do circuito impresso além de aumentar a confiabilidade.

Assim, em 1976, lançados pela INTEL, surgiram os **microcontroladores**, que são dispositivos com as seguintes características:

- Microcomputador de um só *chip*. Possuem, dentro de um mesmo circuito integrado, ou seja, um único componente, CPU, memórias ROM e RAM, temporizadores, oscilador, portas de E/S, etc.;
- Simples;
- Baratos;
- Eficientes para as aplicações a que se destinam.

Os recursos incorporados vêm aumentando com a evolução da integração e, com isso, projetos cada vez mais complexos podem ser resolvidos com bastante facilidade.

4.1. Estrutura Interna de um Microcontrolador



- Microprocessador (CPU)
- Memória de programa, em geral, uma PROM, EPROM, EEPROM ou FLASH
- Memória de dados (RAM)
- Memória para configurações (EEPROM ou FLASH)
- Temporizadores (Timers) / Contadores
- Latches bidirecionais (portas de entrada e saída digitais)
- Interface serial
- Conversores A/D e D/A
- Watch-Dog
- Oscilador
- PWM

4.2. Microcontrolador x Microprocessador

- Baixa capacidade de armazenamento do sistema
- Baixo desempenho, porém adequada à aplicação a que se destina
- Existência de inúmeros periféricos (temporizador, conversores A/D e D/A, interface serial, portas de E/S, etc.)
- Baixo custo
- Pouca expansibilidade
- Pouca flexibilidade
- Pequeno espaço físico

4.3. Aplicações

Os microcontroladores são utilizados em aplicações relativamente simples, que necessitam de processamento da informação com pequena capacidade de armazenamento, devam ter um tamanho reduzido e precisam de um baixo custo. Essas aplicações que utilizam os microcontroladores são conhecidas como sistemas embarcados (*embedded systems*).

Dentre as aplicações dos microcontroladores, podemos citar: computadores de bordo, eletrodomésticos, sistemas de alarme, sistemas de telefonia (celulares, centrais PABX, bloqueados, secretárias eletrônicas), estabilizadores e no-breaks, pequenos robôs, etc.

4.4. Como escolher um microcontrolador

Atualmente, existe uma grande variedade de microcontroladores, de diversos fabricantes, e com capacidades de processamento e quantidade de periféricos internos das mais diversas. Em grande parte dos sistemas, diversos microcontroladores poderiam ser escolhidos para implementação do sistema. Dessa forma, ao selecionar um microcontrolador para um determinado projeto, as seguintes regras podem ser levadas em consideração:

- Características do microcontrolador que mais se adequam ao projeto
- Facilidade de obtenção
- Custo
- Ferramentas de desenvolvimento
- Desempenho
- Familiaridade

4.5. Principais fabricantes

- Microchip (PICs): www.microchip.com
- Atmel (8051 e AVR): www.atmel.com
- Motorola (MC68HC05): www.motorola.com
- Philips (8051): www.philips.com
- National (COP): www.national.com
- ST (STx): www.st.com
- Holtek (HTxxxx): www.holtek.com
- Texas (TMS e MSP43xx): www.ti.com