Capítulo 4

Conceitos de eletrônica digital

Alguma vez você já se perguntou: O que há dentro de circuitos eletrônicos integrados digitais como microcontroladores e microprocessadores? O que faz estes circuitos realizarem operações matemáticas complexas e tomar decisões? Sabia que essa aparente complexidade inclui apenas alguns elementos chamados circuitos lógicos ou portas lógicas?

4.1 Elementos de eletrônica digital

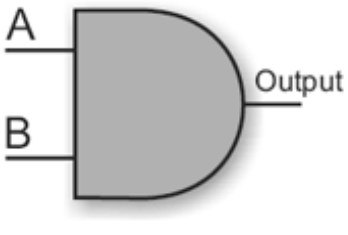
A operação dos circuitos lógicos é baseada em princípios estabelecidos pelo matemático britânico George Boole em meados do século XIX, antes mesmo da primeira lâmpada ter sido inventada.

Originalmente, a ideia principal era expressar formulações lógicas através de funções algébricas. Tal pensamento foi logo transformado em um produto prático, que hoje é conhecido como circuitos lógicos E (AND), OU (OR) e NÃO (NOT), ou inversor. O princípio de operação desses circuitos é conhecido como álgebra booleana.

Algumas instruções encontradas nos programas escritos para microcontroladores dão o mesmo resultado que as portas lógicas e, por isso, a operação desses elementos será discutida em seguida.

Porta lógica E (AND)

A porta lógica "E" tem duas ou mais entradas digitais e uma saída digital. Suponha que a porta usada neste exemplo tem apenas duas entradas. A lógica um (1) irá aparecer em sua saída apenas se as duas entradas (A e B) são acionadas com nível lógico alto (1). A tabela apresentada na figura 1, chamada de tabela da verdade, mostra a dependência mútua entre as entradas e a saída.



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Figura 1 – Porta lógica E e sua tabela da verdade.

Quando usada em um programa em um microcontrolador, uma operação lógica E é realizada por uma instrução da linguagem de programação. Por ora, basta lembrar que a lógica E num programa refere-se à operação lógica AND aplicada aos bits correspondentes a dois números que estão armazenados em posições de memória chamadas de registradores, conforme se pode ver na figura 2. Nessa figura, há dois registradores de 8 bits, A e B, com os valores 0b11001001 e 0b01101000, respectivamente, fornecendo como resultado o valor 0b01001000.

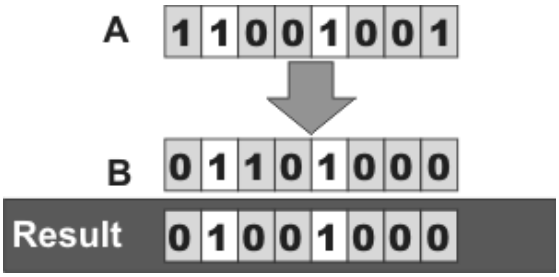
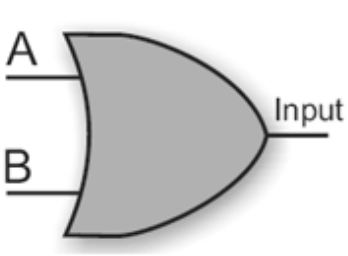


Figura 2 – Operação lógica AND sendo aplicada sobre dois registradores, A e B.

Porta lógica OU (OR)

Da mesma forma, a porta lógica "OU" também tem duas ou mais entradas e uma saída. Se a porta tem apenas duas entradas, uma lógica um (1) irá aparecer em sua saída se uma entrada (A ou B) é acionada com nível lógico alto (1). Ou seja, uma lógica um (1) aparece em sua saída, se pelo menos uma entrada é mantida alta (1). Se todas as entradas estão à lógica zero (0), a saída também apresenta lógica zero (0).

Na figura 3 pode ser vista a tabela da verdade para a porta lógica OU, assim como uma operação OU realizada com os valores de dois registradores, A e B.



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

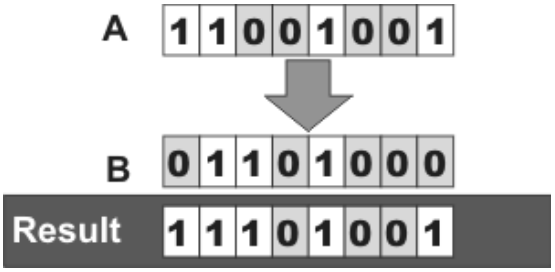


Figura 3 – Porta lógica OU, sua tabela da verdade e operação lógica sendo aplicada sobre dois registradores, A e B.

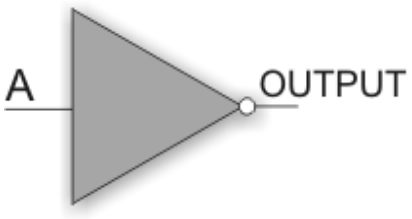
Porta lógica NÃO (NOT)

A porta lógica "NÃO" (NOT) tem apenas uma entrada e uma única saída. Ela opera de maneira extremamente simples: Quando a lógica zero (0) aparece em sua entrada, uma lógica um (1) aparece em sua saída, e vice-versa. Isso significa que esta porta inverte o sinal e, por isso, é frequentemente chamada de porta inversora.

Em um programa, essa operação lógica é feita sobre o valor armazenado em um registrador, ou apenas um bit. O resultado é o valor com os bits invertidos.

Se o valor que está sendo operado é considerado como um número, o valor invertido, na verdade, é um complemento do mesmo. O complemento de um número binário de *n* bits é um valor que adicionado ao número original resulta no maior número possível de ser representado pelos *n* bits. Por exemplo, a soma de um número de 8 bits e seu complemento é sempre 255, que é o maior valor capaz de ser representado utilizando 8 bits.

Na figura 4 pode ser vista a tabela da verdade para a porta lógica NÃO, assim como uma operação NÃO realizada com o valor de um registrador.



A	Output
0	1
1	0

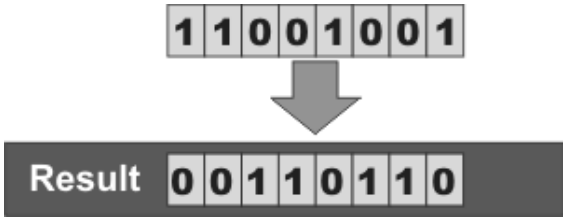


Figura 4 – Porta lógica NÃO, sua tabela da verdade e operação lógica sendo aplicada sobre um registrador.

Porta lógica OU-EXCLUSIVO (XOR)

A porta "OU EXCLUSIVO" (XOR) é um pouco mais complicada para se entender em comparação com as outras portas. A lógica um (1) aparece em sua saída somente quando suas entradas têm estados lógicos diferentes.

Em um programa, esta operação é comumente usada para comparar dois bytes. A subtração pode ser utilizada para o mesmo fim (se o resultado for 0, os bytes são iguais). Ao contrário da subtração, a vantagem desta operação lógica é que não é possível a obtenção de resultados negativos.

Na figura 5 pode ser vista a tabela da verdade para a porta lógica OU EXCLUSIVO, assim como uma operação realizada com os valores de dois registradores.

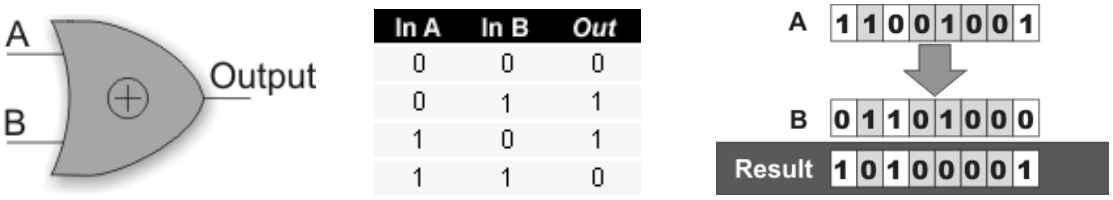


Figura 5 – Porta lógica XOR, sua tabela da verdade e operação lógica sendo aplicada sobre dois registradores, A e B.

Registradores

Em suma, um registrador, ou uma célula de memória, é um circuito eletrônico que pode memorizar o estado de um byte (ou de *n* bits). Dentro de um microcontrolador há vários registradores que são usados para armazenar as informações do programa. Essas informações podem ser variáveis de controle do programa, estado das entradas ou saídas, resultado de uma operação matemática, etc.

Graficamente, podemos ver como funciona um registrador através da figura 6. Nessa figura, a unidade central de processamento (CPU) pode armazenar um byte numa posição de memória, ou registrador. O valor armazenado nesse exemplo é o 0b10110011. Para armazenar esse valor, a CPU envia os bits para o registrador através de 8 linhas de dados e utiliza um sinal de escrita para que o registrador retenha a informação. Esse valor armazenado pode, posteriormente, ser utilizado pela própria CPU para um novo cálculo ou processamento.

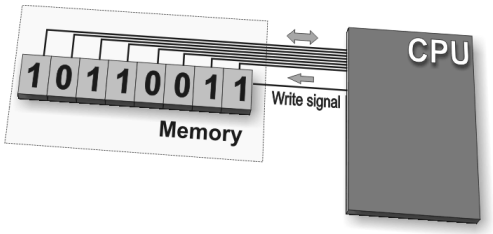


Figura 6 – CPU comunicando-se com um registrador e armazenando o valor 0b10110011.

Além de registradores de uso geral, como o mostrado na figura 6, cada microcontrolador também possui uma quantidade de registradores especiais (Special Function Registers-SFR), cuja função é pré-determinada pelo fabricante. Seus bits são conectados (literalmente) com os circuitos internos do microcontrolador, tais como temporizadores, conversores A/D, osciladores, entre outros, o que significa que eles são usados diretamente no controle da operação destes circuitos periféricos. O estado dos bits dos registradores especiais é alterado a partir do programa e, com isso, permite o controle dos pequenos circuitos dentro do microcontrolador, como ilustrado na figura 7.

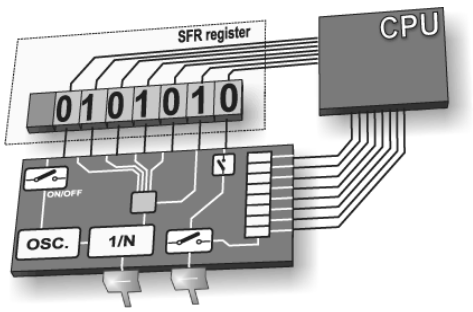


Figura 7 – CPU comunicando-se com um registrador de função especial.

Portas de entrada/saída (portas de I/O)

A fim de tornar o microcontrolador útil, este tem que ser ligado a uma eletrônica adicional externa ao chip. Cada microcontrolador tem um ou mais registradores de função especial (chamados de portas) cujos bits estão diretamente conectados aos pinos do chip.

Por que entrada/saída? Porque pode-se mudar a função do pino que se desejar. Por exemplo, suponha que se deseja que um dispositivo ligue/desligue três LEDs de sinal e, simultaneamente, monitore o estado lógico de cinco sensores, ou teclas. Algumas das portas precisam ser configuradas de modo que hajam três saídas (ligadas aos LEDs) e cinco entradas (ligadas aos sensores). Esta configuração é simplesmente realizada por software, ou seja, através do programa que o projetista escreve, o que significa que a função do pino pode ser alterada durante a operação.

Podemos ilustrar o funcionamento de uma porta de um microcontrolador através da figura 8.

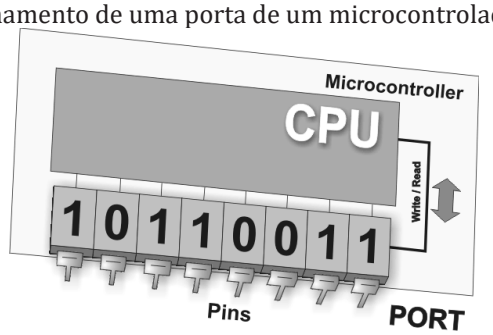


Figura 8 – CPU comunicando-se com uma porta de entrada/saída.

Uma importante especificação dos pinos de entrada/saída (I/O) é a corrente máxima que estes podem suportar. Para a maioria dos microcontroladores atuais, a corrente obtida a partir de um pino é suficiente para ativar um LED ou algum outro dispositivo de baixa corrente (1-20 mA). Quanto mais pinos de I/O, menor a corrente máxima que pode ser suportada por um pino. Em outras palavras, a corrente máxima indicada na folha de especificações de dados para o microcontrolador é compartilhada entre todas as portas de I/O.

Outra função importante de um pino de I/O é que estes podem ter resistências de pull-up externas. Essas resistências conectam os pinos à tensão positiva de alimentação que pode ser diferente daquela que é usada para alimentar o chip do microcontrolador. Isto é útil quando se deseja trabalhar com tensões distintas no mesmo circuito.

Cada porta de I/O está normalmente sob o controle de um registrador especial que indica o sentido de operação dos pinos, o que significa que cada bit do registrador determina o estado do pino correspondente. Por exemplo, escrevendo uma lógica (1) para um bit do registrador de controle (SFR), o pino equivalente da porta é automaticamente configurado como entrada e a tensão presente nele pode ser lida como nível lógico 0 ou 1. Caso contrário, ao escrever zero a um bit desse SFR, o pino equivalente é configurado como uma saída.

Unidade de Memória

A memória é a parte do microcontrolador utilizada para armazenamento de dados de forma temporária ou permanente. Memória é um termo genérico para designar componentes de um sistema capazes de armazenar dados e programas.

A maneira mais fácil de explicar a operação de uma memória é compará-la com um armário de arquivo com muitas gavetas. Suponha que as gavetas estão claramente identificadas para que seu conteúdo possa ser encontrado com facilidade através da leitura da etiqueta na parte frontal da gaveta. Da mesma forma, cada endereço de memória corresponde a um local de memória. O conteúdo de qualquer lugar pode ser acessado e lido a partir de seu endereçamento.

As operações que podem ser realizadas em uma memória são: Operação de escrita, que ocorre quando uma informação é armazenada em uma posição de memória, sobrescrevendo qualquer dado previamente armazenado, o que chamamos de operação destrutiva, e a operação de leitura, que ocorre quando endereçamos uma posição de memória e fazemos a leitura do dado armazenado previamente, o que chamamos de operação não destrutiva, o que significa que os dados ainda permanecem armazenados naquela posição endereçada.

Na figura 9 temos uma representação de uma memória. Do lado esquerdo, temos o barramento de endereços, do lado direito temos o barramento de dados e na parte de baixa temos o barramento de controle de escrita/leitura. Ao endereçar uma posição específica, a linha de dados apresenta o conteúdo armazenado naquela posição endereçada. Esse conteúdo pode ser lido para ser usado em algum processamento, ou ainda, esse conteúdo pode ser alterado mediante a escrita de um novo dado naquele endereço.

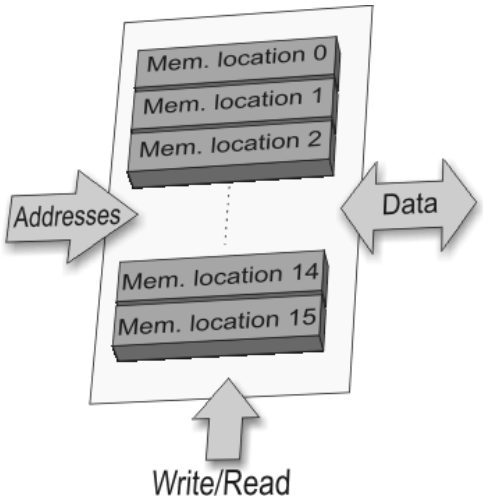


Figura 9 – Memória de escrita/leitura de um microcontrolador.

Existem tipos distintos de memória dentro do microcontrolador, tais como:

* Memória apenas de leitura (*Read Only Memory* - ROM) - É usada para salvar permanentemente o programa sendo executado. O tamanho do programa que pode ser escrito depende do tamanho da memória. Microcontroladores atuais costumam usar 16 bits de endereçamento, o que significa que eles são capazes de endereçar até 64 Kb de memória, ou seja, 65535 localidades. Para iniciantes, o programa irá raramente exceder o limite de algumas centenas de instruções. Existem vários tipos de ROM.

* Memória flash - Este tipo de memória foi concebido nos anos 80 nos laboratórios da Intel e foi apresentada como o sucessor da EPROM UV (tipo de memória ROM que usa luz ultravioleta para ser apagada). Como o conteúdo dessa memória pode ser escrito e apagado praticamente um número ilimitado de vezes, microcontroladores com memória flash são ideais para a aprendizagem, experimentação e produção em pequena escala. Devido à sua grande popularidade, a maioria dos microcontroladores é fabricada com tecnologia flash. Os microcontroladores da família Arduino utilizam memória flash para armazenamento do programa.

* Memória de acesso aleatório (*Random Access Memory* - RAM) - Uma vez que a alimentação é desligada, o conteúdo da RAM é apagado. Ela é usada para armazenar dados temporários e resultados intermediários criados e utilizados durante a operação do microcontrolador. Por exemplo, se o programa realiza uma adição, é necessário ter um registro que representa a "soma". Por esta razão, podemos ter um dos registros de memória RAM chamado de 'soma' e usado para armazenar os resultados da adição.

* Memória ROM eletricamente apagável e programável (*Electrically Erasable Programable ROM* - EEPROM) – Lê-se E²PROM. O conteúdo da EEPROM pode ser alterado durante a operação (semelhante à RAM), mas é permanente, mesmo após a perda de alimentação (similar a ROM). Assim, EEPROM é freqüentemente usada para armazenar valores, criados durante a operação, que devem ser permanentemente guardados. Por exemplo, em uma fechadura eletrônica ou um alarme, seria ótimo que este permitisse ao usuário criar e digitar uma senha, mas seria inútil se essa senha fosse perdida cada vez que a alimentação fosse desligada. A solução ideal é um microcontrolador com uma EEPROM embutida, como os microcontroladores da família Arduino.

Interrupções

O objetivo do microcontrolador é, principalmente, responder às mudanças no seu entorno. Em outras palavras, quando um evento ocorre, o microcontrolador deve fazer alguma coisa. Por exemplo, quando se aperta um botão em um controle remoto, o microcontrolador irá registrá-lo e responder pela alteração de um canal, alteração do volume para cima ou para baixo, etc.

Não seria prático se o microcontrolador passasse a maior parte de seu tempo infinitamente verificando alguns botões por horas ou dias. É por isso que o microcontrolador possui uma técnica de resposta a certos estímulos: Em vez de verificar cada pino ou bit constantemente, o microcontrolador delega essa função a circuitos específicos, chamados de controladores de interrupção, que só responderão quando algo específico acontecer, por exemplo, o pressionamento de um botão.

O sinal que informa a unidade de processamento central quando um evento como esse ocorre é chamado de sinal de interrupção. Quando a interrupção ocorre, o microcontrolador para de executar o programa principal e atende aquele evento em especial, executando uma tarefa específica.

As interrupções são recursos extremamente poderosos e práticos para o desenvolvimento de sistemas microcontrolados. A maioria dos programas usam interrupções em sua execução normal.

Unidade Central de Processamento (CPU)

Como o próprio nome sugere, esta é uma unidade que monitora e controla todos os processos dentro do microcontrolador. É constituída basicamente pelas seguintes subunidades: Unidade lógica e Aritmética (ULA), Registradores, ou unidade de memória, e a unidade de controle.

* Unidade Lógica Aritmética (ULA) realiza todas as operações matemáticas e lógicas sobre os dados;

* A unidade de memória engloba os registradores de uso geral e os de função especial.

* A unidade de controle envia todos os sinais para os diferentes circuitos internos do microcontrolador, a fim de garantir o seu correto funcionamento. O decodificador de Instruções (*Instruction Decoder*) é uma parte da unidade de controle que decodifica as instruções do programa e ativa outros circuitos com base nessa decodificação. O "conjunto de instruções", que é diferente para cada família de microcontroladores, expressa a capacidade do microcontrolador em processar dados;

A unidade lógica e aritmética e o conjunto de registradores formam uma subunidade chamada de caminho dos dados (*Datapath*). Assim, A CPU pode ser considerada como uma unidade de controle que opera sobre uma seção de dados, como mostrado na figura 10.

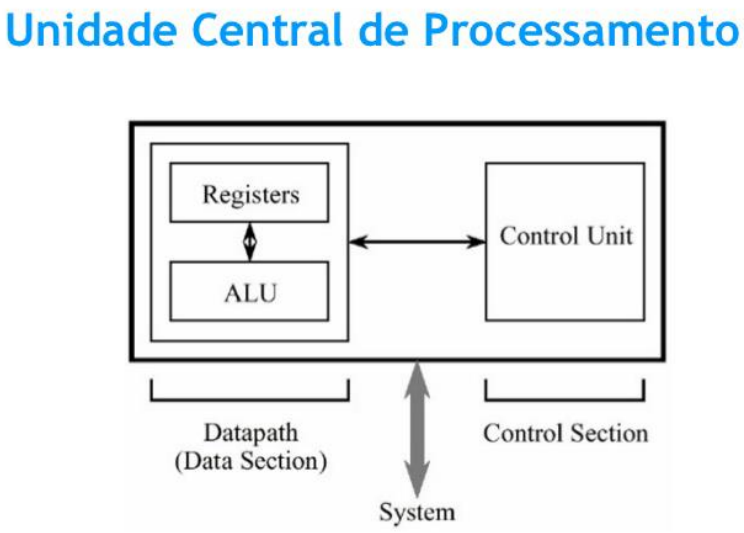


Figura 10 – Unidade Central de Processamento e suas subunidades.

Oscilador

Para que o microcontrolador execute as instruções armazenadas em sua memória, é necessária a presença de um sinal pulsante, normalmente uma onda quadrada, para que as instruções sejam executadas sequencialmente a cada pulso. Esse sinal é chamado de sinal de relógio (*clock*), e é proveniente de um circuito oscilador. O circuito oscilador é normalmente configurado de modo a utilizar um cristal de quartzo ou ressonador cerâmico, conectado a dois pinos do microcontrolador, para garantir a estabilidade de frequência, mas também pode operar como um circuito *stand-alone* (como um oscilador RC interno). É importante dizer que as instruções não são executadas necessariamente ao ritmo imposto pelo oscilador em si, mas pode ser algumas vezes mais lento. Isso acontece porque cada instrução é executada em várias etapas. Em alguns microcontroladores, o mesmo número de ciclos é necessário para executar todas as instruções, enquanto em outros, o número de ciclos é diferente para diferentes instruções. Assim, se o sistema usa um cristal de quartzo com uma frequência de 20 MHz, o tempo de execução de uma instrução não é necessariamente 50 ns, mas 200, 400 ou 800 ns, dependendo do tipo de microcontrolador.

Circuito de alimentação

Há duas coisas que merecem atenção sobre o circuito de alimentação do microcontrolador:

* *Brown out* é uma condição potencialmente perigosa que ocorre no momento em que o microcontrolador é desligado ou quando a tensão cai para um valor mínimo devido a ruído elétrico. Como o microcontrolador é composto por vários circuitos, com diferentes níveis de tensão de funcionamento, este estado pode provocar um funcionamento fora de controle. Para evitar isso, normalmente o microcontrolador possui um circuito de reset que reinicia todos os circuitos assim que o microcontrolador incorre em um estado de emergência como os descritos.

* Pino de reset é utilizado para reiniciar o microcontrolador através da aplicação de uma lógica zero (0) ou um (1) nesse pino, o que depende do tipo do microcontrolador.

Temporizadores/Contadores

O oscilador do microcontrolador utiliza cristal de quartzo para o seu funcionamento. Mesmo não sendo a solução mais simples e barata para a obtenção de um sinal de relógio, existem muitas razões para usá-lo. A frequência do oscilador é precisamente definida e muito estável, de modo que gera pulsos sempre da mesma largura, o que os torna ideais para a medição de tempo. Se for necessário medir o tempo entre dois eventos, basta contar os pulsos gerados por este oscilador. Isto é exatamente o que um temporizador, ou timer, faz.

A maioria dos programas usa este cronômetro eletrônico em miniatura. Os temporizadores são geralmente registradores de funções especiais de 8 ou 16-bits, cujo conteúdo é automaticamente incrementado por cada pulso vindo do circuito oscilador. Uma vez que um registrador atinge seu limite de contagem, uma interrupção, conhecida como estouro do contador, pode ser gerada.

Se o timer usa um oscilador de quartzo para o seu funcionamento interno, então ele pode ser usado para medir o tempo entre dois eventos (se o valor do registrador é T1 no momento em que se inicia a medição, e T2, no momento em que termina, então o tempo decorrido é igual ao resultado da subtração T2-T1). Se os registradores usam pulsos provenientes de fontes externas conectadas a um pino do microcontrolador, então o timer é transformado em um contador.

Conversor Analógico/Digital (A/D)

Sinais externos normalmente são fundamentalmente diferentes daqueles que o microcontrolador entende (zeros e uns) e têm de ser convertidos em valores compreensíveis para o microcontrolador. Um conversor analógico/digital é um circuito eletrônico que converte os sinais contínuos para discretos, ou números digitais. Em outras palavras, este circuito converte um valor analógico em um número binário e o transfere para a CPU para processamento adicional. Este módulo é utilizado para a medição de tensão analógica em um pino de entrada (valor analógico) proveniente de sensores analógicos, por exemplo.

Na figura 11, a seguir, temos uma representação de um conversor A/D sendo controlado pela CPU. O gráfico ao lado mostra a relação de conversão de sinais analógicos de 0V até 5V em valores numéricos hexadecimais que vão de 0x000 a 0x3FF.

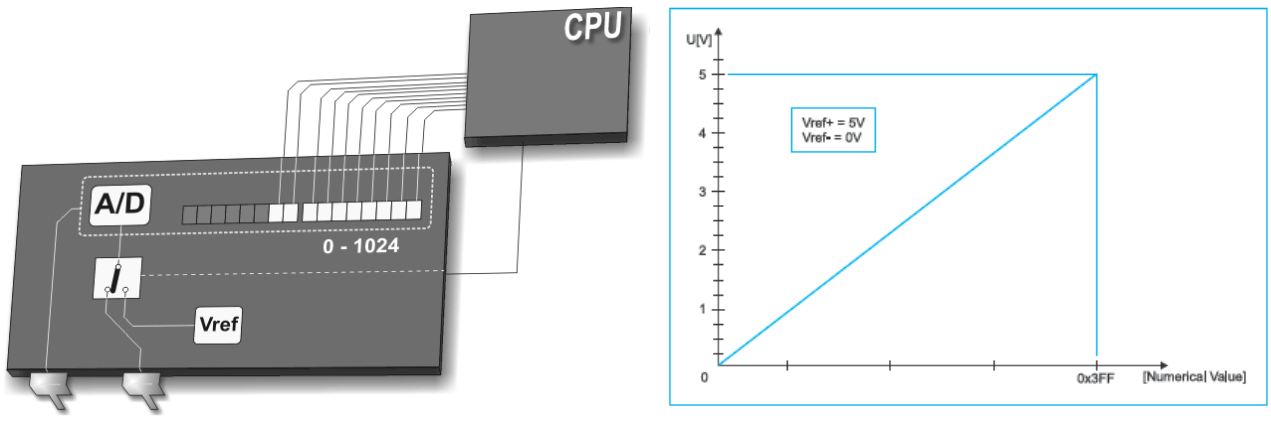


Figura 11 – Operação de um conversor analógico/digital.

Arquitetura Interna

Todos os microcontroladores usam um dos dois modelos de projeto básicos de sistemas digitais, conhecidos como arquitetura Harvard e arquitetura Von-Neumann. Eles representam duas formas diferentes de troca de dados entre a CPU e a memória.

Os microcontroladores com a arquitetura Von-Neumann (figura 12) têm apenas um bloco de memória e um barramento de dados. Como todos os dados são trocados através dessas linhas, o barramento pode ficar sobrecarregado e a comunicação se torna lenta e ineficiente. A CPU pode apenas executar uma instrução de leitura ou gravação de dados para a memória. Ambas não podem ocorrer ao mesmo tempo, uma vez que instruções e dados usam o mesmo barramento. Praticamente todos os computadores modernos adotam essa arquitetura.

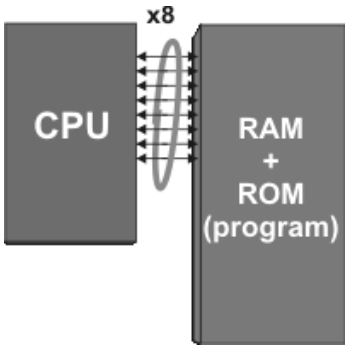


Figura 12 – Arquitetura Von-Neumann com barramento de 8 bits.

Os microcontroladores com arquitetura Harvard (figura 13) tem dois barramentos diferentes. Um deles conecta a CPU com a memória de dados (RAM) e o outro conecta a CPU com a memória de programa. Assim, a CPU pode ler uma instrução e ter acesso a dados da memória, ao mesmo tempo.

No caso do Arduino, durante o processo de escrita, um programa apenas manipula dados de 8 bits (exceto no Arduino Due, cuja arquitetura é de 32 bits). Em outras palavras, tudo o que você pode mudar a partir do programa é de 8 bits. Todos os programas escritos para estes microcontroladores serão armazenados na memória flash do microcontrolador, depois de serem compilados em código de máquina. No entanto, as posições da memória flash não tem apenas 8, mas um número maior de bits. O resto dos bits representa a instrução, especificando para a CPU o que fazer com os dados de 8 bits.

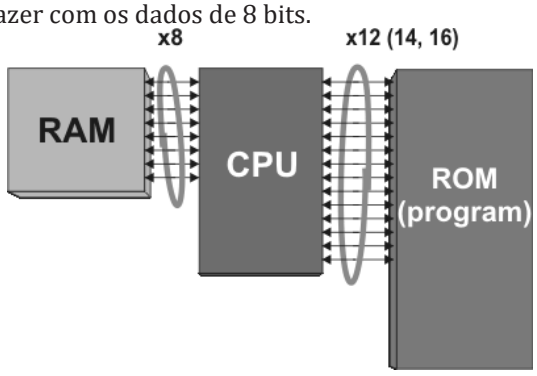


Figura 13 – Arquitetura Harvard.

O Conjunto de Instruções

Todas as instruções compreensíveis para o microcontrolador formam o seu conjunto (*set*) de instruções. Quando você escreve um programa em linguagem Assembly, por exemplo, na verdade você especifica uma série de instruções do seu *set*, na ordem em que devem ser executadas.

De acordo com o número de instruções disponíveis no *set*, os fabricantes costumam considerar uma das duas abordagens a seguir:

* Conjunto reduzido de instruções (RISC-*Reduced Instruction Set Computer*) - Neste caso, o microcontrolador reconhece e executa apenas algumas operações básicas (adição, subtração, cópia, etc.). Outras operações mais complexas são realizadas através da combinação delas. Por exemplo, a multiplicação é feita através da realização de sucessivas adições. Os microcontroladores da família Arduino são máquinas RISC com uma arquitetura Harvard modificada, conhecida como AVR.

* Conjunto complexo de instruções (CISC-*Complex Instruction Set Computer*) - CISC é o oposto ao RISC. São microcontroladores concebidos para reconhecer mais de 200 instruções diferentes, podendo realizar inúmeras tarefas em alta velocidade. No entanto, é preciso entender como aproveitar tudo o que estas instruções oferecem, o que não é nada simples. Processadores, como Intel Core 2 Duo, Intel I3, I5, I7 são do tipo CISC.

4.2 Modelo genérico de um microcontrolador

Todos os elementos de hardware vistos até agora, e tantos outros não apresentados aqui, fazem parte da eletrônica embutida em um microcontrolador. Esses elementos estão todos interligados para desempenhar diferentes tarefas, dependendo das necessidades do projetista. A figura 14 ilustra de forma simplificada como esses elementos estão interligados para potencializar o uso de um microcontrolador.

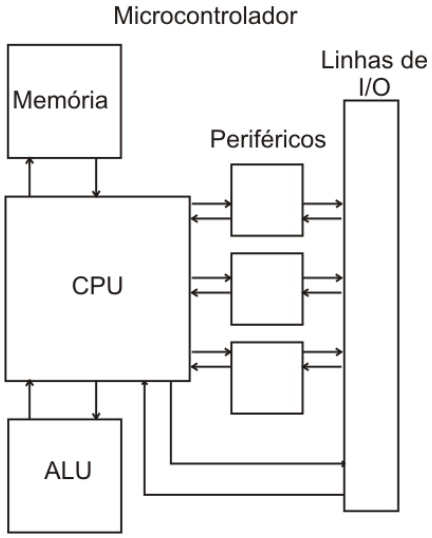


Figura 14 – Elementos de hardware de um microcontrolador.

4.3 Como fazer a escolha certa?

Você é novato e tomou a decisão de se aventurar em trabalhar com os microcontroladores. Parabéns pela sua escolha! No entanto, não é tão fácil escolher o microcontrolador mais adequado para desenvolver uma aplicação, como pode parecer. O problema não está no número limitado de opções, mas o oposto!

Antes de começar a projetar um dispositivo microcontrolado, pense o seguinte: quantas linhas de entrada/saída eu preciso para a aplicação? O dispositivo deve realizar operações especiais além de simplesmente acionar relés? Será que preciso de algum módulo especializado, tal como interfaces de comunicação serial, etc.? Um conversor A/D? Quando você cria uma imagem clara do que você precisa, o intervalo de selecção é bastante reduzido e aí é hora de pensar em preços.

Se você pensar em todas essas coisas pela primeira vez, então, tudo parece um pouco confuso. Primeiro, selecione o fabricante, ou seja, a família do microcontrolador. Estude um modelo específico. Saiba do que você precisa, não entre em detalhes. Resolva um problema específico por vez. Em seguida, você vai perceber que aprendendo um modelo daquela família, você estará apto a trabalhar com qualquer outro modelo.

O Arduino provavelmente é a melhor escolha para iniciar estudos com microcontroladores devido às facilidades de utilização, vasta quantidade de material publicado na internet, baixo preço, versatilidade e grande poder computacional de processamento. ■