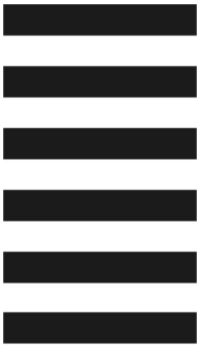


Centro de Formação Profissional Pedro Martins Guerra

ELETRÔNICA DIGITAL

Itabira

2005

FIEMG
CIEMG
SESI
SENAI
IEL

Sistema FIEMG

Presidente da FIEMG
Robson Braga de Andrade

Gestor do SENAI
Petrônio Machado Zica

Diretor Regional do SENAI e
Superintendente de Conhecimento e Tecnologia
Alexandre Magno Leão dos Santos

Gerente de Educação e Tecnologia
Edmar Fernando de Alcântara

Elaboração
Equipe Técnica - Núcleo Eletroeletrônica

Unidade Operacional

Centro de Formação Profissional Pedro Martins Guerra
Itabira – MG
2005

Revisão
Equipe Técnica - Centro de Formação Profissional Pedro Martins Guerra
Itabira – MG 2005

FIEMG
CIEMG
SESI
SENAI
IEL
Sistema FIEMG

Sumário

Apresentação

"Muda a forma de trabalhar, agir, sentir, pensar na chamada sociedade do conhecimento. "

Peter Drucker

O ingresso na sociedade da informação exige mudanças profundas em todos os perfis profissionais, especialmente naqueles diretamente envolvidos na produção, coleta, disseminação e uso da informação.

O **SENAI**, maior rede privada de educação profissional do país,sabe disso , e ,consciente do seu papel formativo , educa o trabalhador sob a égide do conceito da competência***:" formar o profissional com responsabilidade no processo produtivo, com iniciativa na resolução de problemas, com conhecimentos técnicos aprofundados, flexibilidade e criatividade, empreendedorismo e consciência da necessidade de educação continuada.***"**

Vivemos numa sociedade da informação. O conhecimento , na sua área tecnológica, amplia-se e se multiplica a cada dia. Uma constante atualização se faz necessária. Para o **SENAI**, cuidar do seu acervo bibliográfico, da sua infovia, da conexão de suas escolas à rede mundial de informações – Internet - é tão importante quanto zelar pela produção de material didático.

Isto porque, nos embates diários,instrutores e alunos , nas diversas oficinas e laboratórios do **SENAI**, fazem com que as informações, contidas nos materiais didáticos, tomem sentido e se concretizem em múltiplos conhecimentos.

O **SENAI** deseja , por meio dos diversos materiais didáticos, aguçar a sua curiosidade, responder às suas demandas de informações e construir *links* entre os diversos conhecimentos, tão importantes para sua formação continuada !

Gerência de Educação e Tecnologia

1. Introdução

Sob o ponto de vista da evolução tecnológica, os cenários previsíveis são ilimitados. Ela poderá nos assegurar uma qualidade de vida sem precedentes na história humana. Liberando o homem da execução de tarefas tediosas, perigosas e repetitivas. Por outro lado, a expansão e o aprofundamento devem ser realizados de maneira correta e principalmente nunca prejudicando o meio-ambiente.

Se soubermos nos educar e educar as gerações vindouras, de modo a exercermos nossa condição de cidadãos, participando ativamente da construção da sociedade, se lutarmos por uma reformulação educacional que assegure educação a todos com qualidade e, fornecendo ao indivíduo ferramental que possibilite seu crescimento, inserção na sociedade e, principalmente se ao lado destes dois fatores tivermos uma postura ética em nossas relações e atividades, a sociedade e o ser humano alcançarão um novo estágio evolutivo, realizando um salto qualitativo. Caso contrário marcharemos para barbárie. A escolha é nossa.

A Eletrônica Digital está ligada a esta evolução tecnológica de uma forma bem interessante, o homem necessitou já no início desta evolução de criar formas diferentes para a numeração, permitindo manipular dados de forma lógica e aritmética. Criou-se também componentes eletrônicos cada vez mais rápidos e eficientes.

2. Sistema de numeração

2.1. Sistema Decimal

O sistema decimal de numeração é composto por 10 símbolos ou dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9; usando tais símbolos, podemos expressar qualquer quantidade. O sistema decimal, também chamado de sistema de base 10, pois ele usa 10 dígitos, evoluiu naturalmente como resultado do fato de os seres humanos terem 10 dedos.

O sistema decimal é um sistema de valor posicional, no qual o valor de um dígito depende de sua posição. Por exemplo, o número 594 significa:

$$5 \times 100 + 9 \times 10 + 4 \times 1 = 594$$

centena dezena unidade

$$5 \times 10^2 + 9 \times 10^1 + 4 \times 10^0 = 594$$

Neste exemplo podemos notar que o algarismo menos significativo (4) multiplica a unidade (1 ou 10^0), o segundo algarismo (9) multiplica a dezena (10 ou 10^1) e o mais significativo (5) multiplica a centena (100 ou 10^2). A soma desses resultados irá representar o número.

Podemos notar ainda, que de maneira geral, a regra básica de formação de um número consiste no somatório de cada algarismo correspondente multiplicado pela base (no exemplo 10) elevada por um índice conforme o posicionamento do algarismo no número.

2.2. Sistema Binário

No sistema binário de numeração, existem apenas 2 algarismos: **0** (zero) e **1**(um). Por isso sua base é dois.

Cada dígito ou algarismo binário é chamado de bit (do inglês “binary digit”, ou seja dígito binário). Um bit é, pois, a menor unidade de informação nos circuitos digitais.

A tabela 01, mostra a correspondência entre números decimais e binários:

DECIMAL	*BINÁRIO*	*DECIMAL*	*BINÁRIO*
0	0	**10**	1010
1	1	**11**	1011
2	10	**12**	1100
3	11	**13**	1101
4	100	**14**	1110
5	101	**15**	1111
6	110	**16**	10000
7	111	**17**	10001
8	1000	**18**	10010
9	1001	**19**	10011

Tabela 01 – Binário x Decimal

Empregando a propriedade do valor de posição do dígito, podemos representar qualquer valor numérico com os dígitos 0 e 1.

Como a base de numeração binária é 2, o valor de posição é dado pelas potências de base 2, como mostra a tabela a seguir:

Potências de base 2	2^4	2^3	2^2	2^1	2^0
Valor de Posição	16	8	4	2	1

O valor da posição é indicado pelo expoente da base do sistema numérico. Esse valor aumenta da direita para a esquerda. O valor da posição do bit mais significativo (de maior valor) será a base elevada a m-1(m = número de dígitos).

Por exemplo, 101011 é um número binário de 6 bits. Ao aplicar a fórmula, temos 6– 1 = 5. Assim, o bit mais significativo terá como valor de posição 2^5.

Binário	**1**	**0**	**1**	**0**	**1**	**1**
Valor de Posição	2^5	2^4	2^3	2^2	2^1	2^0

MSB – do inglês "most significant bit" ou seja, bit mais significativo
LSB – do inglês "least significant bit" ou seja, bit menos significativo

Conversão do Sistema Binário para o Sistema Decimal

Para converter um número binário em decimal, deve-se multiplicar cada bit pelo seu valor de posição (que é indicado pelo valor da base) e somar os resultados.

Exemplo:

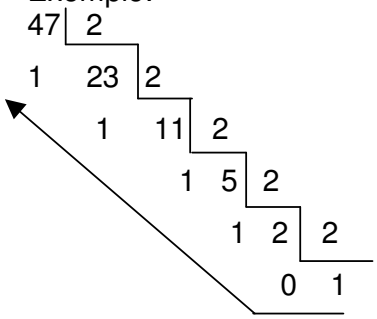
Na conversão de 1010_2 para o sistema decimal, procede-se da seguinte forma:

Potência de 2	2^3	2^2	2^1	2^0	
Binário	1	0	1	0	
Valor de posição	1 x 8	0 x 4	1 x 2	0 x 1	
Nº decimal	8 +	0 +	2 +	0	$= 10_{10}$

Conversão do Sistema Decimal para o Sistema Binário

A conversão de números do sistema decimal para o sistema binário é realizada efetuando-se divisões sucessivas do número decimal pela base a ser convertida (no caso 2) até o último quociente possível. O número transformado será composto por este último quociente (algarismo mais significativo) e, todos os restos, na ordem inversa às divisões.

Exemplo:



O último quociente será o algarismo mais significativo e ficará colocado à esquerda. Os outros algarismos seguem-se na ordem até o 1º resto:

1	***0***	***1***	***1***	***1***	***1***
último	5º	4º	3º	2º	1º
quociente	resto	resto	resto	resto	resto

$101111_2 = 47_{10}$

2.3. Sistema Octal

O sistema octal de numeração é um sistema de base 8 no qual existem 8 algarismos: 0, 1, 2, 3, 4, 5, 6 e 7.

Para representarmos a quantidade oito, agimos do mesmo modo visto anteriormente para números binários e decimais, colocamos o algarismo 1 seguido do algarismo 0, significando que temos um grupo de oito adicionados a nenhuma unidade. A tabela 02 mostra a correspondência entre números decimais e octais

DECIMAL	OCTAL	DECIMAL	OCTAL
0	0	**9**	11
1	1	**10**	12
2	2	**11**	13
3	3	**12**	14
4	4	**13**	15
5	5	**14**	16
6	6	**15**	17
7	7	**16**	20
8	10	**17**	21

Tabela 02 – OCTAL x Decimal

Conversão do Sistema Octal para o Sistema Decimal

Para convertermos um número octal em decimal, utilizamos o conceito básico de formação de um número.

Vamos por exemplo converter o número 144_8 em decimal:

$$\begin{array}{ccccccc} 8^2 & & 8^1 & & 8^0 & & \\ \textbf{\textit{1}} & & \textbf{\textit{4}} & & \textbf{\textit{4}} & & \\ 1x8^2 & + & 4x8^1 & + & 4x8^0 & = & \\ 1x64 & + & 4x8 & + & 4x1 & = & 64 + 32 + 4 = 100_{10} \end{array}$$

Conversão do Sistema Decimal para o Sistema Octal

O processo é análogo à conversão do sistema decimal para binário, somente que neste caso, utilizaremos a divisão por 8, pois sendo o sistema octal, sua base é igual a 8.

Para exemplificar, vamos converter o número 92_{10} para o sistema octal:

	92	8		
1º resto	4	11	8	
2º resto		3	1	$92_{10} = 134_8$
último quociente				

Conversão do Sistema Octal para o Sistema Binário

Vamos usar um número octal qualquer, por exemplo, 27_8. A regra consiste em transformar cada algarismo diretamente no correspondente em binário, respeitando-se o número padrão de bits do sistema, sendo para o octal igual a três ($2^3 = 8$, base do sistema octal). Assim sendo, temos:

2	***7***	
010	111	$27_8 = 10111_2$

Convém lembrar que a regra só é válida entre sistemas numéricos de base múltipla de 2^N, sendo N um número inteiro.

Conversão do Sistema Binário para o Sistema Octal

Para efetuar esta conversão, vamos aplicar o processo inverso ao utilizado na conversão de octal para binário. Como exemplo, vamos utilizar o número 110010_2.

Para transformar este número em octal, vamos primeiramente separá-lo em grupos de 3 bits a partir da direita, e efetuar a conversão de cada grupo de bits diretamente para o sistema octal:

110	010
6	***2***

O número convertido será composto pela união dos algarismos obtidos.
$110010_2 = 62_8$

No caso do último grupo se formar incompleto, adicionamos zeros à esquerda, até completá-lo com 3 bits. Para exemplificar, vamos converter o número 1010_2 em octal:

001	010	$1010_2 = 12_8$
1	***2***	

2.4. Sistema Hexadecimal

O sistema hexadecimal tem a base 16. Os 16 símbolos que constituem a numeração hexadecimal são os seguintes algarismos e letras: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F.

A tabela 03 a seguir mostra relação entre numeração decimal e hexadecimal

Decimal	Hexa	Decimal	Hexa	Decimal	Hexa
0	0	11	B	22	16
1	1	12	C	23	17
2	2	13	D	24	18
3	3	14	E	25	19
4	4	15	F	26	1A
5	5	16	10	27	1B
6	6	17	11	28	1C
7	7	18	12	29	1D
8	8	19	13	30	1E
9	9	20	14	31	1F
10	A	21	15	32	20

Tabela 03 – Hexadecimal x Decimal

Este sistema é muito utilizado na área dos microprocessadores e também no mapeamento de memórias em sistemas digitais, tratando-se de um sistema numérico muito importante, sendo aplicado em projetos de software e hardware.

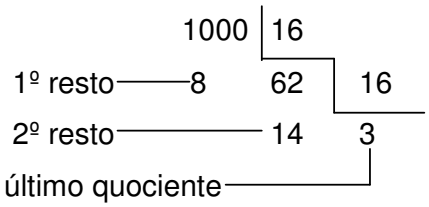
Conversão do Sistema Hexadecimal para o Sistema Decimal

A regra de conversão é análoga à de outros sistemas, somente neste caso, a base é 16. Como exemplo, vamos utilizar o número $3F_{16}$ e convertê-lo em decimal:

$16^0 \quad 16^1$
$3 \quad F$
$3 \times 16^1 + F \times 16^0$
sendo $F_{16} = 15_{10}$, substituindo temos:
$3 \times 16^1 + 15 \times 16^0 = 3 \times 16 + 15 \times 1 = 63_{10}$ $\qquad 3F_{16} = 63_{10}$

Conversão do Sistema Decimal para o Sistema Hexadecimal

Da mesma forma que nos casos anteriores, esta conversão se faz através de divisões sucessivas pela base do sistema a ser convertido. Para exemplificar vamos transformar o número 1000_{10} em hexadecimal:



Sendo $14_{10} = E_{16}$, temos: $3E8_{16}$ $\qquad 1000_{10} = 3E8_{16}$

Conversão do Sistema Hexadecimal para o Sistema Binário

É análoga à conversão do sistema octal para o sistema binário, somente, neste caso, necessita-se de 4 bits para representar cada algarismo hexadecimal.

Como exemplo, vamos converter o número $C13_{16}$ para o sistema binário:

C	($C_{16} = 12_{10}$)	**1**	**3**
1100		0001	0011

$C13_{16} = 110000010011_2$

Conversão do Sistema Binário para o Sistema Hexadecimal

É análoga à conversão do sistema binário para o octal, somente que neste caso, agrupamos de 4 em 4 bits da direita para a esquerda. A título de exemplo, vamos transformar o número 10011000_2 em hexadecimal:

1001	1000	$10011000_2 = 98_{16}$
9	8	

3. Aritimética Binária

As operações aritméticas podem ser realizadas com números binários, exatamente da mesma forma como com números decimais. Em alguns casos, porém, certas operações binárias são feitas de modo diferente das suas equivalentes decimais por causa de considerações de hardware.

Adição Binária

A adição de dois números binários é executada exatamente da mesma maneira que a adição de números decimais. De fato, a adição binária é mais simples, já que há menos casos para aprender. Vamos, primeiro, recordar uma adição decimal.

```
  3  7  6 ◄— LSD
+4  6  1
  8  3  7
```

A posição do dígito menos significativo (LSD) é adicionada primeiro, produzindo-se uma soma igual a 7. Os dígitos na segunda posição são somados, então, para produzir uma soma igual a 13, que produz um *vai-um* de 1 para a terceira posição. Isto produz uma soma igual a 8 na terceira posição.

Estes mesmos passos gerais são seguidos na adição binária. No entanto, existem apenas quatro casos que podem ocorrer na adição dos dígitos binários (bits) em qualquer posição. São eles:

```
    0 + 0 = 0
    1 + 0 = 1
    1 + 1 = 0   Mais um vai-um de 1 para a próxima posição
1 + 1 + 1 = 1   Mais um vai-um de 1 para a próxima posição
```

Este último caso ocorre quando os dois bits em uma dada posição são iguais a 1 e existe um vai-um da posição anterior. Aqui temos vários exemplos da adição de dois números binários:

```
  0 1 1 (3)          1 0 0 1 ( 9)          1 1 , 0 1 1  (3,375)
+ 1 1 0 (6)        + 1 1 1 1 (15)        + 1 0 , 1 1 0  (2,750)
  1 0 0 1 (9)        1 1 0 0 0 (24)        1 1 0 , 0 0 1  (6,125)
```

A adição é a operação aritmética mais importante nos sistemas digitais, pois as operações de subtração, multiplicação e divisão, da forma como elas são executadas na maioria dos computadores digitais e calculadoras modernas, na verdade usam apenas a adição como sua operação básica.

Subtração Binária

Em muitos computadores grandes e na maioria dos minicomputadores, a operação de subtração é realizada usando-se a operação de adição. Este processo requer o uso da forma complemento-de-2.

O complemento-de-2 de um número binário é obtido trocando-se cada 0 por 1, e cada 1 por 0, e somando-se 1 ao resultado. O primeiro passo, a inversão de cada bit, é chamado complementação de 1. Por exemplo, o complemento-de-1 de 10110110 é 01001001.

O complemento-de-2 de um número binário é formado somando-se 1 ao complemento-de-1 do mesmo número. Por exemplo, o complemento-de-2 de 10110110 é obtido como a seguir:

Número	10110110
Complemento-de-1	01001001
Soma-se 1	+ 1
Complemento-de-2	01001010

A operação de subtração pode ser executada convertendo-se o subtraendo (o número a ser subtraído em seu complemento-de-2 e, então, somando-se ao minuendo (o número do qual se subtrai). Para ilustrar, considere a subtração no número 1001 de 1100 (decimal 9 de decimal 12).

Subtração Normal		***Subtração em complemento-de-2***	
Minuendo	1100	Minuendo	1100
Subtraendo	- 1001	Complemento-de-2 do subtraendo	+0111
Diferença	0011	Soma	10011
		Desprezar vai-um final	▲

Assim, o resultado final é 0011 (decimal 3).

Multiplicação binária

A multiplicação de números binários é feita da mesma maneira que a multiplicação de números decimais. O processo é, na verdade, mais simples, já que os dígitos multiplicadores são ou 0 ou 1, de modo que, em qualquer instante, estaremos multiplicando por 0 ou por 1 e por nenhum outro dígito. O exemplo seguinte ilustra o fato:

1 0 0 1	Multiplicando = 9_{10}
1 0 1 1	Multiplicador = 11_{10}
1 0 0 1	
1 0 0 1	
0 0 0 0	Produtos parciais
1 0 0 1	
1 1 0 0 0 1 1	Produto final = 99_{10}

Neste exemplo, tanto o multiplicando como o multiplicador estão na forma binária verdadeira e não são usados bits de sinal. Os passos seguidos no processo são exatamente os mesmos da multiplicação decimal. Primeiro, o LSB do multiplicador é examinado; no nosso exemplo, ele é igual a 1. Este 1 multiplica o multiplicando produzindo 1001, que é escrito abaixo como o primeiro produto parcial. A seguir, o segundo bit do multiplicador é examinado.

Ele é um 1, de modo que 1001 é escrito no segundo produto parcial. Note que este segundo produto parcial é deslocado de uma posição para a esquerda, em relação ao primeiro. O terceiro bit do multiplicador é 0, de modo que 0000 é escrito como sendo o terceiro produto parcial; de novo, ele é deslocado de uma posição para a esquerda em relação ao produto parcial anterior.

O quarto bit do multiplicador é igual a 1, de forma que o último produto parcial é 1001, novamente deslocado de uma posição para a esquerda. Os quatro produtos parciais são, então, somados para produzir o produto final.
A maioria das máquinas digitais pode somar apenas dois números binários de cada vez. Por esta razão, os produtos parciais formados durante a multiplicação não podem ser todos somados ao mesmo tempo. Em vez disto, eles são somados dois por vez, isto é, o primeiro é somado ao segundo e o resultado é somado ao terceiro, e assim por diante. Este processo está ilustrado para o exemplo anterior.

	1 0 0 1	Primeiro produto parcial
Soma-se	1 0 0 1	Segundo produto parcial deslocado para esquerda
	1 1 0 1 1	Soma dos primeiros produtos parciais
Soma-se	0 0 0 0	Terceiro produto parcial deslocado para esquerda
	0 1 1 0 1 1	Soma dos três primeiros produtos parciais
Soma-se	1 0 0 1	Quarto produto parcial deslocado para esquerda
	1 1 0 0 0 1 1	Soma de quatro produtos parciais = produto total

Divisão binária
O procedimento para dividir um número binário (o dividendo) por outro (o divisor) é igual àquele que é seguido para os números decimais, ao qual normalmente nos referimos como “divisão longa”. O processo real é mais simples em binário; pois, quando estamos verificando quantas vezes o divisor “cabe” no dividendo, existem apenas duas possibilidades: 0 ou 1. Para ilustrar, considere o seguinte exemplo:

```
      0 0 1 1
1 1 | 1 0 0 1      (9 / 3 = 3)
      0 1 1
      -------
      0 0 1 1
```

```
        0 0 1 0 , 1
1 0 0 | 1 0 1 0 , 0
        1 0 0          (10 / 4 = 2,5)
        -------
          1 0 0
          1 0 0
          -----
              0
```

Na maioria das máquinas digitais modernas, as subtrações que fazem parte da operação de divisão são, usualmente, executadas usando-se a subtração com complemento-de-2, isto é, complementando o subtraendo e somando.

4. Funções e Portas Lógicas

Em 1854, o matemático inglês George Boole apresentou um sistema matemático de análise lógica conhecido como álgebra de Boole.

Apenas em 1938, o engenheiro americano Claude Elwoode Shannom utilizou as teorias da álgebra de Boole para a solução de problemas de circuitos de telefonia com relés, praticamente introduzindo na área tecnológica o campo da eletrônica digital.

Esse ramo da eletrônica emprega em seus sistemas um pequeno grupo de circuitos básicos padronizados conhecidos como portas lógicas.
Através da utilização conveniente destas portas, podemos "implementar" todas as expressões geradas pela álgebra de Boole, que constituem a base dos projetos dos sistemas já referidos.

Funções Lógicas E, OU, NÃO, NE e NOU.
Faremos a seguir, o estudo das principais funções lógicas que na verdade derivam dos postulados da álgebra de Boole, sendo as variáveis e expressões envolvidas denominadas de booleanas.
Nas funções lógicas, temos apenas dois estados distintos:

*O estado **0** (zero) e o estado **1** (um).*

O estado 0 representará, por exemplo: portão fechado, aparelho desligado, ausência de tensão, chave aberta, não, etc. O estado 1 representará, então: portão aberto, aparelho ligado, presença de tensão, chave fechada, sim, etc.

Note, então, que se representarmos por **0** uma situação, representaremos por **1** a situação contrária. Deve-se salientar aqui, que cada variável booleana da função lógica pode assumir somente 2 situações distintas **0** ou **1**.

Função E ou AND
A função E é aquela que executa a multiplicação de 2 ou mais variáveis booleanas. É também conhecida como função AND, nome derivado do inglês. Sua representação algébrica para 2 variáveis é S = A.B, onde se lê S = A e B.

A figura 01 mostra o circuito elétrico equivalente a porta E.

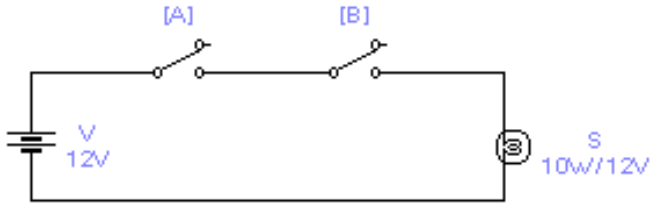


Figura 01 – Circuito elétrico equivalente da Função E

CONVENÇÃO

CHAVE ABERTA	= 0	LÂMPADA APAGADA	= 0
CHAVE FECHADA	= 1	LÂMPADA ACESA	= 1

Tabela-verdade de uma Função E ou AND

Chamamos de tabela-verdade um mapa onde colocamos todas as possíveis situações com seus respectivos resultados. Na tabela 04, iremos encontrar o modo como a função se comporta. A seguir, iremos apresentar a tabela-verdade de uma função E ou AND para 2 variáveis de entrada:

A	***B***	***S***
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 04 – Tabela Verdade função E para 2 entradas

Simbologia

A porta E é um circuito que executa a função E, sendo representada na prática, através do símbolo abaixo:

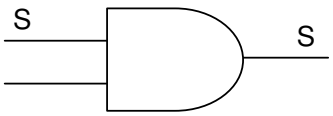
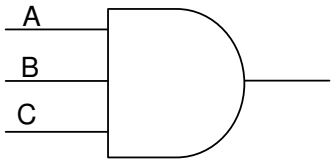


Figura 02 – Simbologia da Função E

Teremos a saída no estado **1** se, e somente se, as 2 entradas forem iguais a **1**, e teremos a saída igual a **0** nos demais casos.

Podemos estender o conceito da função E com 2 variáveis de entrada para qualquer número de entradas. Para exemplificar, mostraremos uma porta E de 3 variáveis de entrada, sua tabela-verdade e sua expressão booleana (Figura 03 e Tabela 4.1):



$$S = A \cdot B \cdot C$$

Figura 03 – Simbologia da Função E para 3 entradas e sua expressão

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabela 04.1 – Tabela Verdade função E para 3 entradas

Notamos que a tabela-verdade mostra as 8 possíveis variações das variáveis de entrada e seus respectivos resultados na saída. O número de situações possíveis é igual a 2^N, onde N é o número de variáveis de entrada.

No exemplo: N = 3 2^3 = 8.

Função OU ou OR

A função OU ou OR é aquela que assume valor **1** quando uma ou mais variáveis da entrada forem iguais a **1** e assume valor **0** se, e somente se, todas as variáveis de entrada forem iguais a **0**. Sua representação algébrica para 2 variáveis de entrada é S = A + B, onde se lê S = A ou B.

O termo OR também utilizado é derivado do inglês.

O circuito da figura 04 abaixo representa a função OU:

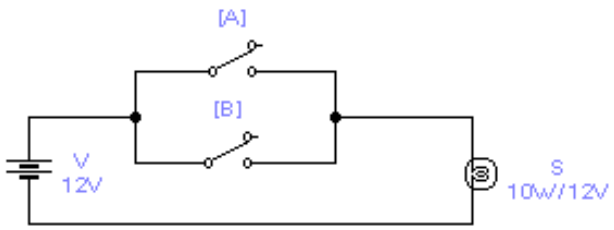


Figura 04 – Circuito elétrico equivalente da Função OR

Tabela-verdade da Função OR ou OU

Nesta tabela 05, temos todas as situações possíveis com os respectivos valores que a função OU assume.

A	*B*	*S*
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 05 – Tabela Verdade função OR para 2 entradas

Simbologia

È a porta que executa a função OU, representada através do símbolo abaixo:

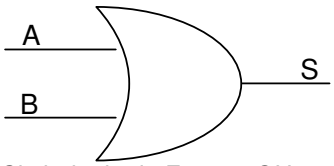


Figura 05 – Simbologia da Função OU para 2 entradas

O conceito da função OU pode ser estendido para mais de 2 variáveis de entrada.

Como exemplo veremos uma porta OU e sua expressão booleana com 4 variáveis de entrada:

S = A + B + C + D

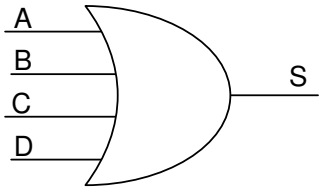


Figura 06 – Simbologia da Função OU para 4 entradas

Função NÃO ou NOT

A função NÃO, ou função complemento, é aquela que inverte ou complementa o estado da variável de entrada. Se a variável de entrada for **1**, ela se tornará **0** na saída. Se a variável de entrada for **0**, ela se tornará **1**na saída.

A operação lógica inversão é realizada pela porta lógica NÃO (“NOT” em inglês). Ela consiste em converter uma dada proposição em uma proposição a ela oposta, é representada algebricamente por: ***S = A.***

Essa expressão é lida da seguinte forma: saída S é igual a não A, pois o traço sobre o A significa não. Para A pode-se dizer também A barrado ou A negado. Veja na figura 07 abaixo o circuito elétrico equivalente de uma porta NÃO.

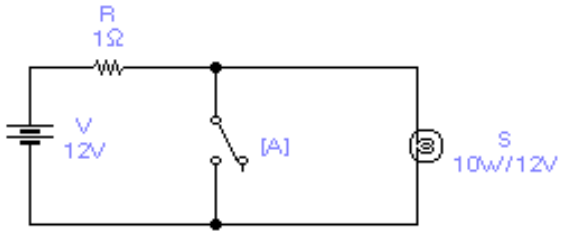


Figura 07 – Circuito elétrico equivalente da Função NÃO

CONVENÇÃO

CHAVE ABERTA	= 0	LÂMPADA ACESA	= 1
CHAVE FECHADA	= 1	LÂMPADA APAGADA	= 0

A lâmpada S acenderá (**1**) quando a chave A estiver aberta (**0**). Quando a chave A estiver fechada (**1**), a lâmpada será curto-circuitada e não acenderá (**0**).

Veja a seguir o símbolo, as combinações possíveis da chave e a respectiva tabela-verdade.

A	***S***
0	1
1	0

Figura 08 – Tabela Verdade e Simbologia da Função NÃO

PORTAS LÓGICAS DERIVADAS

Os sistemas digitais mais complexos, como computadores de grande porte, são construídos a partir das portas lógicas básicas E, OU e NÃO. A partir dessas portas, podem-se construir quatro outras denominadas portas lógicas derivadas.

Elas são: porta NE (ou NÃO E), porta NOU (ou NÃO OU), porta OU EXCLUSIVO e porta NÃO OU EXCLUSIVO.

Função NÃO E, NE ou NAND.

Como o próprio nome "NÃO E" diz: essa função é a composição da função E com a função NÃO, ou seja, teremos a função E invertida. É representada algebricamente da seguinte forma:

$S = \overline{(A \cdot B)}$, onde o traço indica que teremos a inversão do produto A . B.

Porta NE ou NAND é o bloco lógico que executa a função NAND. Sua tabela-verdade e símbolo são mostrados a seguir:

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

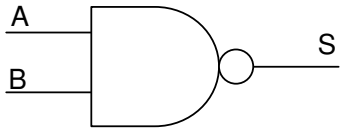


Figura 09 – Tabela Verdade e Simbologia da Função NÃND

Podemos também formar uma porta NAND através da composição de uma porta AND com um inversor ligado a sua saída.

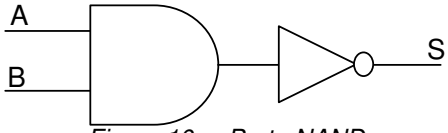


Figura 10 – Porta NAND

Função NÃO OU, NOU ou NOR.

A função NOU é a composição da função NÃO com a função OU, ou seja, a função NOU será o inverso da função OU. É representada da seguinte forma:

$S = \overline{(A+B)}$, onde o traço indica a inversão da soma A + B.

Porta NOU ou NOR é o bloco lógico que executa a função NOU. Sua tabela-verdade e símbolo são mostrados abaixo na figura 11

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

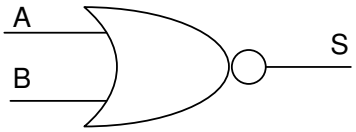


Figura 11 – Tabela Verdade e Simbologia da Função NOR

De maneira análoga, podemos formar uma porta NOU utilizando uma OU e um inversor ligado à sua saída.

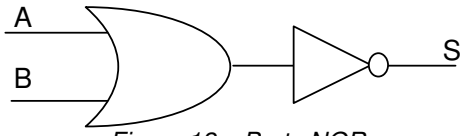
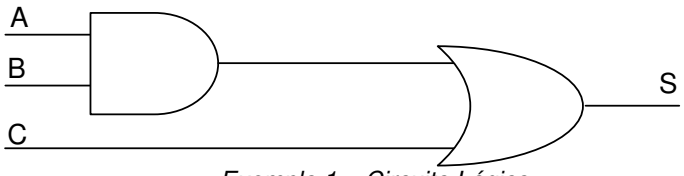


Figura 12 – Porta NOR

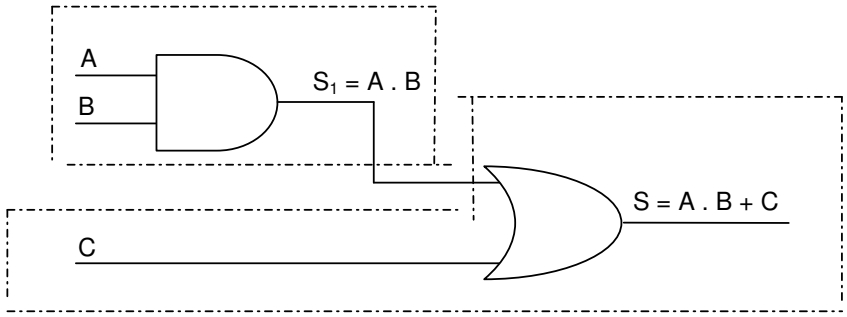
Circuitos Lógicos e suas Expressões Booleanas

Todo circuito lógico executa uma expressão booleana e, por mais complexo que seja, é formado pela interligação das portas lógicas básicas. Podemos obter a expressão booleana que é executada por um circuito lógico qualquer. Para mostrar o procedimento, vamos obter a expressão que o circuito abaixo executa.



Exemplo 1 – Circuito Lógico

Dividindo o circuito em duas partes, teremos na saída S_1 o produto A . B, que equivale à expressão de uma porta AND. Como S_1 é injetada em uma das entradas da porta OU pertencente à Segunda parte do circuito e na outra entrada está a variável C, a expressão de saída será: $S = S_1 + C$.



A expressão final é obtida substituindo a expressão de S1 na expressão acima, obtendo então:

S = A . B + C

A forma mais simples de levantar a expressão de um circuito é escrever nas saídas dos diversos blocos básicos do circuito, as expressões por estes executadas.

Desenvolvimento de um Circuito Lógico a Partir da Expressão

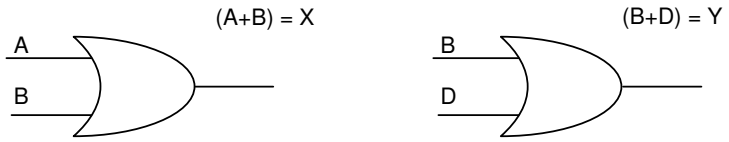
O método para obter o circuito lógico que executa uma expressão booleana consiste em identificar as portas lógicas na expressão e desenhá-las com as respectivas ligações, a partir das variáveis de entrada.

A resolução deve respeitar a ordem de prioridade da aritmética elementar, iniciando pelos parênteses seguido pelos colchetes e por último as chaves. Em expressões que não possuam parênteses, deve-se executar primeiro as funções de produto:

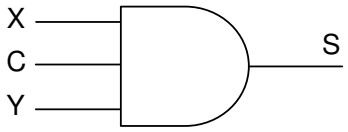
A . B + C = (A . B) + C.

Para exemplificar vamos obter o circuito que executa a expressão

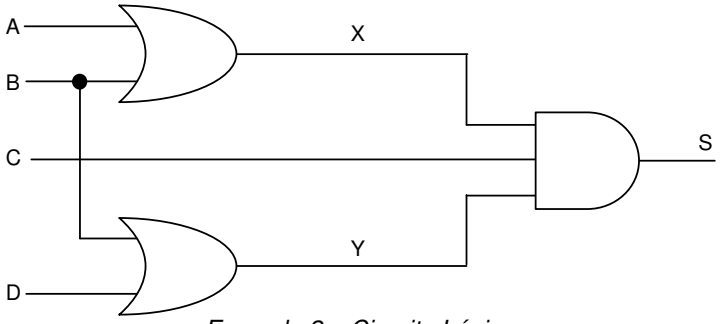
S = (A + B) . C . (B + D).
Para o primeiro parêntese, temos A + B, logo o circuito que o executa será uma porta OU. Para o segundo, temos a soma booleana B + D, logo, o circuito será outra porta OU. Até ai temos então:



A seguir, temos uma multiplicação dos dois parênteses, juntamente com a variável C, sendo executada por uma porta AND:



Substituindo as partes X e Y no bloco acima, obtemos o circuito completo:



Exemplo 2 – Circuito Lógico

Expressões Booleanas Obtidas de Tabelas-verdade
Para demonstrar este procedimento, vamos obter a expressão da tabela abaixo:

A	*B*	*S*
0	0	1
0	1	0
1	0	1
1	1	1

Tabela 06 – Exemplo deTabela Verdade

Observando a tabela, notamos que a expressão é verdadeira (S = 1) nos casos onde A = 0 e B = 0 ou A = 1 e B = 0 ou A = 1 e B = 1. Para obter a expressão, basta montar os termos relativos aos casos onde a expressão for verdadeira e somá-los.

— —

Caso 00: S = 1 quando, A = 0 e B =0 ► $A \cdot B$

Caso 10: S = 1 quando, A = 1 e B = 0 ► $A \cdot \overline{B}$
Caso 11: S = 1 quando, A = 1 e B = 1 ► $A \cdot B$

$S = \overline{A} \cdot \overline{B} + A \cdot \overline{B} + A \cdot B$

Notamos que o método permite obter, qualquer que seja a tabela, uma expressão padrão formada sempre pela soma de produtos. Utilizando a álgebra de Boole é possível simplificar as expressões e obter circuitos mais simplificados.

Porta Lógica XOR (OU-EXCLUSIVO)
A função lógica que esta porta executa, como o próprio nome já diz, consiste em fornecer **1** à saída quando as variáveis de entrada forem diferentes entre si. A partir desta definição, montamos sua tabela-verdade.

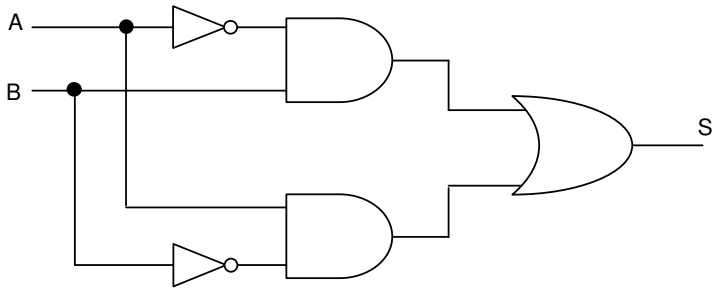
A	*B*	*S*
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 07- Tabela Verdade da Porta XOR

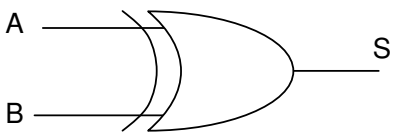
Da tabela obtemos a expressão $S = \overline{A} \cdot B + A \cdot \overline{B}$

Notação Algébrica $S = A \oplus B$

Da expressão esquematizamos o circuito.



Devido a sua definição, o circuito OU Exclusivo só pode ter duas variáveis de entrada e possui um símbolo característico mostrado abaixo:



$S = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$

Figura 13 – Simbologia e Expressão da Função XOR

Porta Lógica XNOR (COINCIDÊNCIA)

Sua função é fornecer **1** à saída quando houver coincidência nos valores das variáveis de entrada.

A	*B*	*S*
0	0	1
0	1	0
1	0	0
1	1	1

Tabela 08- Tabela Verdade da Porta XNOR

Expressão $S = \overline{A} \cdot \overline{B} + A \cdot B$

Notação Algébrica $S = A \odot B$

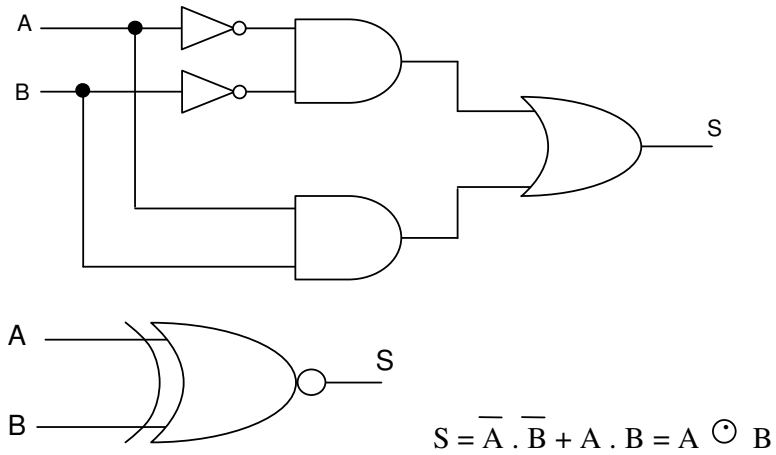


Figura 14 – Circuito, Simbologia e Expressão da Função XNOR

5. Álgebra de Boole

Um circuito lógico pode ser obtido através de uma expressão booleana. No entanto o resultado nem sempre é satisfatório, visto que, as vezes, o circuito resultante pode ser muito complexo ou muito denso.

Para realizarmos a simplificação deste resultado, veremos um resumo da Álgebra de Boole. Pois é através de seus postulados, propriedades, teoremas fundamentais e identidades, que efetuaremos tais simplificações. Estes são apresentados a seguir:

POSTULADOS		
Complementação	***Adição***	***Multiplicação***
$A = 0 \quad \overline{A} = 1$ $A = 1 \quad A = 0$	$0 + 0 = 0$ $0 + 1 = 1$ $1 + 0 = 1$ $1 + 1 = 1$	$0 . 0 = 0$ $0 . 1 = 0$ $1 . 0 = 0$ $1 . 1 = 1$

Tabela 09- Postulados

TEOREMAS DE MORGAN	
1º TEOREMA	$\overline{(A . B)} = \overline{A} + \overline{B}$
2º TEOREMA	$\overline{(A + B)} = \overline{A} . \overline{B}$

Tabela 10 –Teoremas de Morgan

IDENTIDADES		
Complementação	***Adição***	***Multiplicação***
$A = \overline{\overline{A}}$	$A + 0 = A$ $A + 1 = 1$ $A + A = A$ $A + \overline{A} = 1$	$A . 0 = 0$ $A . 1 = A$ $A . A = A$ $A . \overline{A} = 0$
PROPRIEDADES		
COMUTATIVA	Adição $A + B = B + A$	
	Multiplicação $A . B = B . A$	
ASSOCIATIVA	Adição $A + (B + C) = (A + B) + C = A + B + C$	
	Multiplicação $A . (B . C) = (A . B) . C = A . B . C$	
DISTRIBUTIVA	$A . (B + C) = A . B + A . C$	

Tabela 11 – identidades e Propriedades

IDENTIDADES AUXILIARES
$A + A . B = A$
$A + A . \overline{B} = A + B$
$(A + B) . (A + C) = A + B . C$

Tabela 12 – Identidades Auxiliares

6. Mapa de Karnaugh

O mapa de Karnaugh é uma tabela montada de forma a facilitar o processo de minimização das expressões lógicas. Ele é formado por 2n células, onde n é o número de variáveis de entrada. Portanto o mapa de Karnaugh tem tantas células quanto o número de linhas de uma tabela-verdade.

Neste curso, estudaremos somente o mapa para 2 variáveis.

Diagrama de Veitch-Karnaugh para 2 Variáveis

A figura 15 abaixo mostra um diagrama de Veitch-Karnaugh para 2 variáveis:

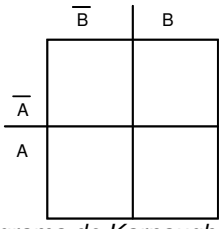
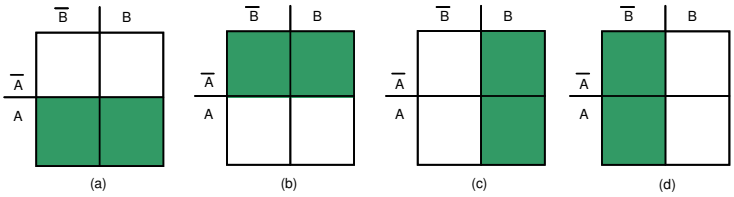


Figura 15 – Diagrama de Karnaugh para 2 variáveis

No mapa, encontramos todas as possibilidades assumidas entre as variáveis A e B. A seguir veja todas as regiões do mapa.

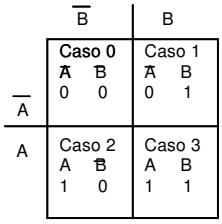


(a) região onde A = 1
(b) região onde A = 0 ($\overline{A}$ = 1)
(c) região onde B = 1.
(d) Região onde B = 0 ($\overline{B}$ = 1).

Com 2 variáveis, podemos obter 4 possibilidades.

A	B	
0	0	caso 0
0	1	caso 1
1	0	caso 2
1	1	caso 3

Podemos distribuir, então, as 4 possibilidades neste diagrama, da seguinte forma:



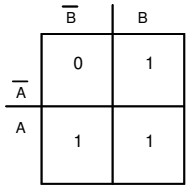
Logo, notamos que cada linha da tabela da verdade possui sua região própria no diagrama de Veitch-Karnaugh.
Essas regiões são, portanto, os locais onde devem ser colocados os valores que a expressão assume nas diferentes possibilidades.
Para entendermos melhor o significado destes conceitos vamos utilizar os exemplos:

1- A tabela da verdade mostra o estudo de uma função de 2 variáveis. Vamos colocar seus resultados no Diagrama de Veitch-Karnaugh.

A	B	S	
0	0	0	caso 0
0	1	1	caso 1
1	0	1	caso 2
1	1	1	caso 3

Levantando a expressão a partir da tabela, obtemos a expressão característica da função:

$S = \overline{A}B + A\overline{B} + AB$
Passando para o mapa os casos da tabela da verdade, temos:

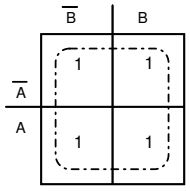


Uma vez entendida a colocação dos valores assumidos pela expressão em cada caso no diagrama de Veitch-Karnaugh, vamos verificar como podemos efetuar as simplificações.

Para obtermos a expressão simplificada do diagrama, utilizamos o seguinte método:
Tentamos agrupar as regiões onde S é igual a 1, no menor número possível de agrupamentos. As regiões onde S é 1, que não puderem ser agrupadas, serão consideradas isoladamente. Para um diagrama de 2 variáveis, os agrupamentos possíveis são os seguintes:

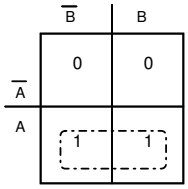
a) Quadra:
Conjunto de 4 regiões, onde S é igual a 1. No diagrama de 2 variáveis, é o agrupamento máximo, proveniente de uma tabela onde todos os casos

valem 1. Assim sendo, a expressão final simplificada obtida é S = 1. A figura abaixo ilustra esta situação.

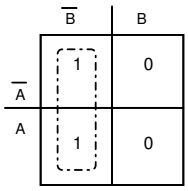


Quadra: S = 1

b) Pares:
Conjunto de 2 regiões onde S é 1, que tem um lado em comum, ou seja, são vizinhos. As figuras abaixo mostram exemplos de 2 pares agrupados e suas respectivas expressões, dentre os 4 possíveis em 2 variáveis:



Par A (está exclusivamente na região A)

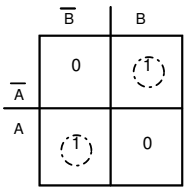


Par $\overline{B}$ (está exclusivamente na região $\overline{B}$)

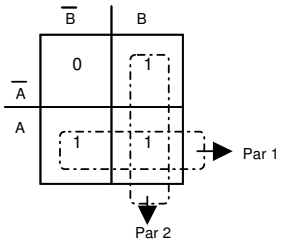
c) Termos isolados:
Regiões onde S é 1, sem vizinhança para grupamentos. São os próprios casos de entrada, sem simplificação. A figura a seguir exemplifica 2 termos isolados, sem possibilidade de agrupamento.

Termo $\overline{A}$ B

Termo A $\overline{B}$



Para o exemplo, efetuando os agrupamentos, temos:



Feito isto, escrevemos a expressão de cada par, ou seja, a região que o par ocupa no diagrama.

O par 1 ocupa a região onde A é igual a 1, então, sua expressão será: Par 1 = A.

O par 2 ocupa a região onde B é igual a 1, então, sua expressão será: Par 2 = B.

Notamos também que nenhum 1 ficou fora dos agrupamentos, e ainda que o mesmo 1 pode pertencer a mais de um agrupamento.

Para obter a expressão simplificada, basta, agora, somarmos os termos obtidos nos agrupamentos.

S = Par 1 + Par 2 S = A + B

Como podemos notar, esta é a expressão de uma porta OU, pois a tabela da verdade também é a da porta OU. Outro fato a ser notado é que a expressão obtida é visivelmente menor do que a extraída diretamente da tabela da verdade, acarretando um circuito mais simples, diminuindo, conseqüentemente, a dificuldade de montagem e o custo do sistema.

2 – Vamos simplificar o circuito que executa a tabela da verdade a seguir:

A	***B***	***S***
0	0	1
0	1	1
1	0	1
1	1	0

Exemplo 03

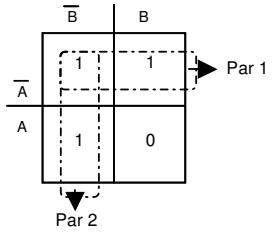
Obtendo a expressão diretamente da tabela, temos:

$S = \overline{A}\,\overline{B} + \overline{A}\,B + A\,\overline{B}$

Transportando a tabela para o diagrama, mediante processo já visto, temos:

	$\overline{B}$	B
$\overline{A}$	1	1
A	1	0

Agora, vamos agrupar os pares:



Vamos escrever as expressões dos pares:

Par 1 -› $\overline{A}$
Par 2 -› $\overline{B}$

Somando as expressões dos pares, temos a expressão simplificada:

$S = \overline{A} + \overline{B}$

Notamos que a tabela-verdade é a de uma porta NE. Aplicando o teorema de Morgan à expressão, após a simplificação, encontramos a expressão de uma porta NE:

$S = \overline{AB}$

7. Circuitos Combinacionais

Um circuito combinacional executa eletronicamente uma função booleana através da interligação de portas lógicas.

Neste capítulo, vamos estudar diversos circuitos combinacionais que são utilizados constantemente em projetos de sistemas digitais devido às funções lógicas que executam, tanto é que eles são encontrados já prontos em circuitos integrados comerciais e, por isso, são chamados de *circuitos combinacionais dedicados.*

Codificadores e Decodificadores

Uma grande parte dos sistemas digitais trabalha com níveis lógicos representando informações que, portanto, devem ser *codificadas*.

Exemplos:

- A calculadora trabalha com informações numéricas;
- O computador trabalha com informações alfanuméricas;
- O sistema de telefonia digital trabalha com canais de voz;
- O Disco Laser trabalha com sinais sonoros.

Estes exemplos correspondem a sistemas digitais que, na realidade, não entendem números, letras, canais de voz ou sinais sonoros, mas sim, *códigos binários* que possuem apenas dois níveis lógicos para representar qualquer tipo de informação.

Devido à grande diversidade de informações e ao desenvolvimento da eletrônica digital, vários códigos foram criados e, conseqüentemente, vários circuitos se fizeram necessários para a *codificação* e *decodificação* destas informações.

Código BCD 8421

O *código BCD 8421* ou, simplesmente, **BCD** (Binary Coded Decimal) que significa Decimal Codificado em Binário, é um dos mais comuns nos sistemas digitais.

Ele é composto por quatro bits, tendo cada bit um peso equivalente ao do sistema numérico binário, ou seja, 1 para o primeiro bit à direita que é chamado de bit menos significativo (LSB – Least Significant Bit), 2 para o segundo bit, 4 para o terceiro bit e 8 para o quarto bit que é chamado de bit mais significativo (MSB – Most Significant Bit).

Desta forma, este código representa os números decimais de 0 a 9 no sistema binário, como mostra a tabela 13.

Decimal	BCD			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Tabela 13 – Tabela BCD

Assim, ao invés de se converter um número formado por diversos dígitos para o sistema binário, os sistemas digitais que trabalham com este código fazem a conversão de cada dígito do número para o código BCD.

Exemplo:

O número 2538 no sistema decimal pode ser representado das seguintes formas:

- No sistema binário: $(100111101010)_2$
- No código BCD: (0010 0101 0011 1000)

Código Excesso 3

O código excesso 3 é composto por quatro bits que correspondem aos números decimais no código BCD acrescidos de 3 (0011), por isso o seu nome. Ele foi criado para facilitar as operações de subtração no sistema binário devido ao fato do complemento de um algarismo do sistema decimal corresponder ao complemento bit a bit do código excesso 3, como mostra a tabela abaixo:

Decimal	EXCESSO 3			
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Tabela 14 – Tabela Excesso 3

Exemplo:

- No sistema decimal: complemento de 6 é 3
- No código excesso 3: complemento de bit a bit de 1001 é 0110

Obs.:

Atualmente, com o desenvolvimento dos circuitos integrados que executam operações lógicas e aritméticas, principalmente os microprocessadores que internamente também, contêm estes circuitos, o código excesso 3 deixou de ter grande aplicação prática.

Código 9876543210

Este código de 10 bits foi bastante utilizado na época em que os sistemas mostradores de algarismos eram válvulas eletrônicas (Nixie e Numitron). Algumas dessas válvulas possuíam cada algarismo composto por uma placa ou filamento, arranjado apropriadamente no formato do número.

Notamos no código, que em 10 saídas somente uma vale 1 em cada caso, acendendo assim o algarismo correspondente. A formação deste código é vista na tabela a seguir.

Decimal	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0

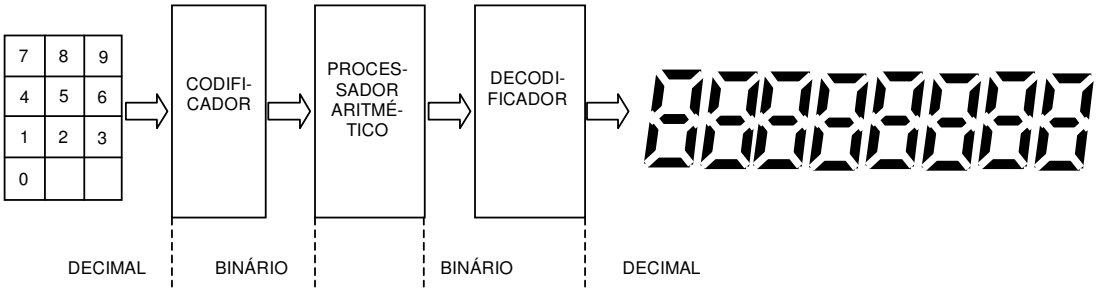
Tabela 15 – Tabela 9876543210

Codificadores e Decodificadores

Vamos, agora, tratar de circuitos que efetuam a passagem de um determinado código para outro. Primeiramente, vamos fazer uma análise do significado das palavras codificador e decodificador.

Chamamos de codificador o circuito combinacional que torna possível a passagem de um código conhecido para um desconhecido. Como exemplo, podemos citar o circuito inicial de uma calculadora que transforma uma entrada decimal, através do sistema de chaves de um teclado, em saída binária para que o circuito interno processe e faça a operação.

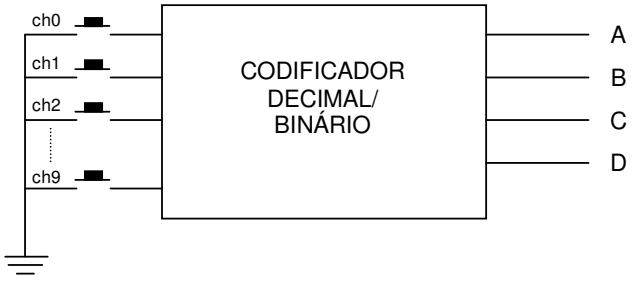
Chamamos de decodificador o circuito que faz o inverso do codificador, ou seja, passa um código desconhecido para um conhecido. No exemplo citado é o circuito que recebe o resultado da operação em binário e o transforma em saída decimal, na forma compatível para um mostrador digital apresentar os algarismos. A figura ilustra o exemplo utilizado.



Os termos codificador e decodificador, porém, diferenciam-se em função do referencial. Se para o usuário da calculadora o sistema de entrada é um codificador, para o processador será um decodificador, pois passa de um código desconhecido para ele (decimal), para um conhecido (binário). Na prática, é comum se utilizar a denominação de decodificador para o sistema que passa de um código para outro, quaisquer que sejam.

Codificador Decimal/Binário

Vamos, neste item, elaborar um codificador para transformar um código decimal em binário (BCD8421). A entrada do código decimal vai ser feita através de um conjunto de chaves numeradas de 0 a 9 e a saída por 4 fios, para fornecer um código binário de 4 bits, correspondente à chave acionada. A figura a seguir mostra a estrutura geral deste sistema, sendo convencionado que a chave fechada equivale a nível 0, para evitar o problema prático, principalmente da família TTL, que um terminal de entrada em vazio é equivalente a nível lógico 1.



A seguir, vamos construir a tabela da verdade do codificador que relaciona cada chave de entrada decimal com a respectiva saída em binário:

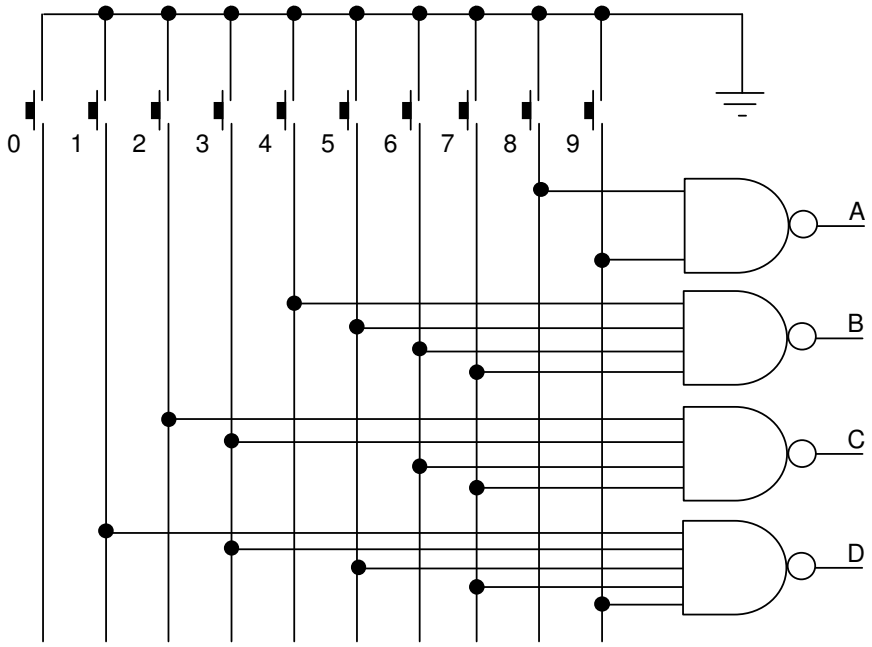
Chave	*A*	*B*	*C*	*D*
Ch0	0	0	0	0
Ch1	0	0	0	1
Ch2	0	0	1	0
Ch3	0	0	1	1
Ch4	0	1	0	0
Ch5	0	1	0	1
Ch6	0	1	1	0
Ch7	0	1	1	1
Ch8	1	0	0	0
Ch9	1	0	0	1

Através da tabela, concluímos que a saída A valerá 1 quando Ch8 ou Ch9 for acionada. A saída B quando Ch4, Ch5, Ch6 ou Ch7 for acionada. A saída C quando Ch2, Ch3, Ch6 ou Ch7 for acionada. A saída D quando Ch1, Ch3, Ch5, Ch7 ou Ch9 for acionada.

Usaremos para a construção do circuito, uma porta NE em cada saída, pois esta fornece nível 1 quando qualquer uma de suas entradas assumir nível 0, situação compatível com a convenção adotada para o conjunto de chaves. A ligação das entradas de cada porta será feita, conforme a análise efetuada, às chaves responsáveis pelos níveis 1 de cada saída.

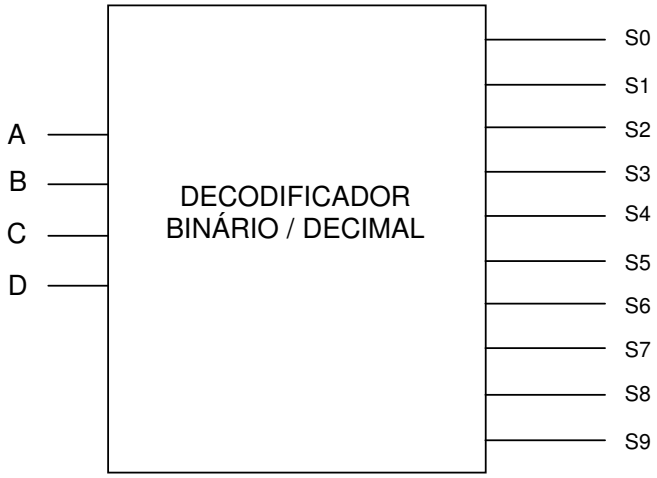
O circuito, assim constituído, é visto na página seguinte.

Pela figura, notamos que a chave Ch0 não está ligada a nenhuma das entradas das portas, sendo irrelevante o seu acionamento, pois a saída também será igual a 0 (A = B = C = D = 0) quando nenhuma das chaves for acionada.



Decodificador Binário/Decimal

A estrutura geral deste decodificador é vista na figura abaixo:



Vamos montar a tabela da verdade do circuito no qual as entradas são bits do código BCD 8421 e as saídas são os respectivos bits do código decimal 9876543210.

BCD 8421				***Código 9876543210***									
A	***B***	***C***	***D***	***S9***	***S8***	***S7***	***S6***	***S5***	***S4***	***S3***	***S2***	***S1***	***S0***
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

O código BCD 8421 não possui números maiores que 9, logo, tanto faz o valor assumido nas possibilidades excedentes, visto que, quando passarmos do código BCD 8421 para o código 9876543210 estas não irão ocorrer.

Decodificador para Display de 7 Segmentos

O *display de 7 segmentos* possibilita escrevermos números decimais de 0 a 9 e alguns outros símbolos que podem ser letras ou sinais. A figura a seguir representa uma unidade do display genérica, com a nomenclatura de identificação dos segmentos usual em manuais práticos.

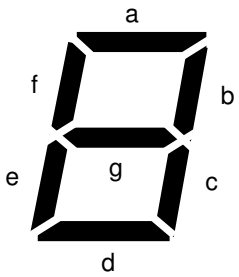


Figura 16 – Display de 7 segmentos

Entre as tecnologias de fabricação das unidades de display usaremos o mais comum que é o display a led, que possui cada segmento composto por um led, existindo um tipo denominado *catodo comum* e outro *anodo comum*.

O display tipo catodo comum é aquele que possui todos os catodos dos led's interligados, sendo necessário aplicar nível 1 no anodo respectivo para acender cada segmento. Já o de anodo comum possui todos os anodos interligados, sendo preciso aplicar nível 0 ao catodo respectivo.

Vamos a título de exemplo, elaborar um decodificador para a partir de um código binário (BCD 8421) escrever a seqüência de 0 a 9 em um display de 7 segmentos catodo comum. O esquema geral deste decodificador é visto na figura abaixo:

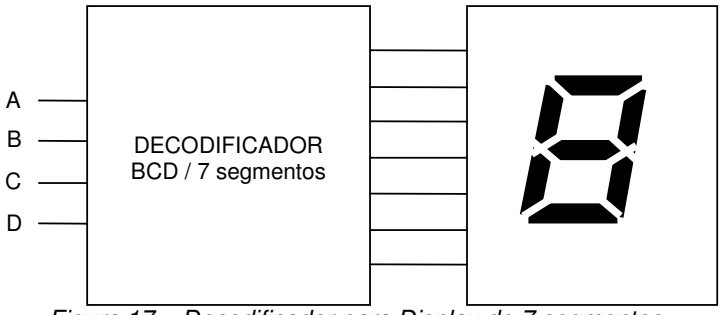


Figura 17 – Decodificador para Display de 7 segmentos

Para efetuar o projeto deste decodificador, devemos verificar em cada caractere os segmentos que devem ser acessos e atribuir o nível 1 (no caso do catodo comum), em função da respectiva entrada no código binário. A tabela a seguir apresenta a seqüência de caracteres, o respectivo código de entrada, e os níveis aplicados em cada segmento para que tal ocorra.

Carac-	BCD 8421				Código para 7 segmentos						
teres	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Para fins de simplificação, vamos considerar os casos fora da seqüência como irrelevantes. Transpondo as saídas para os diagramas de Karnaugh, temos após simplificação:

$a = A + C + B \odot D$

$b = \overline{B} + D \odot D$

$c = B + \overline{C} + D$

$d = A + \overline{B}\,\overline{D} + \overline{B}\,C + C\,\overline{D} + B\,\overline{C}\,D$

$e = \overline{B}\,\overline{D} + C\,\overline{D}$

$f = A + \overline{C}\,\overline{D} + B\,\overline{C} + B\,\overline{D}$

$g = A + B \oplus C + C\,\overline{D}$

O circuito do decodificador BCD 8421 para display de 7 segmentos obtido, é visto na figura 18 abaixo:

Convém observar que o circuito poderia ser otimizado, pois as expressões dos segmentos possuem vários termos em comum, resultando no emprego de um menor número de portas. Porém, para melhor clareza didática, este foi deixado na sua forma original de acordo com as expressões extraídas dos diagramas.

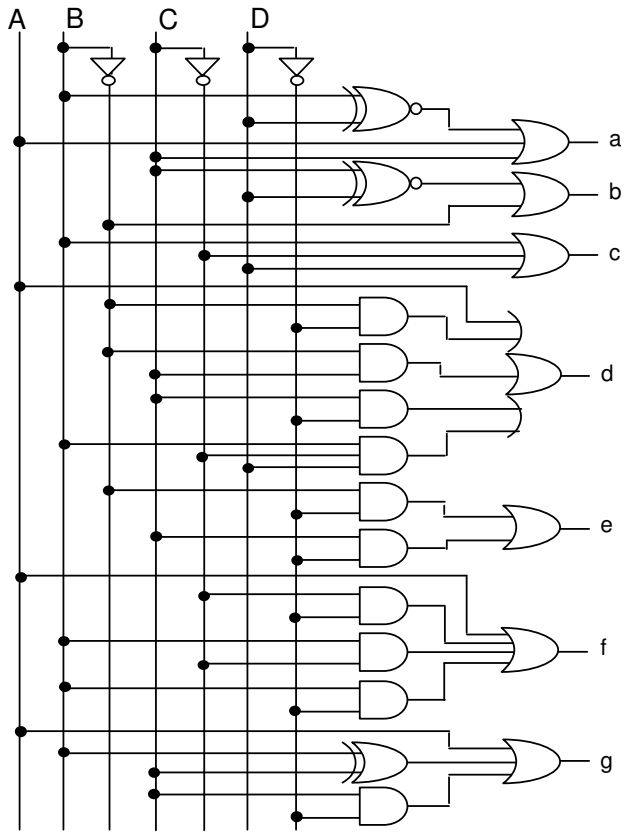


Figura 18 – Circuito Equivalente do Decodificador para Display de 7 segmentos

Um outro ponto a ser realçado é que numa montagem prática, a ligação do display se faz, conforme a família lógica, através de resistores para observar os limites máximos de corrente nos led's, ou ainda, utilizando outras estratégias para controlar o brilho, como por exemplo, blocos open-collector.

Os displays de 7 segmentos podem ainda escrever outros caracteres, que são freqüentemente utilizados em sistemas digitais para representar outras funções, bem como formar palavras-chave em software de programação.

Para efetuar o projeto, basta verificar caso a caso quais segmentos devem acender e montar assim a tabela da verdade.

Circuitos Aritméticos

Dentro do conjunto de circuitos combinacionais aplicados para finalidade específica nos sistemas digitais, destacam-se os circuitos aritméticos. São utilizados, principalmente, para construir a ULA (Unidade Lógica Aritmética) dos microprocessadores e, ainda, encontrados disponíveis em circuitos integrados comerciais.

8. Famílias Lógicas

Entende-se por famílias lógicas, os tipos de estruturas internas que nos permitem a confecção dos blocos lógicos em circuitos integrados. Cada família lógica utiliza determinados componentes em seus blocos e, de acordo com estes, a família possuirá determinadas características relacionadas ao seu funcionamento e desempenho prático.

As famílias utilizadas atualmente dentro da área eletrônica digital são a TTL – Transistor-Transistor Logic e a CMOS – Complementary Metal Oxide Semicondutor, porém derivam de uma série de famílias lógicas, hoje obsoletas.

Algumas famílias utilizadas anteriormente:

DCTL	=	Direct-Coupled Transistor Logic
RTL	=	Resistor-Transistor Logic
RCTL	=	Resistor-Capacitor Transistor Logic
DTL	=	Diode-Transistor Logic
HTL	=	High-Threshold Logic
ECL	=	Emitter-Coupled Logic

Vamos estudar somente as famílias CMOS e TTL, devido a tecnologia atual.

Família TTL

Os circuitos TTL são produzidos em duas séries comerciais: a série 74XX e a 54XXX, sendo esta última denominada série militar ou profissional, devido à maior margem de variação nas especificações de alimentação e temperatura, assegurando a confiabilidade no desempenho em condições máximas**.**

Alimentação: temos para todos os blocos uma alimentação de 5V. Para a série 54 temos Vcc mínimo = 4,5V e Vcc máximo = 5,5V que são valores dentro da especificação militar de 10% de tolerância. Para a série 74, temos Vcc mínimo de 4,75V e Vcc máximo =5,25v que são valores dentro da especificação comum de 5% de tolerância.

Fan-out: Na versão padrão, o Fan-out é igual a 10, ou seja, pode-se ligar à saída destes blocos, no máximo outros 10 blocos.

Tempo de atraso de propagação: Em média de 10ns.

Imunidade ao ruído: De maneira geral é igual a 0,4V, e é considerada baixa em relação a CMOS

Potência Dissipada: É da ordem de 10mW por porta.

Tipos: Podemos destacar os blocos open-collector, tri-state e schimitt-trigger.

Família CMOS

Os circuitos CMOS são construídos por transistores MOS-FET complementares do tipo canal N e canal P. Suas configurações básicas permitem obter-se uma série de vantagens, tais como: alto Fan-Out, alta margem de imunidade ao ruído e baixíssimo consumo, sendo esta uma das principais características.

Alimentação: temos para as séries 4000 e 74C, a faixa de 3V a 15V, para a versão HC, a faixa de 2V a 6V e para a HCT de 4,5V a 5,5V. Para as séries de baixa voltagem, a faixa de 1V a 3,6V para a LV, e 1,2V a 3,6V para a LVC.

Fan-out: Na versão padrão, o Fan-out é igual a 50, porém varia conforme as versões empregadas.

Tempo de atraso de propagação: Em média de 90ns.

Imunidade ao ruído: De maneira geral é igual 45% de Vdd (tensão de alimentação)

Potência Dissipada: É da ordem de 1nW por porta da série 4000 e 2,5nW ba série 74HC, a uma tensão de alimentação de 5V.

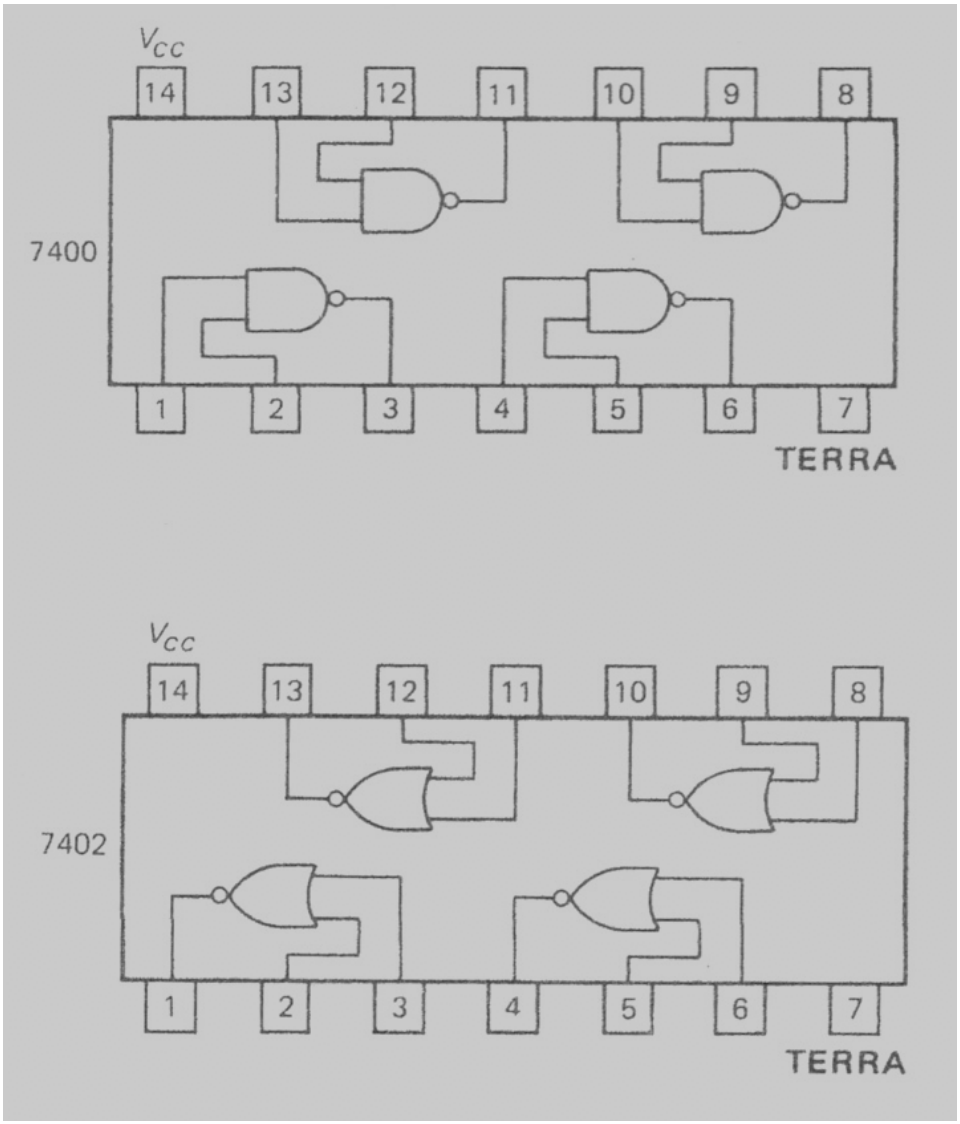
Manuseio: Esta família necessita, ao contrário da TTL, um cuidado extra no manuseio dos circuitos integrados, que devido à eletricidade estática, provoca a degradação das junções internas dos chips, comprometendo sua vida útil.

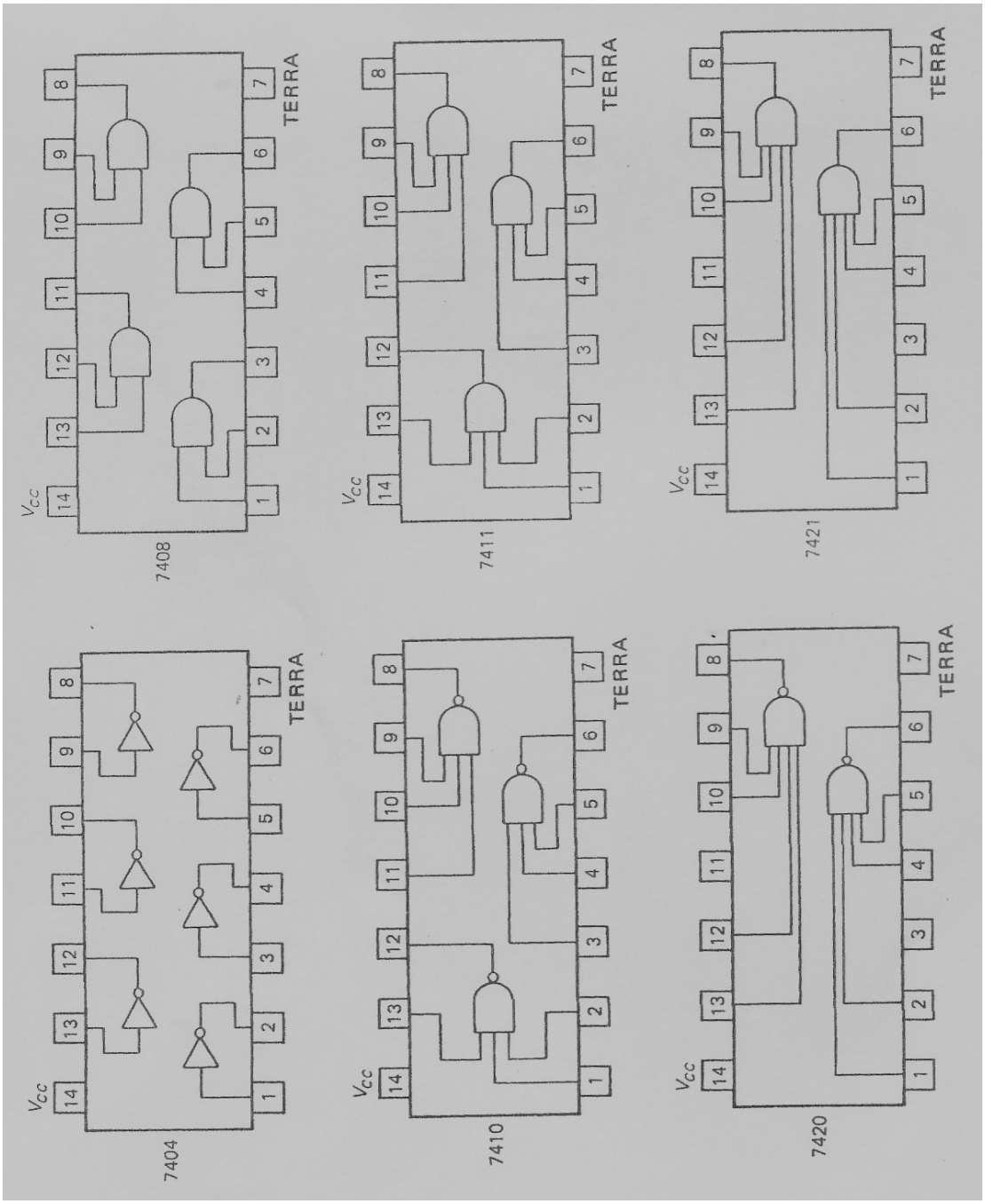
9. Dispositivos TTL

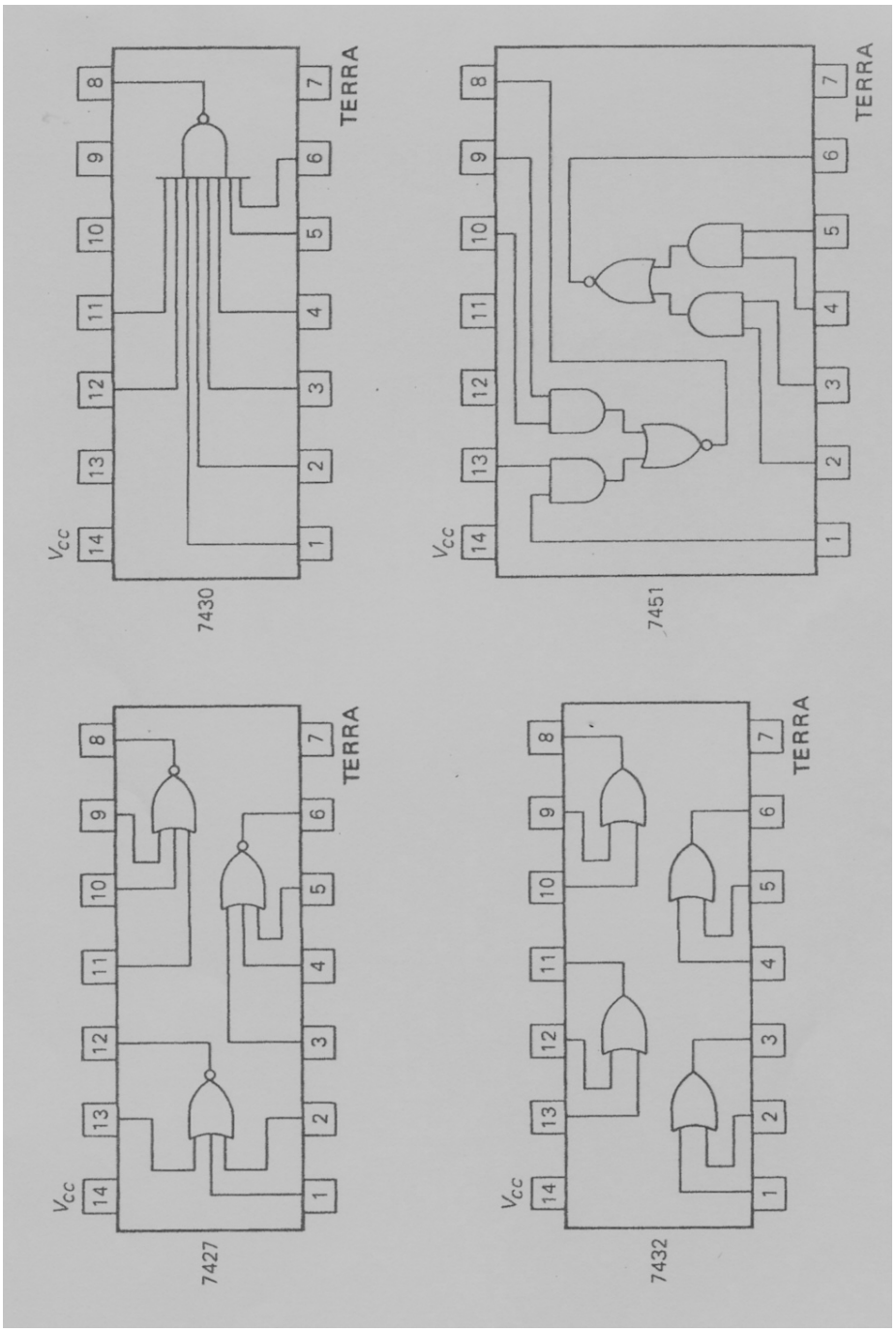
Número	Função
7400	Quatro portas NAND de 2 entradas
7401	Quatro portas NAND de 2 entradas (coletor aberto)
7402	Quatro portas NOR de 2 entradas
7403	Quatro portas NOR de 2 entradas (coletor aberto)
7404	Seis inversores
7405	Seis inversores (coletor aberto)
7406	Seis acionadores buffer inversores
7407	Seis acionadores buffer
7408	Quatro portas AND de 2 entradas
7409	Quatro portas AND de 2 entradas (coletor aberto)
7410	Três portas NAND de 3 entradas
7411	Três portas AND de 3 entradas
7412	Três portas NAND de 3 entradas (coletor aberto)
7413	Dois limitadores Schmitt
7414	Seis limitadores Schmitt
7416	Seis acionadores buffer inversores
7417	Seis acionadores buffer
7420	Duas portas NAND de 4 entradas
7421	Duas portas AND de 4 entradas
7422	Duas portas NAND de 4 entradas (coletor aberto)
7423	Duas portas NOR expandíveis de4 entradas
7425	Duas portas NOR de 4 entradas
7426	Quatro portas NAND de 2 entradas para interface TTL – CMOS
7427	Três portas NOR de 3 entradas
7428	Quatro buffers NOR de 2 entradas
7430	Porta NAND de 8 entradas
7432	Quatro portas OR de 2 entradas
7437	Quatro buffers NAND de 2 entradas
7438	Quatro buffers NAND de 2 entradas
7439	Quatro buffers NAND de 2 entradas(coletor aberto)
7440	Dois buffers NAND de 4 entradas
7441	Acionador Nixie decodificador BCD - para - decimal
7442	Decodificador BCD - para - decimal
7443	Decodificador excesso 3 - para - decimal
7444	Excesso Gray – para –decimal
7445	Acionador – decodificador BCD - para - decimal
7446	Acionadores – decodificadores BCD – para –7 segmentos (saída 30 v)
7447	Acionadores – decodificadores BCD – para –7 segmentos (saída 15 v)
7448	Acionadores – decodificadores BCD – para –7 segmentos
7450	Duas portas AND-OR-INVERSOR expandíveis de 2 entradas 2 amplas
7451	Duas portas AND-OR-INVERSOR de 2 entradas 2 amplas
7452	Portas AND-OR expandíveis de 2 entradas 4 amplas
7453	Duas portas AND-OR-INVERSOR expandíveis de 2 entradas
7454	Portas AND-OR-INVERSOR de 2 entradas 4 amplas

74153	Dois multoplexadores 4/1
7455	Duas portas AND-OR-INVERSOR expandíveis de 4 entradas 2 amplas
7459	Duas portas AND-OR-INVERSOR de 2-3 entradas 2 amplas
7460	Dois expansores de 4 entradas
7461	Três expansores de 3 entradas
7462	Expansores 2-2-3-3- entradas 4 amplas
7464	Portas AND-OR-INVERSOR de 2-2-3-4 entradas 4 amplas
7465	Portas AND-OR-INVERSOR de 4 amplas (coletor aberto)
7470	Flip-flop JK disparado pela borda
7472	Flip-flop JK mestre-escravo
7473	Dois flip-flops JK mestre-escravo
7474	Dois flip-flops D
7475	Quatro latches
7476	Dois flip-flops JK mestre-escravo
7480	Portas totalizadoras
7482	Totalizador binário de 2 bits
7483	Totalizador binário de 4 bits
7485	Comparador de magnitude de 4 bits
7486	Quatroportas OR- Exclusivo
7489	Memória de leitura escrita de acesso aleatório de 64 bits
7490	Contador de décadas
7491	Registrador de deslocamento de 8 bits
7492	Contador divisor-por-12
7493	Contador binário de 4 bits
7494	Registrador de deslocamento de 4 bits
7495	Registrador de deslocamento direita e deslocamento esquerda de 4 bits
7496	Registrador de deslocamento entrada paralela-saída paralela de 5 bits
74100	Latch biestável de 4 bits
74104	Fflip-flops JK mestre-escravo
74105	Flip-flops JK mestre-escravo
74107	Dois flip-flops JK mestre-escravo
74109	Dois flip-flops JK disparados pela borda positiva
74116	Dois latches de 4 bits com limpar
74121	Multivibrador monoestável
74122	Multivibrador monoestável com limpar
74123	Multivibrador monoestável
74125	Quatro buffers de via de três-estados
74126	Quatro buffers de via de três-estados
74132	Quatro limitadores Schmitt
74136	Quatro portas OR- Exclusivo de 2 entradas
74141	Decodificador-acionador BCD-para-decimal
74142	Contador-acionador latch BCD
74145	Decodificador-acionador BCD-para-decimal
74147	Codificador de prioridade 10/4
74148	Codificador de prioridade
74150	Multiplexador 16-linhas-para-1-linha
74151	Multiplexador digital de 8 canais

74152	Multiplexador seletor de dados de 8 canais
74190	Contador de décadas reversível
74154	Demultiplexador decodificador 4-linhas-para-16-linhas
74155	Dois demultiplexadores 2/4
74156	Dois demultiplexadores 2/4
74157	Quatro seletores de dados 2/1
74160	Contador de décadas com limpar assíncrono
74161	Contador de 4 bits síncrono
74162	Contador de 4 bits síncrono
74163	Contador de 4 bits síncrono
74164	Registrador de deslocamento serial de 8 bits
74165	Registrador de deslocamento serial de 8 bits carga paralela
74166	Registrador de deslocamento de 8 bits
74173	Registrador de três estados de 4 bits
74174	Seis flip-flops F com limpar
74175	Quatro flip-flops D com limpar
74176	Contador de décadas preestabelecível de 35 MHz
74177	Contador binário preestabelecível de 35 MHz
74179	Registrador de deslocamento de acesso paralelo de 4 bits
74180	Verificador gerador de paridade ímpar-par de 8 bits
74181	Unidade lógica aritmética
74182	Gerador de transporte de “olhar para frente”
74184	Conversor BCD-para-binário
74185	Conversor binário -para-BCD
74189	Memória de acesso aleatório de 64 bits de três-estados
74191	Contador binário reversível síncrono
74192	Contador binário reversível
74193	Contador binário reversível
74194	Registrador de deslocamento direcional de 4 bits
74195	Registrador de deslocamento de acesso paralelo de 4 bits
74196	Contador de décadas preestabelecível
74197	Contador binário preestabelecível
74198	Registrador de deslocamento de 8 bits
74199	Registrador de deslocamento de 8 bits
74221	Dois limitadores Schmitt monoestáveis
74251	Multiplexador de 8 canais de três-estados
74259	Latch endereçável de 8 bits
74276	Quatro flip-flops JK
74279	Quatro eliminadores de ricochete
74283	Totalizador binário de 4 bits co transporte rápido
74284	Multiplexador de 4 bits, de três-estados
74285	Multiplexador de 4 bits, de três-estados
74365	Seis buffers de três estados
74366	Seis buffers de três estados
74367	Seis buffers de três estados
74368	Seis buffers de três estados
74390	Relógios individuais com flip-flops
74393	Dois contadores binários de 4 bits


V_{CC}
14
13
12
11
10
9
8
7400
1
2
3
4
5
6
7
TERRA
V_{CC}
14
13
12
11
10
9
8
7402
1
2
3
4
5
6
7
TERRA


7408
7411
7421
7404
7410
7420
TERRA
V_{CC}


7430
7451
7427
7432
TERRA
V_{CC}

10. Microcontroladores e Microprocessadores

Microcontroladores X Microprocessadores
Os microcontroladores diferem dos microprocessadores em muitos aspectos. O Primeiro e o mais importante é sua funcionalidade. Em um sistema com microprocessador outros dispositivos tais como: memória ou componentes para comunicação serial ou paralela, devem ser adicionados ao sistema.

Em outras palavras, o microprocessador é o coração do sistema e possui uma capacidade de processamento maior, visto que, as tarefas atribuídas aos periféricos são realizadas por outros dispositivos.

8051
O 8051, da Intel, é, sem dúvida, o microcontrolador mais popular atualmente. O dispositivo em si é um microcontrolador de 8 bits relativamente simples, mas com ampla aplicação. Porém, o mais importante é que não existe somente o CI 8051, mas sim uma família de microcontroladores baseada no mesmo. Entende-se família como sendo um conjunto de dispositivos que compartilha os mesmos elementos básicos, tendo também um mesmo conjunto básico de instruções.

Unidade de memória
A memória é a parte do microcontrolador que possui a função de armazenar os dados.

A maneira mais fácil de entender como ela funciona é imaginar a memória como um armário com várias gavetas contendo alguns objetos. Se marcarmos cada gaveta de maneira que elas não possam ser confundidas, então, cada uma pode ser facilmente acessível. Isso é o suficiente para sabermos o destino de cada objeto e o conteúdo de cada gaveta.

Unidade Central de Processamento
Esse bloco possui a capacidade de multiplicar, dividir, somar, e subtrair o conteúdo desses registradores, bem como mover o conteúdo desses registradores para outras posições de memória. Esse bloco é chamado de CPU "Central Processing Unit" ou UCP "Unidade Central de Processamento".

Registradores
Registradores são, portanto, posições de memória cuja função é auxiliar a CPU na realização de várias operações matemáticas ou qualquer outra operação que envolva transferência de dados. Vamos ver então a situação em que nos encontramos. Temos dois blocos independentes (memória e CPU) os quais estão interligados e qualquer transferência de dados de um bloco para outro deverá ser realizada sob o comando da CPU. Se, por exemplo, quisermos somar o conteúdo de duas posições de memória e retornar o resultado para uma outra posição de memória, nós precisamos de uma conexão entre a memória e a CPU. É

necessário então um caminho físico para que os dados possam fluir da CPU para memória e vice-versa.

Aspecto físico do interior de um microcontrolador
Na parte central do chip está localizado o substrato semicondutor e integrado nesse substrato estão todos os blocos descritos anteriormente. Os fios que saem da parte central para os pinos ligam os blocos ao mundo externo. O diagrama de blocos a seguir representa a parte central do chip.

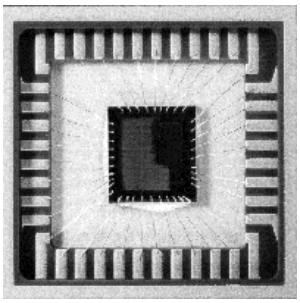


Figura 19 – Aspecto físico do interior de um microcontrolador

Barramento
Existem dois tipos de barramento: barramento de dados e barramento de endereços.

O barramento de endereço consiste de uma quantidade de linhas correspondente ao tamanho da memória que queremos endereçar. Esse barramento é utilizado para informar a memória qual a posição de memória será acessado em um ciclo de leitura ou escrita.

O barramento de dados consiste de uma quantidade de linhas correspondente ao tamanho ou magnitude do dado a ser manipulado pela CPU. Em nosso estudo estaremos trabalhando com microcontroladores de 8-bits, portanto, teremos um barramento de dados de 8-bits ou 8 linhas de conexão para dados. Esse barramento conecta todos os blocos dentro do microcontrolador.

Unidade de Entrada e Saída I/O
Chamamos esse bloco de portas. Existem vários tipos de portas: entrada, saída ou porta bidirecional. Quando trabalhamos com as portas, primeiro precisamos configurar o tipo de porta que queremos trabalhar. Depois escrevemos ou lemos os dados dependendo do tipo de porta selecionado.

As operações com as portas funcionam como simples operação de leitura ou escrita em uma posição de memória específica. Uma simples operação de escrita na porta levará o dado correspondente para os pinos externos do C.I.

Microprocessador

Essencialmente, o microprocessador é o computador, e mesmo chamado como "um computador num chip". Ele faz todos os cálculos, a computação e a compilação. Todas as outras partes do computador só servem para ajudar o microprocessador a realizar as tarefas identificadas pelo usuário.

Os microcontroladores são projetados para realizarem todos as funções em um único chip. Nenhum componente externo é necessário para sua aplicação porque todos os periféricos estão dentro do microcontrolador. Assim, uma vantagem obvia dos microcontroladores é o ganho de espaço físico no projeto de hardware.

Cada tipo de microprocessador tem um conjunto único de instruções. Programas escritos em assembler, a linguagem de programação mais básica além do binário, é específica para cada microprocessador. Um programa escrito em assembler para um Intel 486 não vai ser executável por um Motorola 68040.

Tipos de arquitetura de microprocessador.

RISC - Processadores baseados em RISC (Reduced Instruction Set Computing / Computação com conjunto de instruções reduzido) não têm microprogramas. Instruções são implementadas diretamente no hardware. Qualquer tarefa complexa demais para ser executada num ciclo é feita por uma série de instruções básicas. A arquitetura RISC é especialmente efetiva para tarefas simples como carregar, armazenar e ramificar. A arquitetura simples de um RISC permite que instruções simples sejam processadas através do processador rapidamente e facilmente. Os chips de arquitetura RISC podem ser desenvolvidos mais rapidamente que aqueles de arquitetura CISC por causa de sua simplicidade.

CISC - A arquitetura CISC (Complex Instruction Set Computing / Computação com conjunto de instruções complexo) é o predecessor do RISC. A arquitetura CISC não pode implementar instruções de hardware. As instruções precisam ser traduzidas por um microprograma escondido para as verdadeiras instruções do hardware. O CISC tem um conjunto de instruções completo para ser capaz de completar todos as tarefas de uma maneira mais eficiente.

Referências Bibliográficas

SENAI. MG. *Eletrônica Digital*

IDOETA, Ivan Valeije, CAPUANO, Francisco Gabriel. *Elementos de Eletrônica Digital*. 34ª Edição. São Paulo. Ed. Érica. 2002.

LOURENÇO, Antônio Carlos, CRUZ, Eduardo C. A.. *Circuitos Digitais - Estude e Use*. 5ª Edição. São Paulo. Ed. Érica. 2002.

TOCCI, Ronald J., LASKOWSKI,Lester P.. *Microprocessadores e Microcomputadores*. 2ª Edição. Rio de Janeiro. Ed. Prentice-Hall do Brasil. 1983.

http://www.i-magazine.com.br/imagazine/picbook/cap1.htm em 15/07/2004.