



**Escola de Engenharia de São Carlos**  
Dep<sup>to.</sup> de Eng. Materiais, Aeronáutica e  
Automobilística

Notas de aula do Prof. Dr. Luís Carlos Passarini

# SMM-166 Eletrônica Aplicada aos Motores CI

1

## Fundamentos da Eletrônica Digital



2

# Introdução

- A palavra **digital** deriva de “*dígito*”, que por sua vez procede do latim “*digitus*”, significando “*dedo*”.
- Desde que a humanidade desenvolveu o processo de contagem, os dedos foram os instrumentos mais simples e eficientes para contar pequenos valores.
- Com os dedos só é possível contar valores inteiros (0 a 10).
- Por causa dessa característica, a palavra *digital* também é usada para se referir a qualquer dispositivo que trabalha com valores discretos (estados).

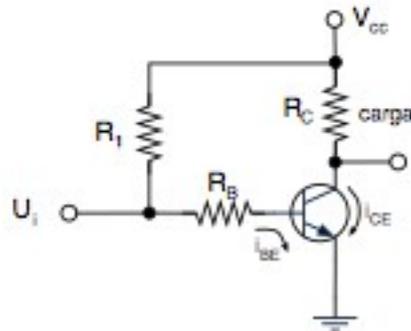
3

- Os circuitos digitais, incluindo os computadores digitais, são formados a partir de *circuitos binários*.
- Circuitos binários são circuitos cuja resposta pode ser apenas uma de dois *estados* diferentes.
- Cada estado é indicado por um determinado nível de corrente ou de tensão.
- exemplo: luz de cortesia (ligada ou desligada).



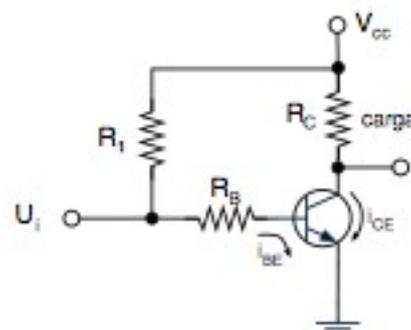
4

- Nos sistemas digitais eletrônicos o transistor bipolar é usado como *chave*. Para isto, devemos lembrar que o transistor possui 3 regiões de operação: corte, proporcional e saturado. Para operarmos o transistor como chave precisamos fazê-lo operar apenas nas regiões de corte e saturado. Quando o transistor está saturado, este apresenta resistência muito baixa e estado *ON*. Quando o transistor está em corte, sua resistência é muito alta e seu estado é *OFF*.



5

- Quem determina em que região o transistor vai operar é o nível de tensão presente em sua base.
- Em circuitos digitais, o nível de tensão na entrada do transistor-chave deve ser capaz de ou saturá-lo ou cortá-lo sem permitir sua operação na região proporcional.



6

# Os Sistemas Numéricos

- Diversos sistemas numéricos diferentes são utilizados no mundo todo (romano, decimal/métrico, duodecimal/inglês, ...).
- Contudo, o *sistema binário* predomina nos equipamentos digitais.
- O aspecto mais notável do sistema binário é a simplicidade: tudo é expresso em função de “zeros” e “uns”.

7

- O Sistema Decimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  - É um sistema que o homem adotou naturalmente em virtude de termos 10 dedos nas mãos, geralmente.
  - Os números são expressos como combinações de potências de 10, como nos exemplos:

$$\begin{aligned}1984_{10} &= 1000 + 900 + 80 + 4 \\ &= 1 \cdot 10^3 + 9 \cdot 10^2 + 8 \cdot 10^1 + 4 \cdot 10^0\end{aligned}$$

$$\begin{aligned}19,84_{10} &= 10 + 9 + 0,8 + 0,04 \\ &= 1 \cdot 10^1 + 9 \cdot 10^0 + 8 \cdot 10^{-1} + 4 \cdot 10^{-2}\end{aligned}$$

8

- Um *deslocamento para a esquerda* equivale a uma multiplicação por 10, enquanto que um *deslocamento para a direita* equivale a uma divisão por 10.

- Exemplo:

$$\overleftarrow{2004} = 20040$$

$$\overrightarrow{3017} = 301,7$$

9

- O Sistema Binário (0, 1)

- Este sistema simplificou os projetos dos circuitos eletrônicos e aumentou a segurança.

- Os números são expressos como combinações de potências de 2, como nos exemplos:

$$\begin{aligned} 10101_2 &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \\ &= 16 + 4 + 1 = 21_{10} \end{aligned}$$

$$\begin{aligned} 1,1101_2 &= 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = \\ &= 1 + 0,5 + 0,25 + 0,00625 = 1,8125_{10} \end{aligned}$$

10

- Um deslocamento para a esquerda equivale a uma multiplicação por 2, enquanto que um deslocamento para a direita equivale a uma divisão por 2.

Exemplo:  $\overleftarrow{11,101}_2 = 111,01_2$   
 $\overrightarrow{111,01}_2 = 11,101_2$

$$111,01_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} =$$

$$= 4 + 2 + 1 + 0 + 0,25 = 7,25_{10}$$

$$11,101_2 = 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} =$$

$$= 2 + 1 + 0,5 + 0 + 0,125 = 3,625_{10}$$

11

### Conversão decimal $\rightarrow$ binário

	57	2						
	$\overline{17}$	18	2					
	<b>1</b>	$\overline{08}$	14	2				
bit menos significativo (LSB)		$\overline{0}$	$\overline{0}$	7	2			
				$\overline{1}$	3	2		
					$\overline{1}$	1		
						<b>1</b>		

$57_{10} = 111001_2$

bit mais significativo (MSB)

12

### ● *Bits, bytes, nibbles e words:*

- *bit* = um dígito binário ("binary digit")
- *byte* = uma unidade de 8 bits
- *nibble* = uma unidade de 4 bits = 1/2 byte
- *word* = uma unidade de 16 bits = 2 bytes

13

### ● **Aritmética Binária**

- Os números binários podem sofrer operações da mesma forma que os decimais; na verdade, a aritmética binária é bem mais fácil que a decimal.
- A seguir vamos ver como são a *adição*, *subtração*, *multiplicação* e *divisão* binárias.

14

## ● Adição binária

<i>Decimal</i>	<i>Binário</i>
	11 ← vai um
3	011
+6	110
<hr/>	<hr/>
9	1001

15

## ● Subtração binária

- Os circuitos digitais não podem subtrair dígitos binários. Só conseguem somá-los.
- Isso não será um problema se pudermos transformar a subtração numa *soma*.
- Subtrair um número decimal é equivalente a somar seu complemento a 10.

16

● Complemento a 10

<i>Subtração</i>	<i>Soma Complementar</i>
	<del>X</del> ← <i>despreza – se o vai um</i>
9	9
<u>-2</u>	<u>+8</u> ← <i>complemento a 10 de 2</i>
7	<del>X</del> 7

17

● Complemento a 1

<i>número original</i>	1010100
<i>complementado a um</i>	0101011

● Complemento a 2

<i>número original</i>	1010100
<i>complementado a um</i>	0101011
	<u>+1</u>
<i>complementado a dois</i>	0101100

18

● *Subtração A - B, com A > B*

<i>Subtração</i>	<i>Soma Complementar a 2</i>
	<del>X</del> ← <i>despreza – se o vai um</i>
10001	10001
<u>-01011</u>	<u>+10101</u> ← complemento a 2 de 01011
00110	<del>X</del> 00110

19

● *Subtração A - B, com A < B*

<i>Subtração</i>	<i>Soma Complementar a 2</i>
	1 ← <u>não</u> despreza – se o vai um
101	101
<u>-11011</u>	<u>+00101</u> ← complemento a 2 de 11011
-10110	<u>+01010</u> ← complemento a 2 de 10110
	 <b>carry final</b>

- Quando se faz a subtração por complemento a dois, o “vai um” final fornece o sinal da resposta. Se 1, ela é positiva; se 0, negativa.

20

## ● Multiplicação Binária

<i>Decimal</i>	<i>Binário</i>
9	1001
<u>×5</u>	<u>×101</u>
45	1001
	0000
	<u>1001</u>
	101101

21

## ● Divisão binária

- É feita usando-se os mesmos passos da divisão decimal.

$$\begin{array}{r} 110111 \quad |101 \\ - 101 \quad 1011 \\ \hline 0011 \\ 00111 \\ - 00101 \\ \hline 000101 \\ - 000101 \\ \hline 0 \end{array}$$

22

## ● O Sistema Octal (0, 1, 2, 3, 4, 5, 6, 7)

- É usado em técnicas de computadores porque a conversão de binário para octal é simples.
- Os números são expressos como combinações de potências de 8, como nos exemplos:

$$\begin{aligned}765_8 &= 7 \cdot 8^2 + 6 \cdot 8^1 + 5 \cdot 8^0 = \\ &= 448 + 48 + 5 = 501_{10}\end{aligned}$$

$$\begin{aligned}3,14_8 &= 3 \cdot 8^0 + 1 \cdot 8^{-1} + 4 \cdot 8^{-2} = \\ &= 3 + 0,125 + 0,0625 = 3,1875_{10}\end{aligned}$$

23

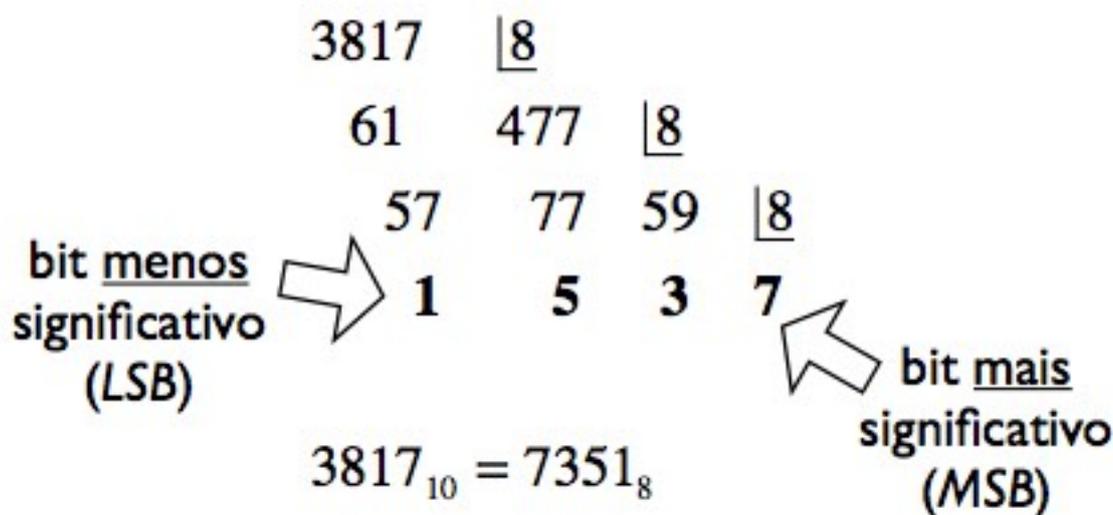
- Como a base do sistema octal é o número 8 =  $2^3$ , é muito fácil a conversão de números octais em binários e vice-versa:

$$101010_2 = 101|010 = 5_8|2_8 = 52_8$$

$$\begin{aligned}101010_2 &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = \\ &= (32 + 8 + 2)_{10} = 42_{10} \\ 52_8 &= 5 \cdot 8^1 + 2 \cdot 8^0 = 42_{10}\end{aligned}$$

24

## ● Conversão decimal → octal



25

## ● O Sistema Hexadecimal

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

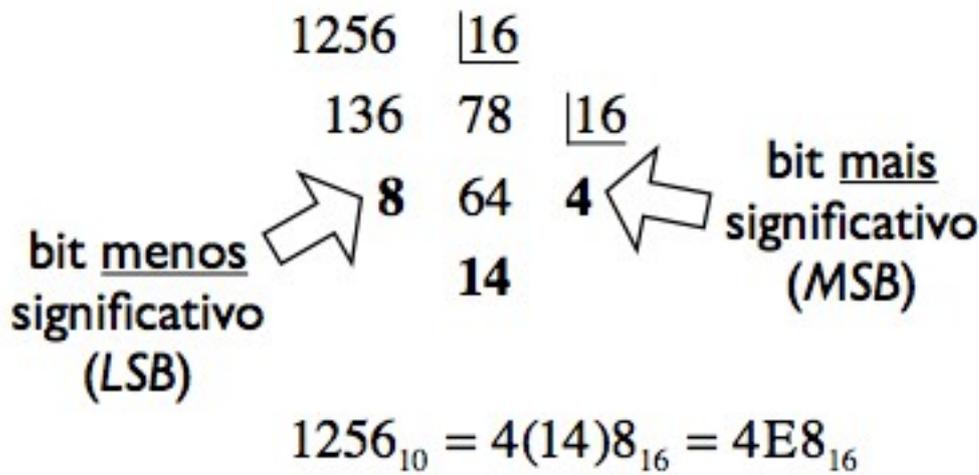
● É muito usado em técnicas de computadores porque a conversão de binário e octal para hexadecimal é simples.

● Os números são expressos como combinações de potências de 16, como nos exemplos:

$$\begin{aligned} 11010110_2 &= 011_2 | 010_2 | 110_2 = 3_8 | 2_8 | 6_8 = 326_8 \\ &= 1101_2 | 0110_2 = D_{16} | 6_{16} = D6_H = D6h \\ D6h &= 13 \cdot 16^1 + 6 \cdot 16^0 = 214_{10} \end{aligned}$$

26

● *Conversão decimal → hexadecimal*



● **Resumo:**

EQUIVALENTES BINARIOS								
Número Decimal	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	=		Número Hexa
0	0	0	0	0	0	=	0·2 <sup>0</sup>	0
1	0	0	0	0	1	=	1·2 <sup>0</sup>	1
2	0	0	0	1	0	=	1·2 <sup>1</sup> + 0·2 <sup>0</sup>	2
3	0	0	0	1	1	=	1·2 <sup>1</sup> + 1·2 <sup>0</sup>	3
4	0	0	1	0	0	=	1·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	4
5	0	0	1	0	1	=	1·2 <sup>2</sup> + 0·2 <sup>1</sup> + 1·2 <sup>0</sup>	5
6	0	0	1	1	0	=	1·2 <sup>2</sup> + 1·2 <sup>1</sup> + 0·2 <sup>0</sup>	6
7	0	0	1	1	1	=	1·2 <sup>2</sup> + 1·2 <sup>1</sup> + 1·2 <sup>0</sup>	7
8	0	1	0	0	0	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	8
9	0	1	0	0	1	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	9
10	0	1	0	1	0	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	A
11	0	1	0	1	1	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	B
12	0	1	1	0	0	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	C
13	0	1	1	0	1	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	D
14	0	1	1	1	0	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	E
15	0	1	1	1	1	=	1·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	F
16	1	0	0	0	0	=	1·2 <sup>3</sup> + 0·2 <sup>2</sup> + 0·2 <sup>2</sup> + 0·2 <sup>1</sup> + 0·2 <sup>0</sup>	10

# Representação dos estados digitais

- Os estados digitais são representados através de níveis de tensão elétrica ou níveis lógicos.
- Estes níveis dependem da tecnologia empregada na construção dos circuitos digitais.
  - TTL (*Transistor-Transistor Logic*) ou
  - CMOS (*Complementary Metal-Oxide-Semiconductor*, i.e., semicondutor metal-óxido complementar)

29

## TTL

- Os circuitos digitais TTL tem com principal característica a utilização de sinais de 5 volts para níveis lógicos altos (1) e 0 volts para níveis lógicos baixos (0).
- Seus circuitos integrados são constituídos basicamente de transístores, o que os torna pouco sensíveis à eletricidade estática.

30

# CMOS

- A principal vantagem dos circuitos integrados CMOS é o baixíssimo consumo de energia, embora não sejam capazes de operar tão velozmente quanto circuitos integrados de outras tecnologias.
- Por causa disso, são largamente utilizados em calculadoras, relógios digitais, e outros dispositivos alimentados por pequenas baterias.

31

## Circuitos Integrados

- Um *chip* contendo um grande número de “pernas” (na verdade, portas internas) denomina-se *circuito integrado* (CI). Com o uso dos CIs, é possível produzir diversos circuitos como contadores, registradores, Amp-Ops numa única pastilha de silício. O que determina a quantidade de componentes que estarão presentes numa pastilha é a *escala de integração*:

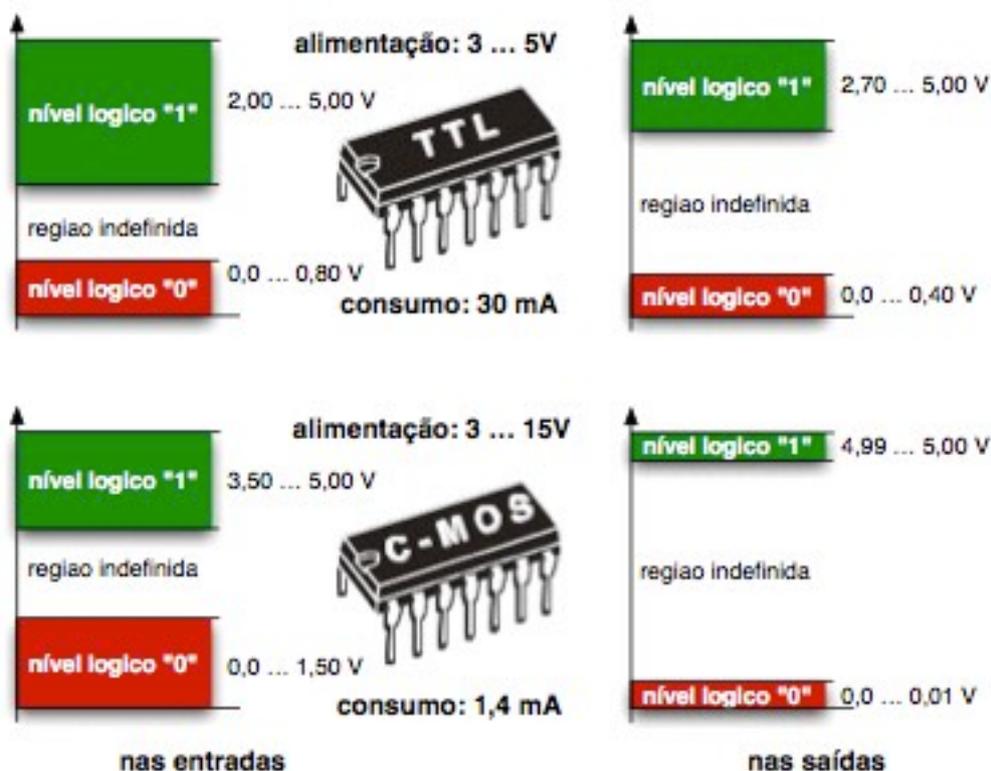
SSI (integração em baixa escala)	de 10 a 100 portas
MSI (integração em média escala)	de 100 a 1000 portas
LSI (integração em grande escala)	de 1000 a 10000 portas
VSI ou VLSI (integração em escala muito grande)	acima de 10000 portas

32

- O custo de um *chip* é função de seu tamanho físico (pastilha de silício) — não é função de quanta lógica foi implementada dentro do *chip*.
- Portanto, quanto mais os *chips* se tornarem complexos, mais barato será fazer a unidade de controle eletrônico (ECU).

33

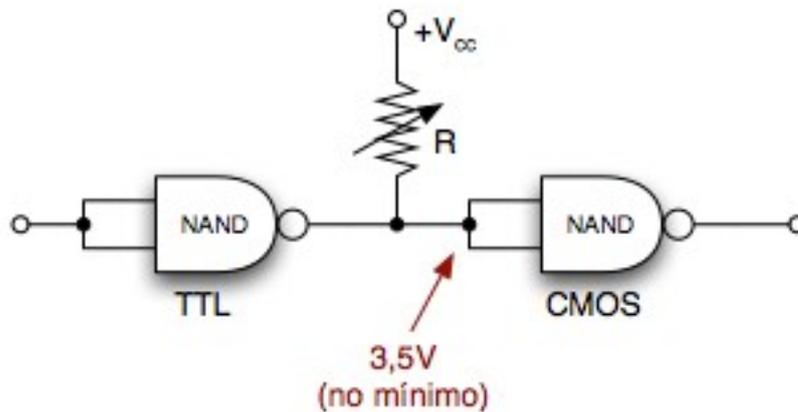
## Níveis Lógicos



34

## ● Casamento entre um TTL e um CMOS

Às vezes é necessário misturar circuitos...



35

## ● *Tri-state*: o terceiro estado

Além dos estados "*high*" e "*low*", existe um terceiro, chamado de estado de alta impedância.

Nesse terceiro estado, a **saída** assume o nível de tensão ligado nela.

Com isso podemos conectar juntas as saídas de diversos dispositivos digitais numa mesma linha ou duto (*bus*).

Apenas um dispositivo pode acessar o duto.

36

# Considerações

- **Ruído:** os TTLs e a maioria das famílias lógicas geram picos de corrente quando chaveiam. Esse ruído pode ser filtrado acoplando um capacitor de 10nF por porta e 100nF para cada 20 portas. Os pinos não usados devem ser aterrados (0 V) com um resistor de 1-10 K $\Omega$ .
- **Potência:** com os TTLs não é possível acionar diretamente outros dispositivos que consomem uma potência razoável. Entretanto, um coletor aberto pode ser usado como uma chave de 15V e 100mA.

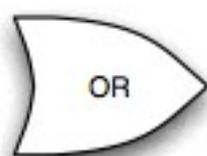
37

# Símbolos Lógicos

- São usados para representar facilmente uma expressão lógica. Os símbolos representam também os elementos que realizam essa expressão. Os elementos têm a denominação de "*elementos lógicos*" ou "*operadores lógicos*". Os símbolos mais usados são:



(a) &



(b) ou



(c) amplificação

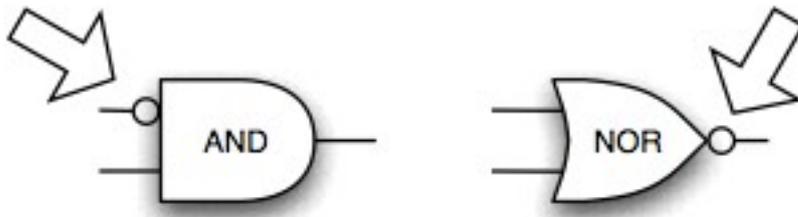


(d) indicativo de estado

38

## ● O elemento indicativo de estado:

O símbolo indicativo de estado que é representado por um pequeno círculo, pode ser colocado tanto na *entrada* como na *saída* do operador, e ele indica uma *negação* do estado lógico.



39

# Operadores Lógicos

- Não existe nada de misterioso sobre os operadores lógicos. Adição, subtração, multiplicação e divisão são operadores aritméticos.
- Um operador lógico pega dados de entrada, executa alguma coisa não aritmética com eles e cria um resultado da mesma maneira como os operadores aritméticos pegam dados de entrada, fazem alguma coisa aritmética sobre eles e geram um resultado.

40

● Existem quatro operadores lógicos básicos:

1. NOT (*negação* ou *inversor*);
2. AND (*e*);
3. OR (*ou*) e;
4. XOR (Exclusive-OR ou *ou-exclusivo*).

41

## Circuitos Combinacionais

● Em um circuito combinacional, a saída depende apenas de uma combinação das entradas.

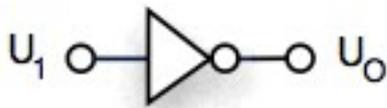
## Tabela verdade

● Tabela verdade ou tabela de verdade é um tipo de tabela matemática usada em lógica digital para determinar se uma expressão é verdadeira e válida.

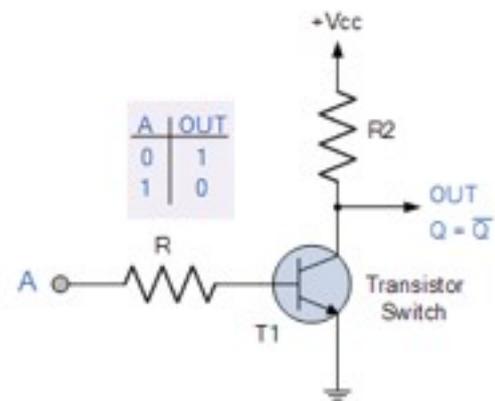
42

## Operador NOT

O NOT nega sua entrada, convertendo um 1 em um 0 ou um 0 em um 1.



negação



Entrada	Saída
0	1
1	0

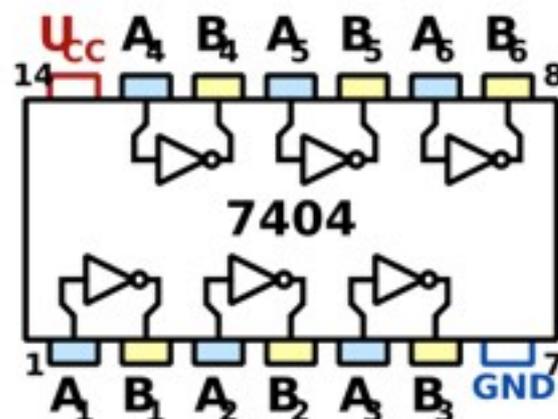
$$U_0 = \overline{U_1}$$

43

## Exemplos:

1. TTL: 7404.

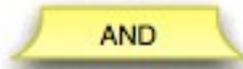
2. CMOS: 4001.



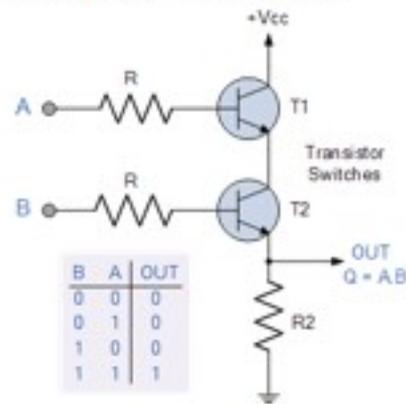
44

## Operador AND

O AND verifica se todas as suas entradas estão no nível lógico 1.

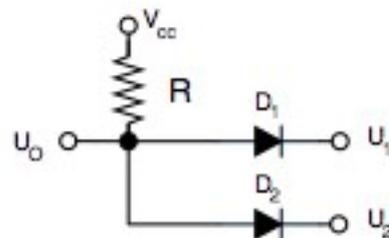


Entrada 1	Entrada 2	Saída
0	0	0
0	1	0
1	0	0
1	1	1



B	A	OUT
0	0	0
0	1	0
1	0	0
1	1	1

$$U_0 = U_1 \cdot U_2$$

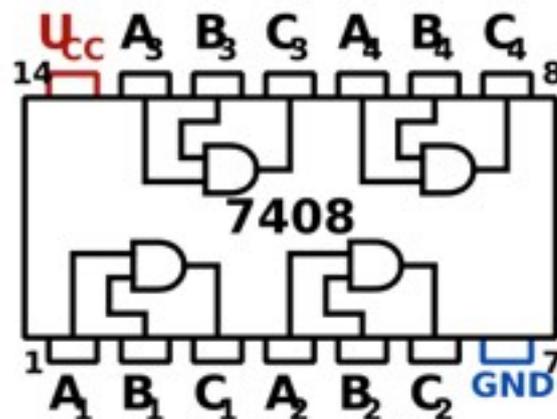


45

## Exemplos:

1. TTL: 7408.

2. CMOS: 4081.



46

● Aplicação: teste de condição de aceleração

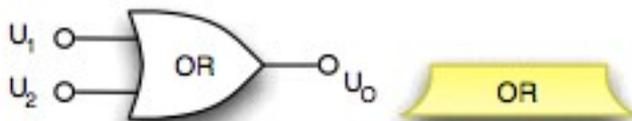


Entrada 1	Entrada 2	Saída
0	0	0
0	1	0
1	0	0
1	1	1

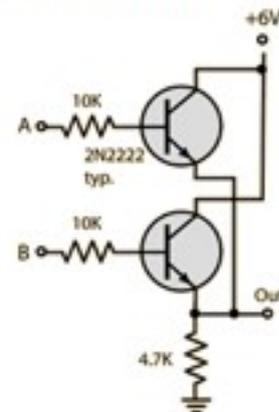
47

● Operador OR

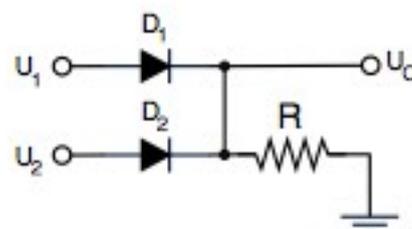
O OR testa se qualquer uma das entradas está no nível lógico 1.



Entrada 1	Entrada 2	Saída
0	0	0
0	1	1
1	0	1
1	1	1



$$U_0 = U_1 + U_2$$

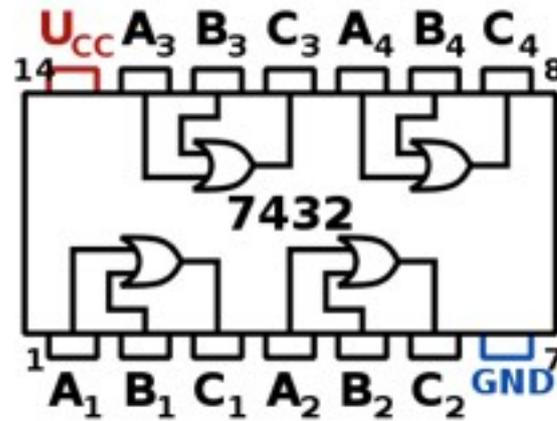


48

Exemplos:

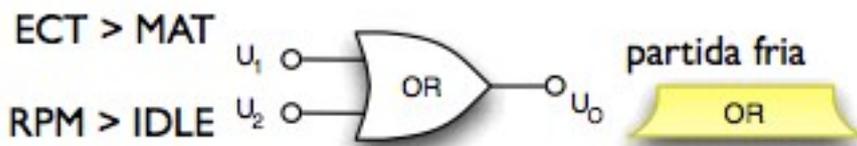
1. TTL: 7432.

2. CMOS: 4071



49

Aplicação: teste de condição de partida fria



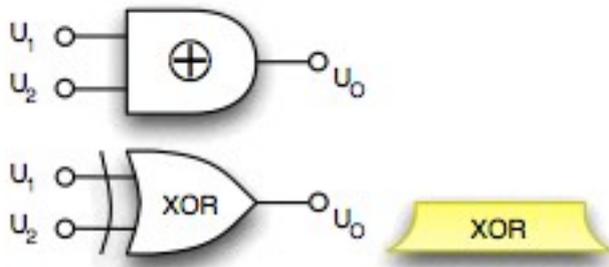
Entrada 1	Entrada 2	Saída
0	0	0
0	1	1
1	0	1
1	1	1

$$U_0 = U_1 + U_2$$

50

## Operador XOR

O XOR identifica ou checa as diferenças e mudanças entre suas entradas.



Entrada 1	Entrada 2	Saída
0	0	0
0	1	1
1	0	1
1	1	0

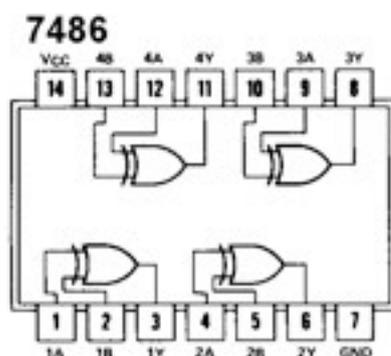
$$U_0 = U_1 \oplus U_2$$

51

## Exemplos:

1. TTL: 7486.

2. CMOS: 4070



52

## Outros operadores

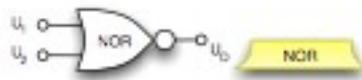
Estes outros operadores também são muito úteis:



Entrada 1	Entrada 2	Saída
0	0	1
0	1	1
1	0	1
1	1	0

$$U_0 = \overline{U_1 \cdot U_2}$$

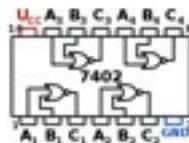
TTL: 7400 e 7401  
CMOS: 4011



Entrada 1	Entrada 2	Saída
0	0	1
0	1	0
1	0	0
1	1	0

$$U_0 = \overline{U_1 + U_2}$$

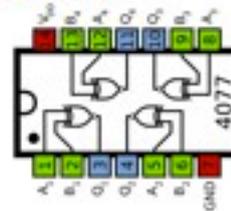
TTL: 7402  
CMOS: 4001



Entrada 1	Entrada 2	Saída
0	0	1
0	1	0
1	0	0
1	1	1

$$U_0 = U_1 \oplus U_2$$

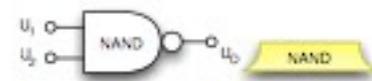
CMOS: 4077



53

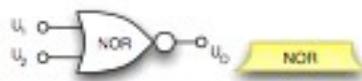
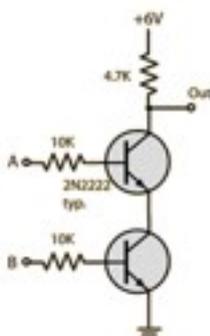
## Outros operadores

Estes outros operadores também são muito úteis:



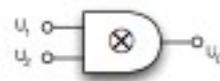
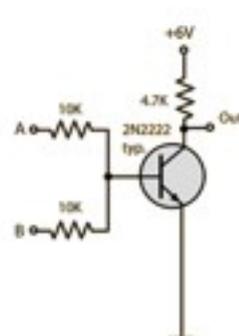
Entrada 1	Entrada 2	Saída
0	0	1
0	1	1
1	0	1
1	1	0

$$U_0 = \overline{U_1 \cdot U_2}$$



Entrada 1	Entrada 2	Saída
0	0	1
0	1	0
1	0	0
1	1	0

$$U_0 = \overline{U_1 + U_2}$$



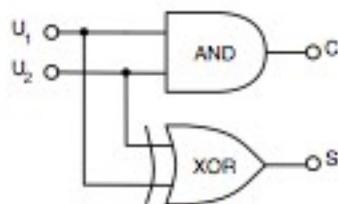
Entrada 1	Entrada 2	Saída
0	0	1
0	1	0
1	0	0
1	1	1

$$U_0 = U_1 \oplus U_2$$

54

## ● Aplicações

### ● semi-somador (*half-adder*)



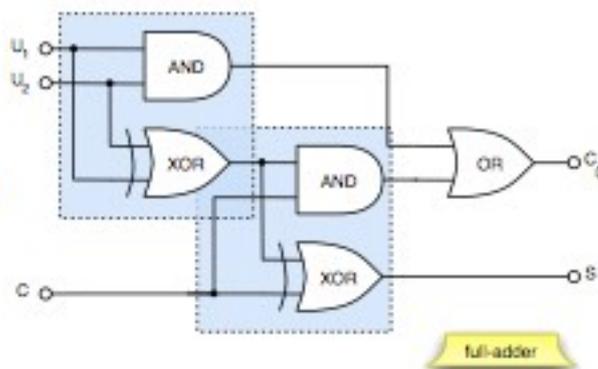
half-adder

Entrada 1	Entrada 2	Saída C	Saída S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

55

## ● Aplicações

### ● somador completo (*full-adder*)



full-adder

Entrada C	Entrada 1	Entrada 2	Saída Co	Saída S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

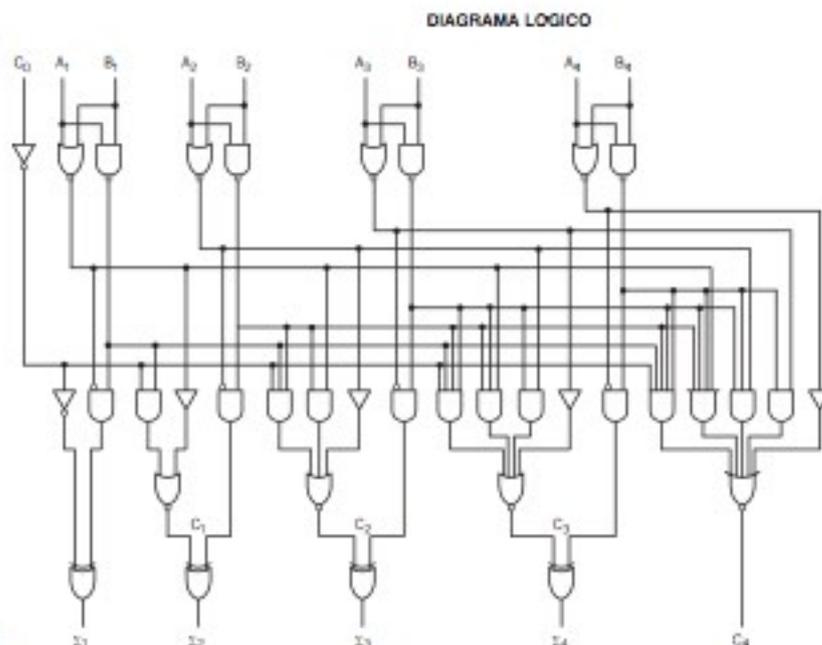
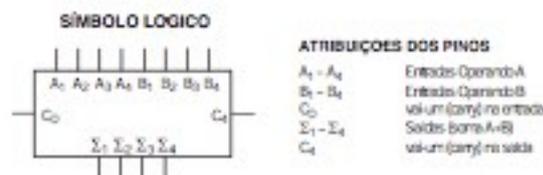
$$Co = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

56

## Aplicações

## Somador completo de 4 bits



Exemplo: 4008

$$C_0 + (A_1 + B_1) + 2(A_2 + B_2) + 4(A_3 + B_3) + 8(A_4 + B_4) = \Sigma_1 + 2 \Sigma_2 + 4 \Sigma_3 + 8 \Sigma_4 + 16C_4$$

Em que: (+) = mais

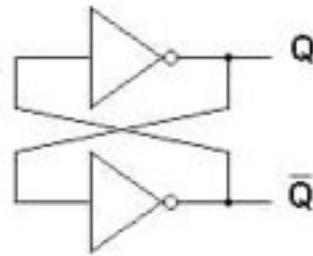
# Circuitos Lógicos Seqüenciais

- Os circuitos seqüenciais são uma classe de circuitos digitais cuja saída depende das entradas no estado presente e no estado passado, ou seja, são circuitos que mantêm informação armazenada que é utilizada juntamente com a informação atual nas suas entradas para gerar a saída.
- Os circuitos lógicos seqüenciais armazenam informação mesmo depois de removidas as entradas.

## ● **latch** (trinco ou ferrolho)

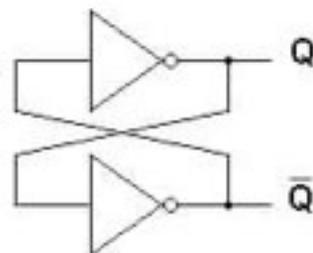
A forma mais básica de implementar-se um circuito lógico de memória é conhecida como *latch*.

Sua arquitetura é composta de duas portas lógicas inversoras, possuindo duas saídas: a variável lógica  $Q$  e o seu complemento lógico,  $\bar{Q}$ .



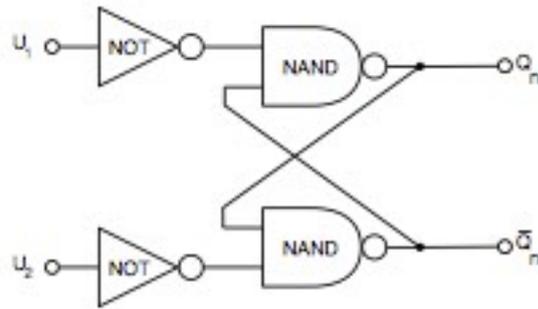
59

- Se você impõe nível lógico alto (1) em  $Q$ , seu complemento vai para o nível lógico baixo (0). Esse estado ( $\bar{Q} = 0$ ) permanecerá até que você imponha nível lógico baixo a  $Q$ .
- É ou não é um dispositivo de memória?



60

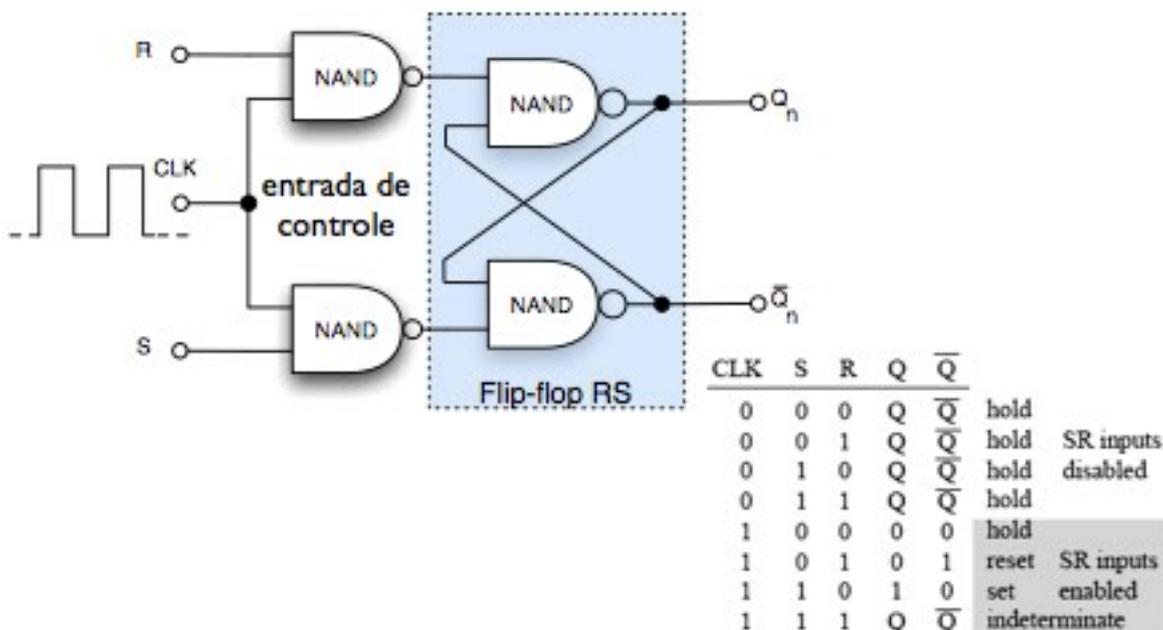
- Outro circuito seqüencial bem simples pode ser construído interligando dois NANDS, conforme a figura abaixo:



Entrada 1	Entrada 2	Saída Qn	Saída Qn
0	0	$Q_{n-1}$	$\bar{Q}_{n-1}$
0	1	0	1
1	0	1	0
1	1	?	?

61

- *Latch* RS com entrada de controle



62

- Se  $S=1$  e  $R=0$ ,  $Q=1$  e permanecerá assim independentemente do novo valor de  $S$ , se 1 ou 0, mantendo-se  $R=0$ .
- Dizemos que o estado "high" de  $S$  está trancado com o estado da saída  $Q$ . A única maneira de  $Q$  ser destrancada e ser "low" é fazer  $R=1$  (high) e  $S=0$  (low). Isto "zerará" a tranca (latch).
- Este tipo de dispositivo de memória é chamado de *flip-flop R-S (reset-set)*. O termo "flip-flop" descreve a ação das mudanças do nível lógico em  $Q$ .

63

### Flip-flop JK

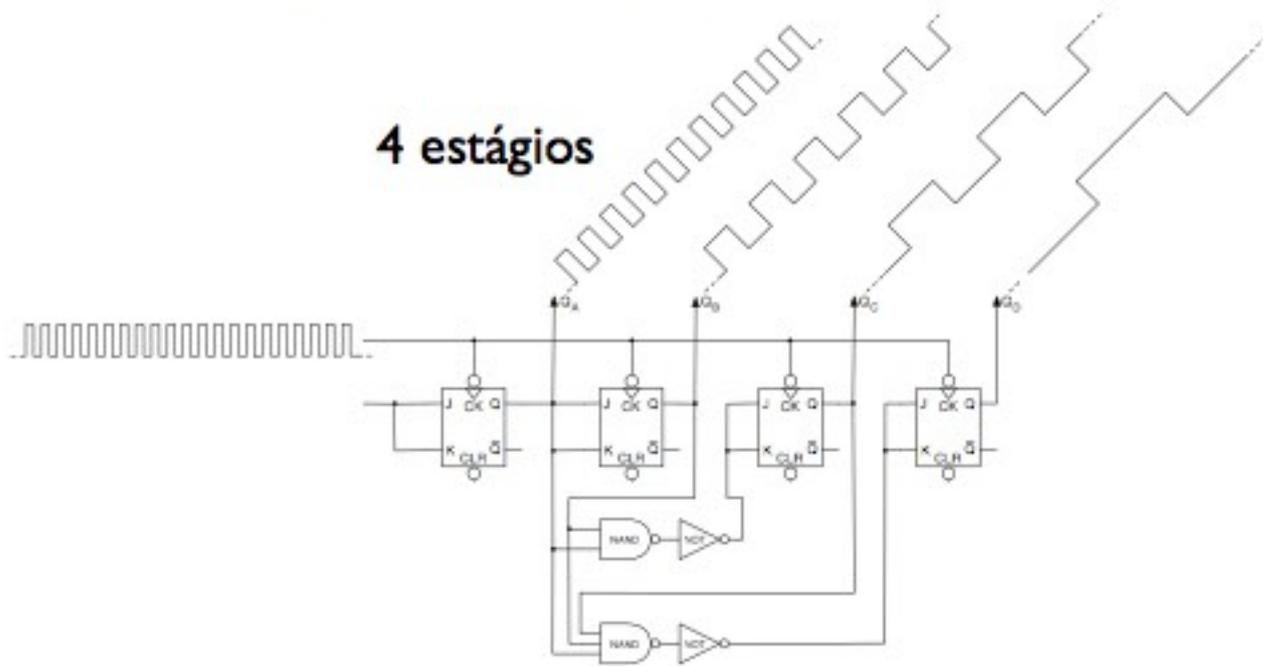
O *flip-flop RS* não teve uma aplicação efetiva por causa da questão mal resolvida quando ambas as entradas estavam simultaneamente em "high". A solução definitiva veio com o *flip-flop JK*. Este muda o estado da saída na transição do sinal  $CK$  (clock) de acordo com o nível lógico das entradas  $JK$ . A entrada  $CLR$  inicializa o JK.



64

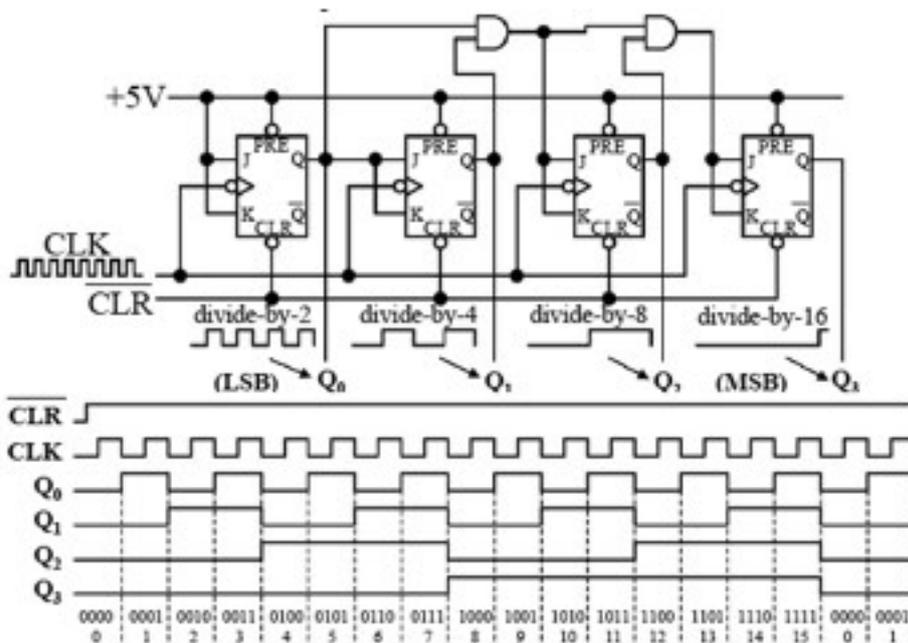
# Contadores Lógicos

- A combinação de 2 ou mais flip-flops produz um contador lógico, como o da figura abaixo:



65

## Contador síncrono de 4 bits



exemplo: 74939 (2 contadores de 4 bits)

66

# Registradores e Memórias

- Quando juntamos diversos *flip-flops* com a finalidade de armazenar temporariamente uma informação, temos o que se diz *registrador*.
- Os registradores são unidades de memória capazes de armazenar  $n$  bits. Os registradores estão no topo da hierarquia de memória, sendo assim, são o meio mais rápido e caro de se armazenar um dado.
- Se o armazenamento for mais duradouro, o registrador recebe o nome de *memória*.

67

## Circuitos Aritméticos

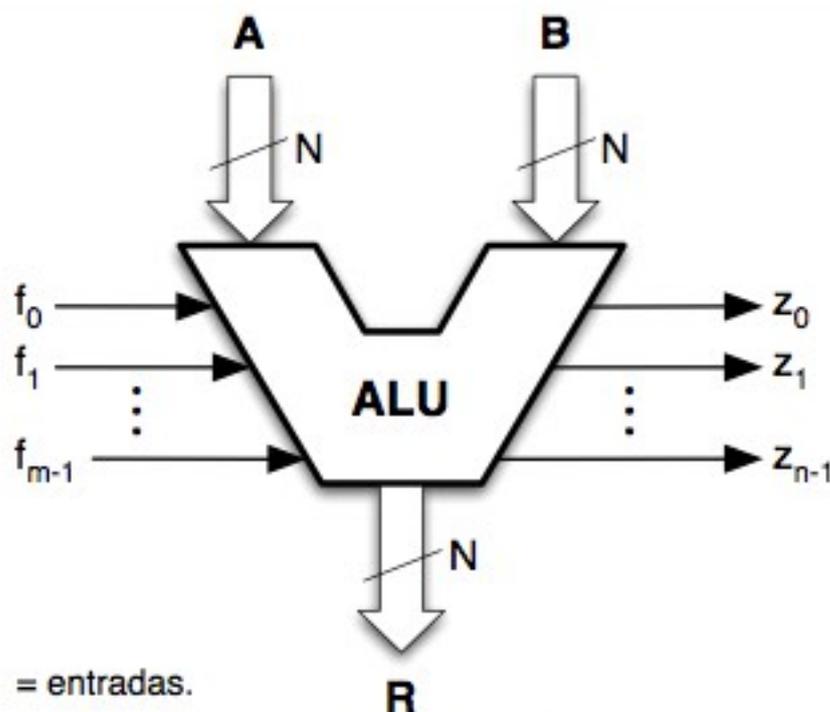
- Circuito aritmético é um tipo de circuitos combinacionais que executa operações de subtração, adição, multiplicação, divisão, AND lógico, OR lógico ou qualquer outra função que possa ser implementada num circuito combinacional.
- Os circuitos aritméticos são a base das chamadas ULA (unidade lógica aritmética) ou ALU do inglês.

68

# Unidade Lógica e Aritmética (ALU)

- A ALU é o circuito responsável pelas operações aritméticas de números binários e também pelas operações lógicas principais. As primeiras ALU's podiam realizar apenas somas e as operações lógicas *AND*, *OR* e *NOT*.
- Hoje, encontramos ALU's que realizam operações de multiplicação, aritmética de ponto flutuante e até funções trigonométricas e hiperbólicas.

69



A, B = entradas.

$f_i$  = funções aritméticas selecionáveis.

$Z_j$  = indicador de estados da ALU, p. ex., negativo, vai-um, ... geralmente e transmitido para o registrador da condição de código.

70

# O Microprocessador (MPU)

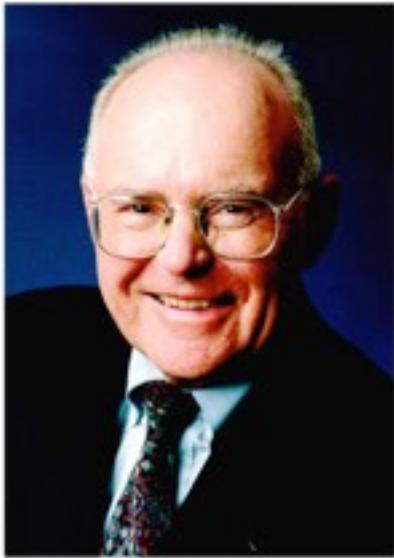
- O microprocessador (MPU) é um dispositivo eletrônico no qual são incorporados uma infinidade de circuitos lógicos digitais.
- Entretanto, sozinho um microprocessador não é capaz de nada. É necessário complementá-lo com outros circuitos digitais, dos quais um que seja capaz de fornecer instruções na forma de sinais elétricos codificados digitalmente.

71

- A partir das instruções o microprocessador executará operações lógicas e aritméticas.
- Uma das vantagens do microcontrolador é que o *software* pode substituir o uso de diversas portas lógicas e interfaces.
- Além disso, um microcontrolador possui *hardware* necessário para adquirir sinais contínuos e transformá-los em discretos.

72

# Lei de Moore

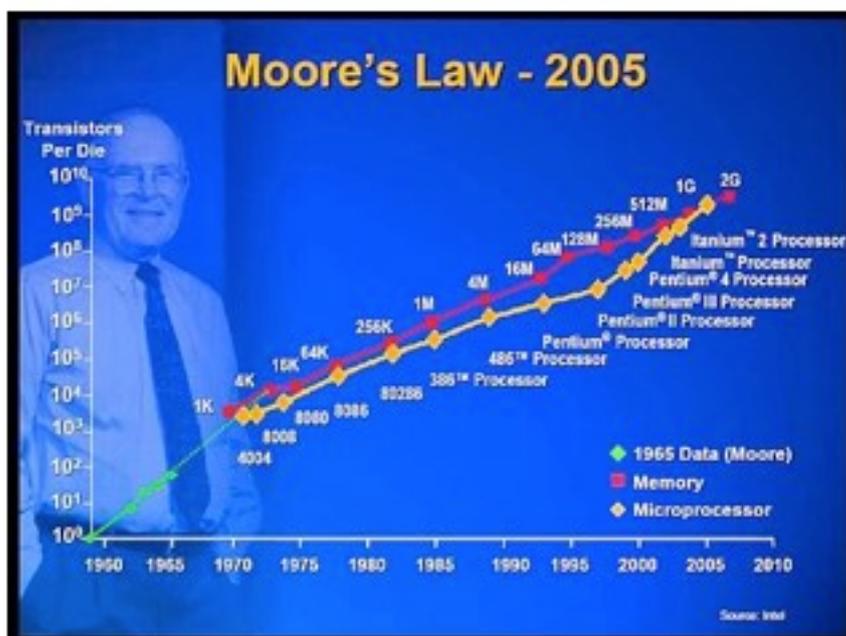


Em 1965 Gordon E. Moore era presidente da Intel.

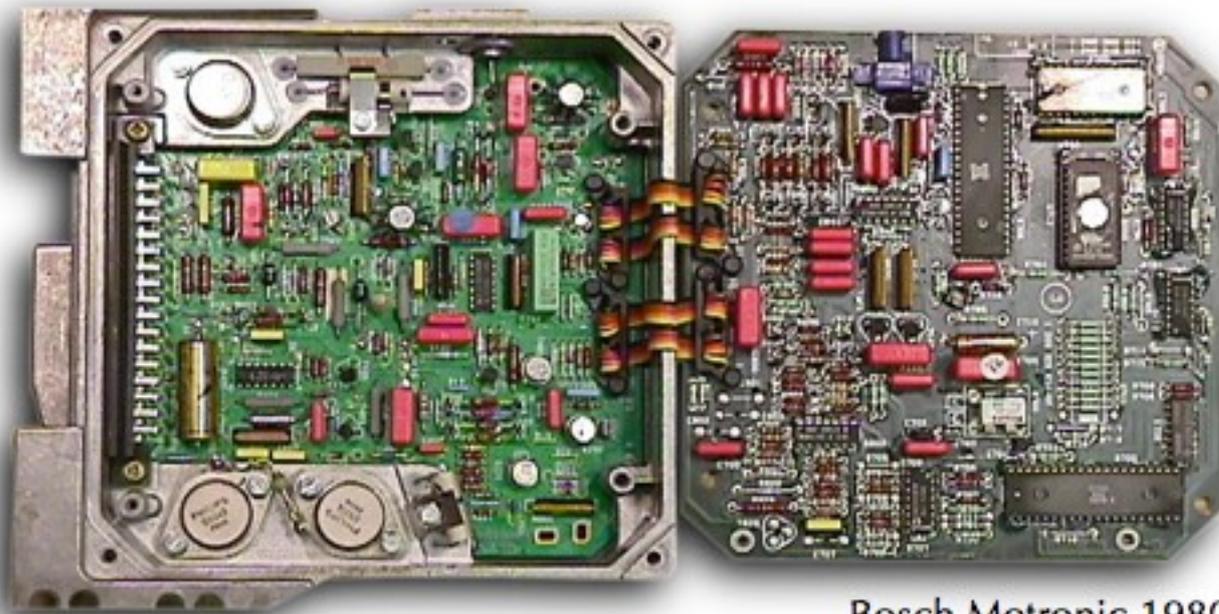
- Moore fez uma profecia em 1965: *"o número de transistores dos chips terá um aumento de 100%, pelo mesmo custo, a cada período de 18 meses"*. (Lei de Moore).
- Esta serve de parâmetro para uma elevada gama de dispositivos digitais, além de CPUs. Na verdade, qualquer chip está ligado a lei de Moore.

- Esse padrão continua a se manter, e não se espera que pare até, no mínimo, 2015.

73



74

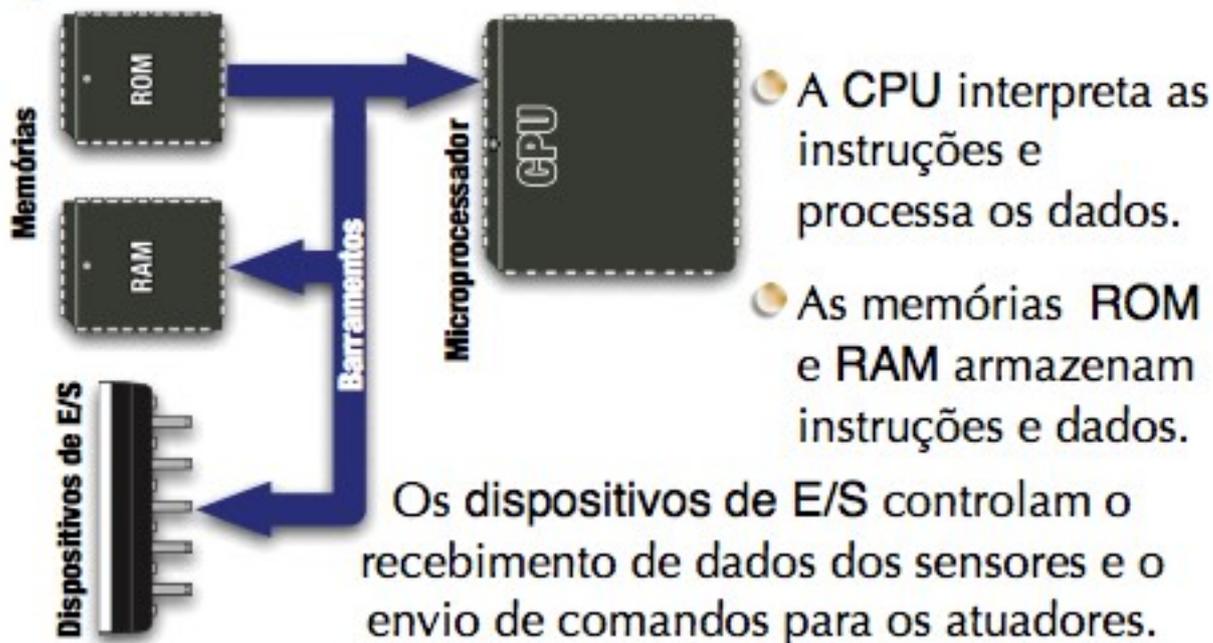


Bosch Motronic 1980

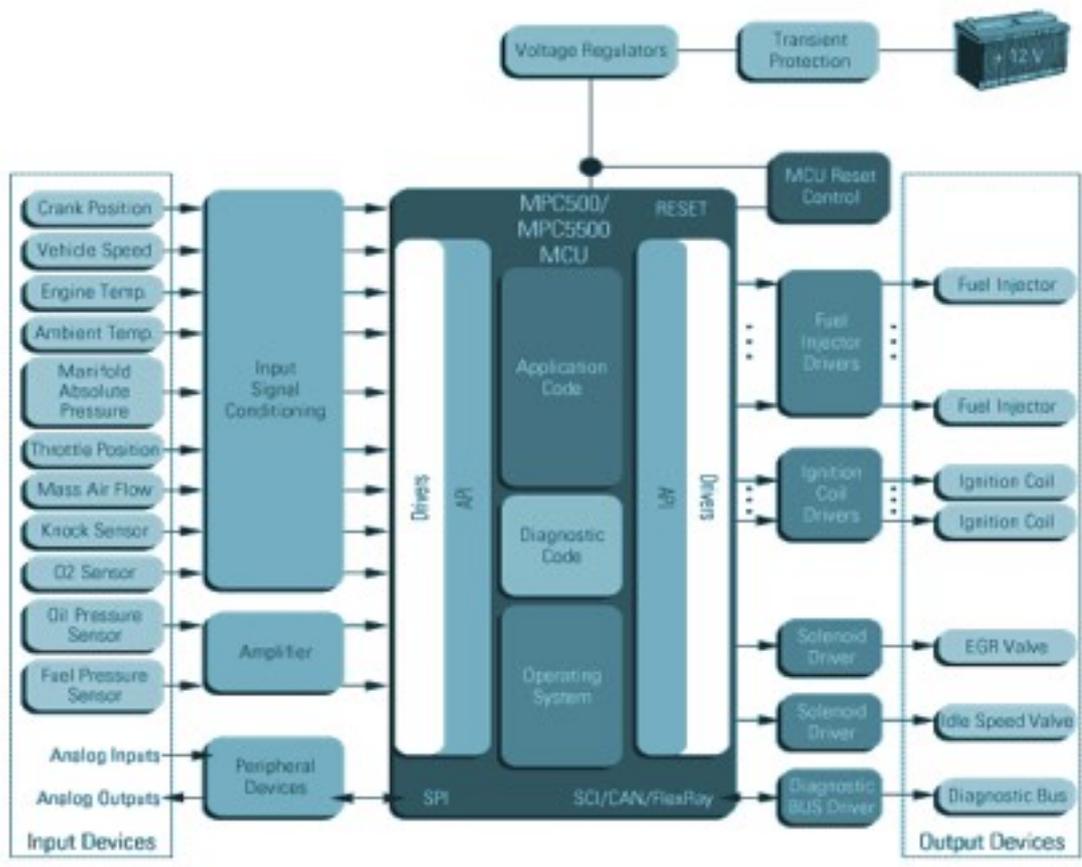
## A ECU

75

### Estrutura básica de uma ECU para Instrumentação e Controle



76



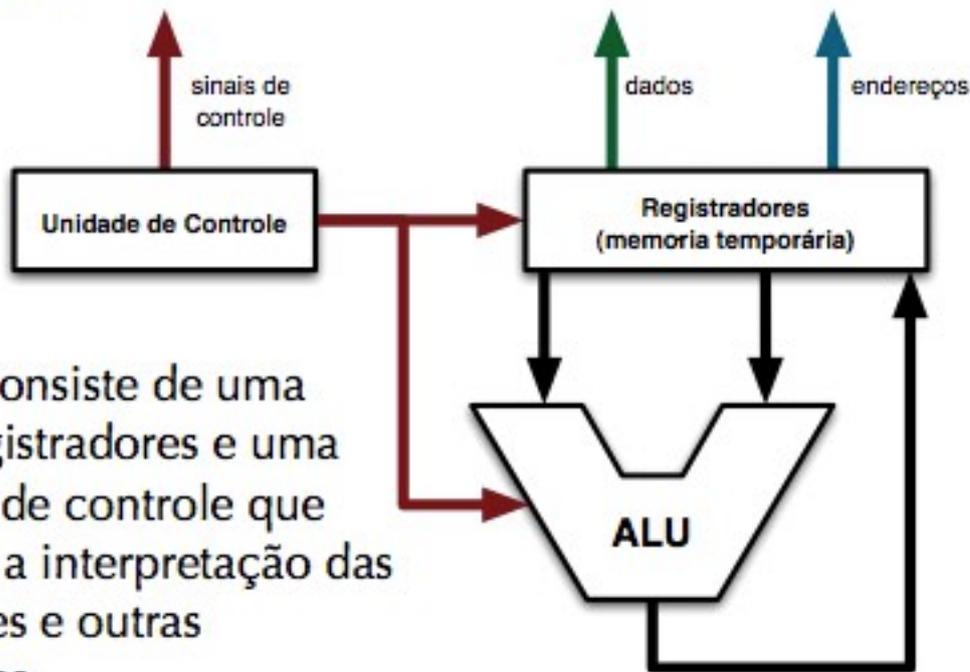
Engine Control System Block Diagram

Fonte: Motorola

## A evolução das ECU



## ● CPU



A CPU consiste de uma ULA, registradores e uma unidade de controle que controla a interpretação das instruções e outras operações.

79

- Uma CPU ou microprocessador é um dispositivo computacional completo que é fabricado em um único *chip*.
- A CPU é o coração de qualquer computador seja ele um *desktop*, um servidor ou um *laptop*.
- Uma CPU executa um conjunto de instruções (de máquina) que dizem passo a passo para a CPU o que fazer.

80

● Basicamente uma CPU pode fazer 3 coisas:

1. por meio de sua ULA executar operações matemáticas (+, -, # e ');
2. mover dados de um lugar da memória para outro;
3. tomar decisões e pular para um novo conjunto de instruções baseando-se em suas decisões.

81

● Os registradores, feitos de *flip-flops*, mantêm a informação temporariamente. A informação é transmitida para dentro e para fora dos registradores através dos barramentos. Estes dividem-se em:

1. uso geral;
2. endereço;
3. contador de programa

82

● Memórias ROM (somente leitura)

a) PROM - programada de fábrica;

b) OTPROM - programada só uma vez;

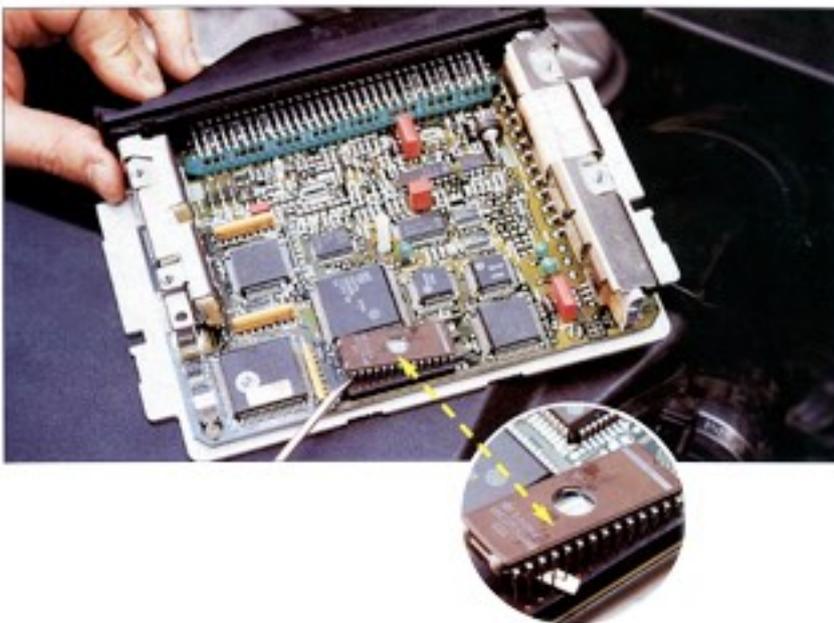
c) EPROM;

d) EEPROM;

e) *flash*.

83

● onde se localiza a ROM



84

## ● Memórias RAM

a) DRAM - RAM dinâmica (capacitores);

b) SRAM - RAM estática (flip-flops);

● As memórias SRAM são mais rápidas, caras e ocupam maior espaço que as DRAM sendo mais usadas como memórias tipo *cache*.

● A CPU pode ler ou escrever na RAM informações dependendo do nível lógico dos sinais RD e WR.

85

## ● Dispositivos de E/S

a) portas seriais

b) portas paralelas

c) conversores A/D

d) temporizadores (*timers*)

● A CPU, as memórias e os dispositivos de E/S podem ser reunidos num único chip que recebe o nome de *single chip* ou *one-chip MCU*.

86

# Microcontrolador (MCU)

- Todos os automóveis modernos possuem pelo menos um microcontrolador. O motor é controlado por um MCU, assim como os sistemas *ABS*, *powertrain*, *cruise control*, etc.
- Um MCU possui um computador que está “**embutido**” (*embedded*) dentro dele de modo a controlar suas ações e características.

87

- Os MCUs são **dedicados** a uma determinada tarefa e rodam um programa específico.
- Frequentemente são dispositivos de baixíssimo consumo de energia. Enquanto um computador doméstico consome da ordem de 50W, o consumo de um MCU é da ordem de 50 mW.
- Outra característica é que geralmente (mas nem sempre) possuem um dispositivo dedicado de entrada e um de saída (teclado e *LEDs* ou *LCD*).

88

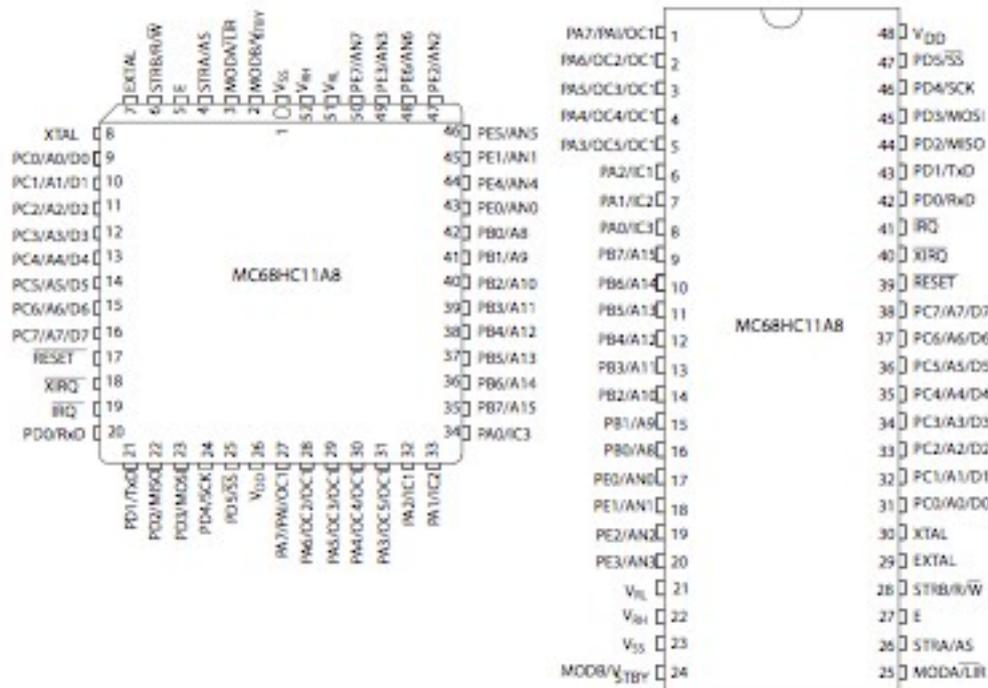
- O MCU recebe informações da planta que está controlando e controla-a enviando sinais a diferentes componentes da planta.
- Um MCU é geralmente pequeno e de baixo custo.
- De alguma maneira são muito robustos (nem sempre) para trabalhar em condições ambientes extremas. Como, por exemplo, temperaturas de  $-39^{\circ}\text{C}$  a  $+80^{\circ}\text{C}$ .

89

- MCU, além do microprocessador (MPU), agrega outros circuitos eletrônicos específicos como:
  - a. conversores A/D;
  - b. contadores;
  - c. PWM;
  - d. temporizadores;
  - e. CAN (*controller area network*);
  - f. memórias RAM, ROM, E(E)PROM e *Flash*;
  - g. outros.

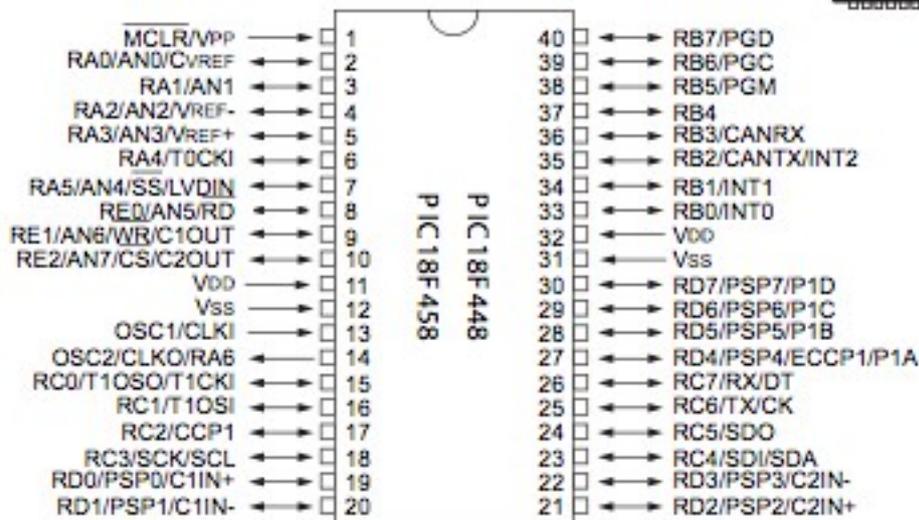
90

## Exemplo: Motorola HC11



91

## exemplo: Microchip PIC



92



- Os DSPs têm sido utilizados em diversas aplicações automotivas destacando-se duas áreas principais:

1. diagnóstico e;
2. controle de motores.

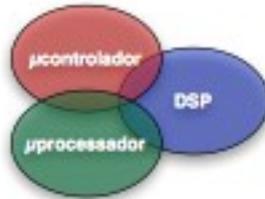
- Na área de diagnóstico, o DSP tem sido usado no diagnóstico de falhas e monitoramento das características da combustão, vibrações e emissões acústicas.
- Na área de controle, o DSP tem sido usado na captura e análise de sinais auxiliando o MCU da ECU a fazer o controle do motor.

# TriCore

- TriCore é o nome dado a uma tecnologia mais recente que reúne os MCUs e os DSPs.



passado



presente



futuro

97

- Exemplo: DSPic



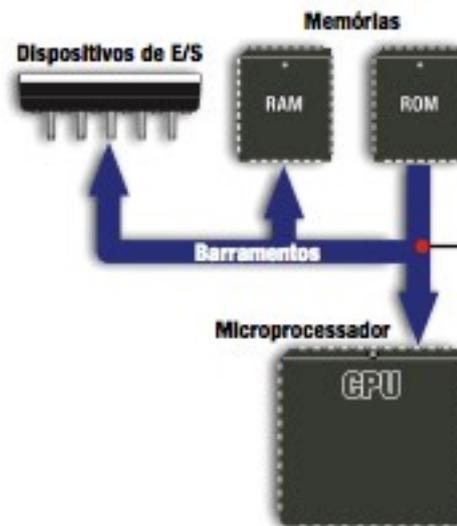
16 bits  
30 – 40 MIPS

98

- **Barramentos** ou dutos ou vias de comunicação

- Os barramentos ou dutos são os meios por onde a CPU e os demais chips trocam informações. São linhas elétricas condutoras sobre as quais os dados binários são transferidos. Basicamente estão divididos em:

1. dutos de endereços;
2. dutos de dados;
3. dutos de controle.



99

- *Barramento de endereços (4, 8, 16, 32 ou 64 bits)*

- O tamanho dos dutos depende do número de bits que a CPU manipulará de uma vez (*tamanho da palavra da CPU*).
- Localiza um lugar ou posição na memória (*posição de memória*)

100

- *Barramento de dados (4, 8, 16, 32 ou 64 bits)*
- na posição de memória definida pelo barramento de endereços, um dado pode ser escrito ou lido da memória

101

### ● *Barramento de Controle*

- Conduz sinais especiais que controlam o fluxo da informação, tais como:
  - a) RD (leitura) e WR (escrita);
  - b) RESET (re-inicia a execução);
  - c) CS (seleção de *chip*);
  - d) *clock*.

102



- As FPGAs são muito adequadas para aplicações de alta velocidade porque o processamento das informações pode ser feito em paralelo. Essas características possibilitam combinar software e hardware para o processamento de sinais dentro de um único chip. Assim o hardware pode ser bastante simplificado e robustecido.
- Até o presente momento as FPGAs não podem manipular números em ponto flutuante como fazem os DSPs.

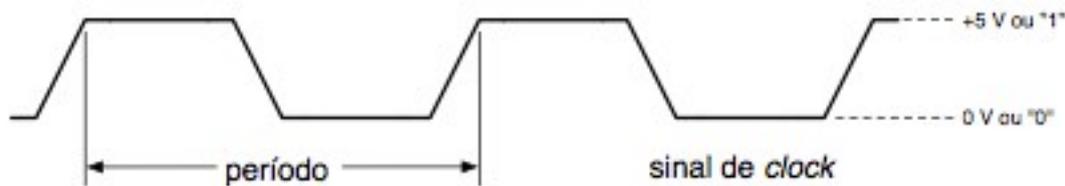
105

- Recentemente um sistema de controle embarcado baseado em FPGA foi desenvolvido para o controle de um motor CI. Esse sistema contou com uma combinação de FPGAs e DSP para explorar as vantagens de cada dispositivo.
- As FPGAs foram escolhidas devido aos benefícios que apresentam em relação aos processadores tais como sincronização, rapidez e paralelismo. Uma vez que as FPGAs estão limitadas hoje à matemática de números inteiros, os processadores de ponto flutuante utilizados (DSPs) supriram essa deficiência. Essa solução pode ser uma tendência de migração da tecnologia dos processadores para a tecnologia dos FPGAs a médio e longo prazos.

106

# Diagramas de Tempo

- As operações dentro de um microcontrolador são controladas por um sinal de sincronismo (*clock*) cuja frequência pode variar de um mínimo de 2 MHz a um máximo de 50 MHz dependendo do estado-da-técnica.



107

## *clock*

- O *clock* é um sinal usado para coordenar as ações de dois ou mais circuitos eletrônicos. Um sinal de *clock* oscila entre os estados alto e baixo, normalmente usando um *duty cycle* de 50%, e gerando uma onda quadrada.
- Circuitos que usam o sinal de *clock* para sincronização podem se tornar ativos no ápice, na queda ou em ambos os momentos do sinal de *clock*.

108

- O sinal de *clock* geralmente possui duas transições e dois estados por período.

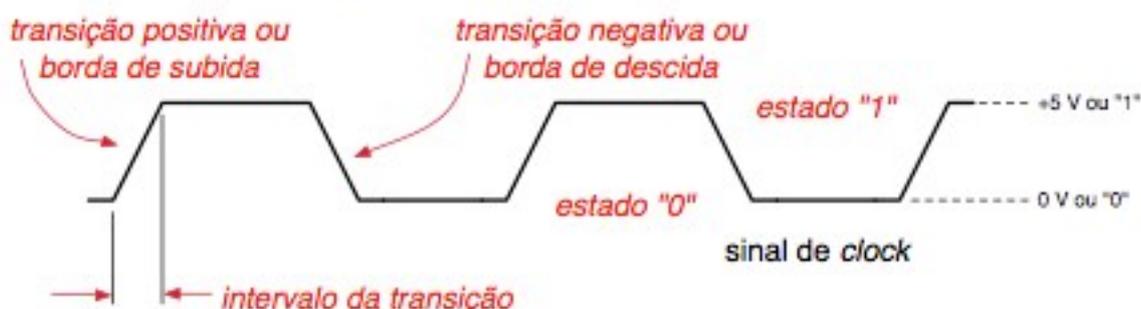


- Ao contrário dos componentes *assíncronos*, os componentes *síncronos* necessitam do sinal de *clock* para funcionar.

109

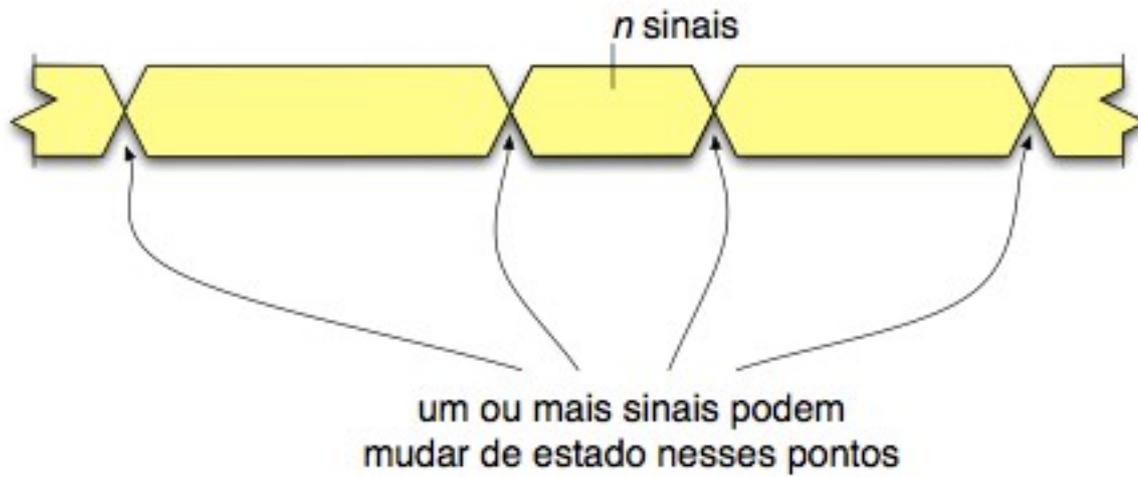
- Os níveis lógicos e as transições de um sinal de *clock* disparam mudanças de níveis em outros sinais, de forma a realizar seqüências de eventos lógicos. Essas seqüências de eventos podem ser vistas utilizando-se os *diagramas de tempo*.

- As transições de um sinal ocorrem dentro de um certo intervalo de tempo:



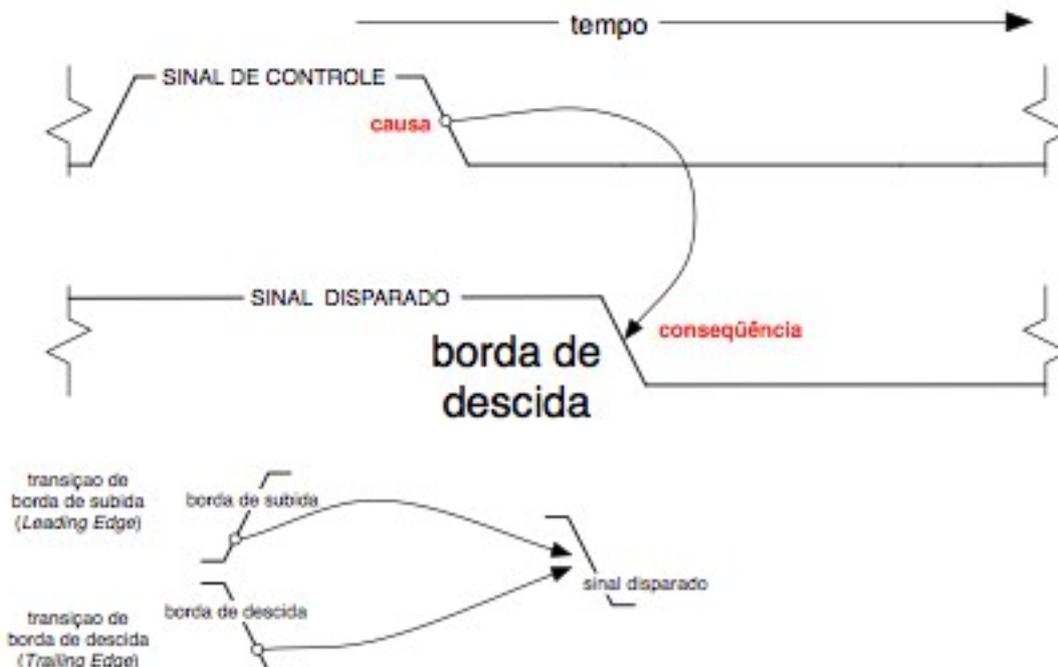
110

- Usamos a seguinte notação para representar as transições de níveis de múltiplos sinais:



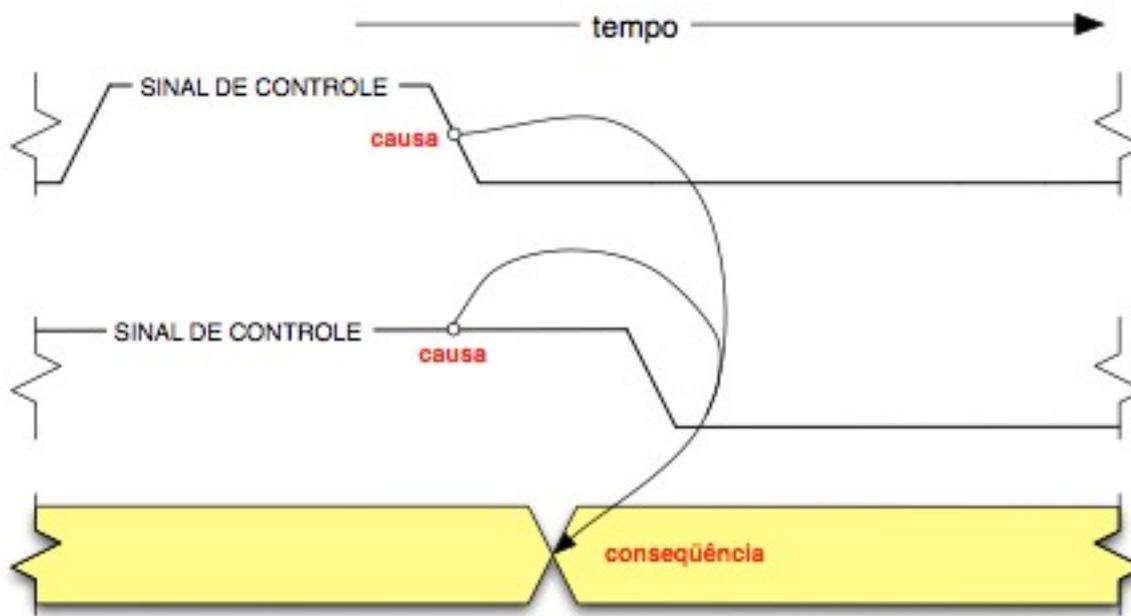
111

- A relação de causa e conseqüência entre sinais é identificada utilizando-se setas:



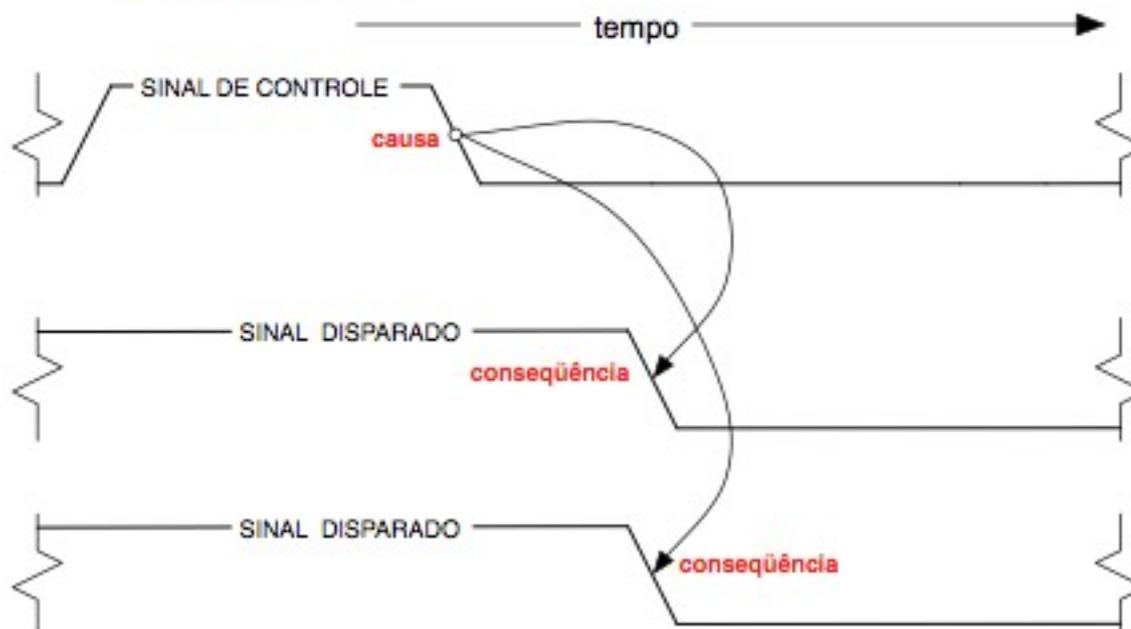
112

● Múltiplas causas podem gerar uma única consequência:



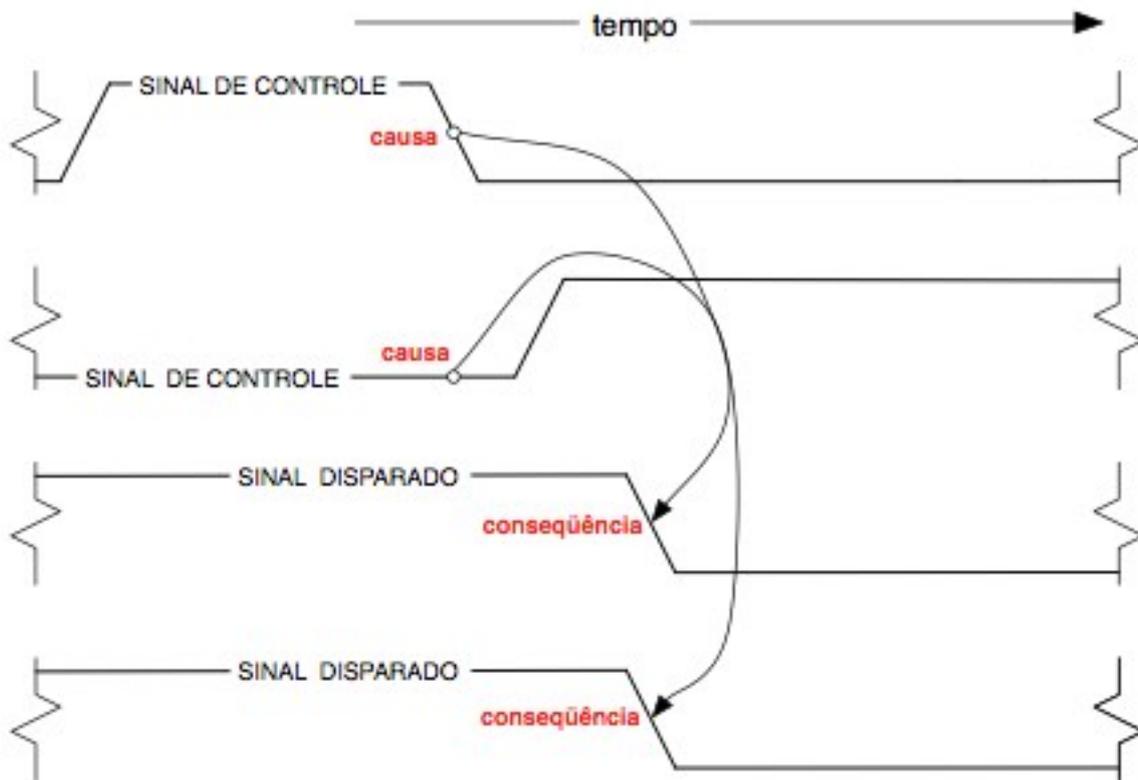
113

● Uma única causa pode gerar múltiplas consequências:



114

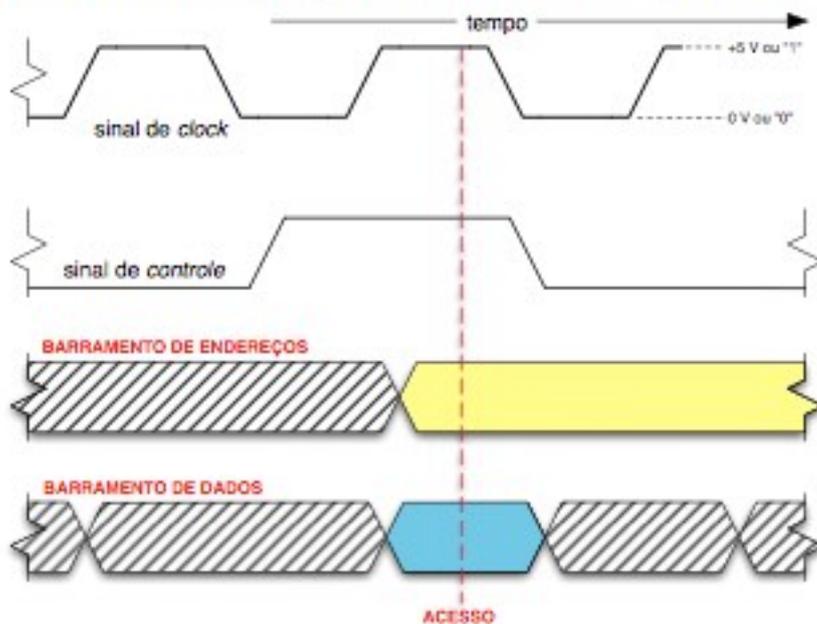
## Múltiplas causas podem ter múltiplas conseqüências:



115

## Temporização

- As CPUs usam o sinal de clock para determinar quando os dados podem ser transferidos.



116

