

# ELF52 - Sistemas Microcontrolados

## Revisão

**Professor:**

Prof. Marcos Eduardo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

# Revisão

# Propósito

- Até o momento, vocês viram muita programação para PC:

# Propósito

- Até o momento, vocês viram muita programação para PC:



Fonte: <http://goo.gl/vIWAZk>

- Mas é só isso que podemos fazer?

# Propósito

# Propósito

- Não!

# Propósito

- **Não!**
- Podemos desenvolver também para sistemas embarcados!



Fonte: <http://goo.gl/mXpwOH>

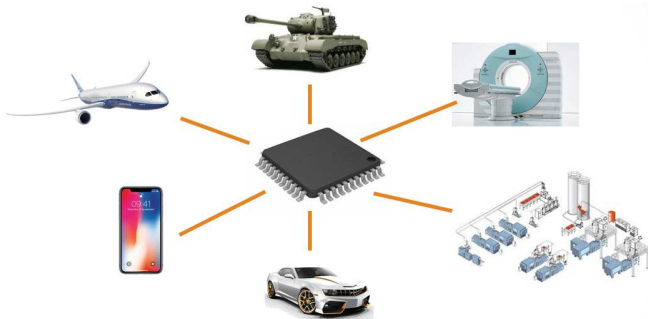
# Propósito

- Os microcontroladores estão presentes em quase todos equipamentos eletrônicos:
- Disciplina importante para a sequência do curso e profissional.



# Propósito

- Os microcontroladores estão presentes em quase todos equipamentos eletrônicos:



- Disciplina importante para a sequência do curso e profissional.

# Famílias Lógicas

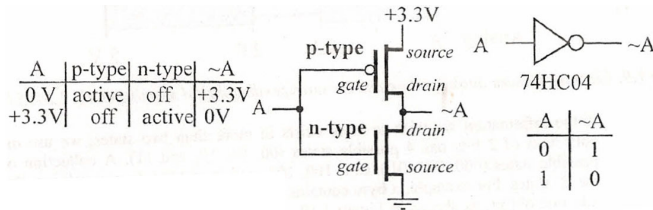
- Cada família lógica representam um conjunto de circuitos integrados implementados para cobrir um determinado grupo de funções lógicas que possuem características de fabricação e elétricas similares.
- As primeiras lógicas diferiam entre si essencialmente pelos respectivos circuitos integrados serem construídos com base em:
  - Transistores bipolares (TTL - Transistor-Transistor Logic);
  - CMOS (Complementary Metal-Oxide-Semiconductor).

# Famílias Lógicas

- Cada família lógica representam um conjunto de circuitos integrados implementados para cobrir um determinado grupo de funções lógicas que possuem características de fabricação e elétricas similares.
- As primeiras lógicas diferiam entre si essencialmente pelos respectivos circuitos integrados serem construídos com base em:
  - Transistores bipolares (TTL - Transistor-Transistor Logic);
  - CMOS (Complementary Metal-Oxide-Semiconductor).

## Exemplo de Componentes CMOS

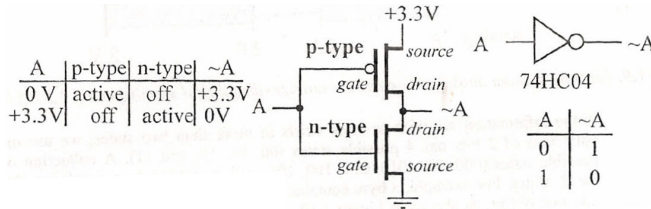
- Por exemplo, a porta lógica *NOT* pode ser criada com:



- Nesse circuito é apresentado o 74HC04, que é uma porta lógica CMOS NOT;
- A lógica por trás do circuito é bem simples:
  - Cada transistor atua como um interruptor entre a fonte e o dreno;
  - Para um transistor tipo p, o interruptor estará fechado (ativo) quando o porta (*gate*) estiver em nível baixo;
  - Para um transistor tipo n, o interruptor estará fechado (ativo) quando o porta (*gate*) estiver em nível alto.

# Exemplo de Componentes CMOS

- Por exemplo, a porta lógica *NOT* pode ser criada com:



- Nesse circuito é apresentado o 74HC04, que é uma porta lógica CMOS NOT;
- A lógica por trás do circuito é bem simples:
  - Cada transistor atua como um interruptor entre a fonte e o dreno;
  - Para um transistor tipo p, o interruptor estará fechado (ativo) quando o porta (*gate*) estiver em nível baixo;
  - Para um transistor tipo n, o interruptor estará fechado (ativo) quando o porta (*gate*) estiver em nível alto.

# Circuitos Digitais

# Circuitos Digitais

- Circuitos combinacionais:
  - Decodificador e Codificador;
  - Transcodificador ou Conversor de Código;
  - Multiplexador e Demultiplexador;
  - Somador;
  - Comparador.
- Circuitos sequenciais:
  - *Flip-flop*;
  - Registrador;
  - Memória.

# Circuitos Digitais

- Circuitos combinacionais:
  - Decodificador e Codificador;
  - Transcodificador ou Conversor de Código;
  - Multiplexador e Demultiplexador;
  - Somador;
  - Comparador.
- Circuitos sequenciais:
  - *Flip-flop*;
  - Registrador;
  - Memória.



# Circuitos Combinacionais

# Circuitos Combinacionais: Decodificador

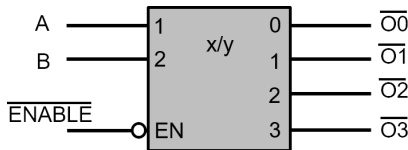
- Recebe **um conjunto de entradas** que representa um número binário e **ativa apenas a saída** que corresponde ao número recebido;
- É **imprescindível** para selecionar dispositivos (E/S ou memórias) que compartilham os mesmos recursos (barramentos e portas).

# Circuitos Combinacionais: Decodificador

- Recebe **um conjunto de entradas** que representa um número binário e **ativa apenas a saída** que corresponde ao número recebido;
- É **imprescindível** para selecionar dispositivos (E/S ou memórias) que compartilham os mesmos recursos (barramentos e portas).

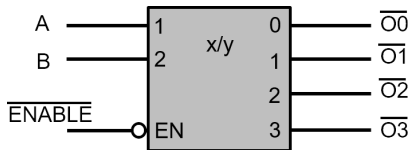
# Circuitos Combinacionais: Decodificador 2x4 (Ativo em baixa)

- Com a seguinte tabela verdade:
- O decodificador 2x4 pode ser representado como:



# Circuitos Combinacionais: Decodificador 2x4 (Ativo em baixa)

- O decodificador 2x4 pode ser representado como:
- Com a seguinte tabela verdade:

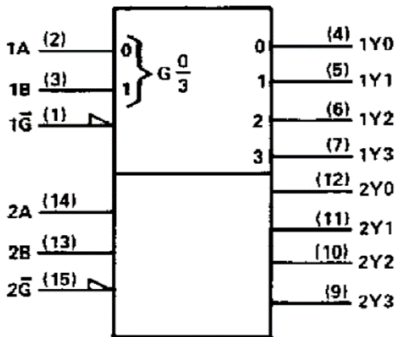


$\overline{EN}$	B	A	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

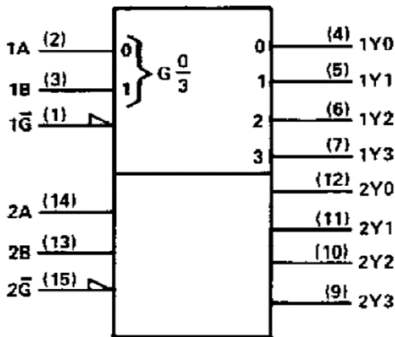
# Circuitos Combinacionais:

## Decodificador/Demux 74xx139

- Ativo em baixa;
- Dual 2x4;
- 1 pino de controle ( $\overline{G}$ ):



## Circuitos Combinacionais: Decodificador/Demux 74xx139



- Ativo em baixa;
- Dual 2x4;
- 1 pino de controle ( $\overline{G}$ ):

INPUTS			OUTPUTS			
ENABLE	SELECT					
$\overline{G}$	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = high level, L = low level, X = irrelevant

# Circuitos Combinacionais: Decodificador/Demux 74xx138

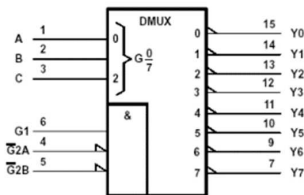
- Ativo em baixa;
- 3x8;
- 3 pinos de controle, ( $G1$ ,  $\overline{G2A}$ ,  $\overline{G2B}$ );



# Circuitos Combinacionais:

## Decodificador/Demux 74xx138

- Ativo em baixa;
- 3x8;
- 3 pinos de controle, ( $G1$ ,  $\overline{G2A}$ ,  $\overline{G2B}$ );



INPUTS					OUTPUTS								
ENABLE			SELECT										
G1	G2A	G2B	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	H	H	H	H	H	L	H	H	H
H	L	L	H	H	L	H	H	H	H	H	L	H	H
H	L	L	H	H	H	H	H	H	H	H	H	L	H

# Circuitos Combinacionais: Codificador

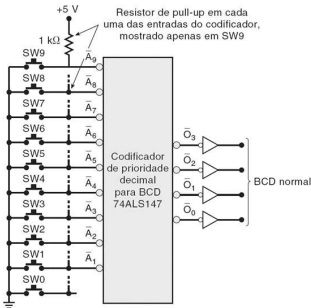
- Tem um certo **número de linhas de entrada**, em que somente uma é ativada por vez, e produz **um código de saída**:
  - Oposto do decodificador;
- Fornece na saída o número binário correspondente à entrada ativada. Somente uma entrada pode estar ativa ou deve-se ter um **codificador com prioridade**.

# Circuitos Combinacionais: Codificador

- Tem um certo **número de linhas de entrada**, em que somente uma é ativada por vez, e produz **um código de saída**:
  - Oposto do decodificador;
- Fornece na saída o número binário correspondente à entrada ativada. Somente uma entrada pode estar ativa ou deve-se ter um **codificador com prioridade**.

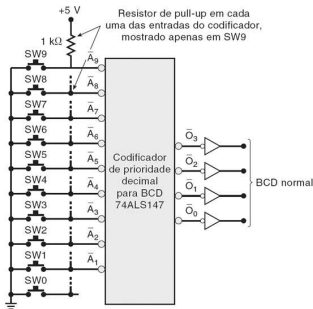
# Circuitos Combinacionais: Codificador com Prioridade

- 74xxx147;
- Exemplo de Utilização: Codificador de chaves decimal para BCD



# Circuitos Combinacionais: Codificador com Prioridade

- 74xxx147;
- Exemplo de Utilização: Codificador de chaves decimal para BCD



$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	$\bar{A}_8$	$\bar{A}_9$	$\bar{O}_3$	$\bar{O}_2$	$\bar{O}_1$	$\bar{O}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

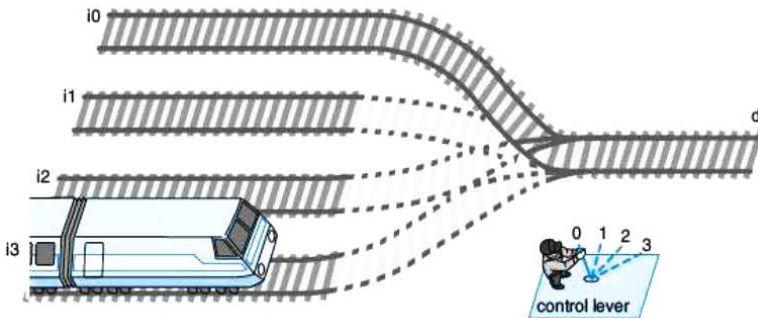
X = 0 ou 1

# Multiplexador

- O que é um Multiplexador?

# Multiplexador

- O que é um Multiplexador?



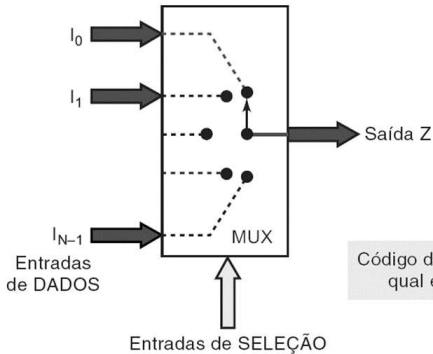
# Multiplexador

- Multiplexador (MUX) ou **Seletor** de dados:
  - Seleciona **uma de N fontes de entrada** de dados e transmite os dados selecionados para **uma única saída**.



# Multiplexador

- Multiplexador (MUX) ou **Seletor** de dados:
  - Seleciona **uma de N fontes de entrada** de dados e transmite os dados selecionados para **uma única saída**.



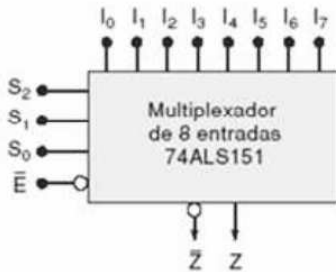
Código de entrada de SELEÇÃO determina qual entrada é transmitida à saída Z

# Multiplexador

- Multiplexador de 8 entradas (74xxx151):

# Multiplexador

- Multiplexador de 8 entradas (74xxx151):



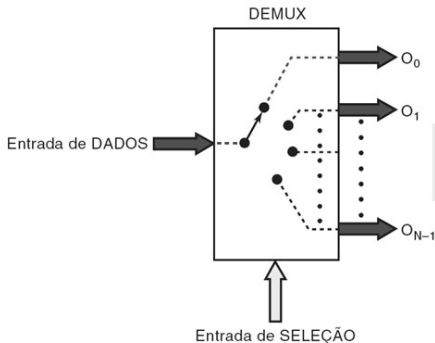
Entradas				Saídas	
$\bar{E}$	$S_2$	$S_1$	$S_0$	$\bar{Z}$	$Z$
H	X	X	X	H	L
L	L	L	L	$\bar{I}_0$	$I_0$
L	L	L	H	$\bar{I}_1$	$I_1$
L	L	H	L	$\bar{I}_2$	$I_2$
L	L	H	H	$\bar{I}_3$	$I_3$
L	H	L	L	$\bar{I}_4$	$I_4$
L	H	L	H	$\bar{I}_5$	$I_5$
L	H	H	L	$\bar{I}_6$	$I_6$
L	H	H	H	$\bar{I}_7$	$I_7$

# Demultiplexador

- Demultiplexador (DEMUX) ou Distribuidor de dados:
  - Recebe uma **única entrada** e a distribui para **várias saídas**;
  - O código de entrada de seleção determina para qual saída os dados de entrada serão transmitidos:

# Demultiplexador

- Demultiplexador (DEMUX) ou Distribuidor de dados:
  - Recebe uma **única entrada** e a distribui para **várias saídas**;
  - O código de entrada de seleção determina para qual saída os dados de entrada serão transmitidos:



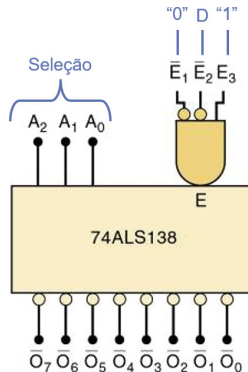
A entrada de DADOS é transmitida a apenas uma das saídas, como determinado pelo código de seleção de entrada.

# Demultiplexador

- 74xx138:
  - Além de poder ser usado como decodificador, também pode ser usado como DEMUX;
  - O dado de entrada é o resultado  $\bar{E}_1\bar{E}_2E_3$ :

# Demultiplexador

- 74xx138:
  - Além de poder ser usado como decodificador, também pode ser usado como DEMUX;
  - O dado de entrada é o resultado  $\bar{E}_1\bar{E}_2E_3$ :



# Circuitos Sequenciais



# Latch

- Circuito para armazenamento de dados que trabalha com os **níveis** de entrada:
- Tipos:
  - *Latch SR (Set/Reset)*
  - *Gated Latch D (Latch Transparente)*

# Latch

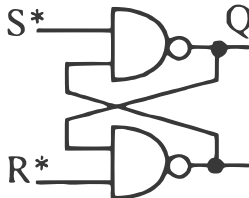
- Circuito para armazenamento de dados que trabalha com os **níveis** de entrada:
- Tipos:
  - *Latch SR (Set/Reset)*
  - *Gated Latch D (Latch Transparente)*

## Latch SR (Set/Reset)

- O *Latch SR* apresenta a forma mais simples de armazenamento;
  - Quando  $S^* = 1$  e  $R^* = 1$ ,  $Q$  "lembrará" o último estado;
  - Quando  $S^* = 0$  e  $R^* = 1$ , temos  $Q = 1$ ;
  - Quando  $S^* = 1$  e  $R^* = 0$ , temos  $Q = 0$ ;
  - Quando  $S^* = 0$  e  $R^* = 0$ , temos  $Q$  indefinido.

## Latch SR (Set/Reset)

- O *Latch SR* apresenta a forma mais simples de armazenamento;
- Quando  $S^* = 1$  e  $R^* = 1$ ,  $Q$  "lembrará" o último estado;
- Quando  $S^* = 0$  e  $R^* = 1$ , temos  $Q = 1$ ;
- Quando  $S^* = 1$  e  $R^* = 0$ , temos  $Q = 0$ ;
- Quando  $S^* = 0$  e  $R^* = 0$ , temos  $Q$  indefinido.

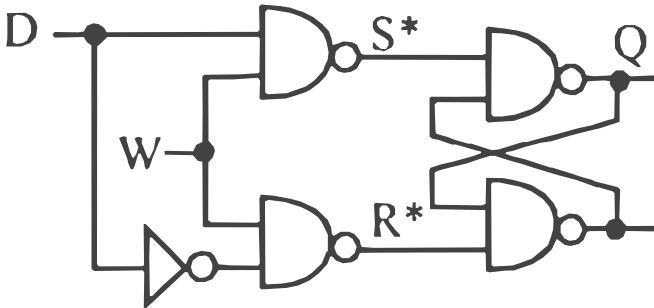


# Gated D Latch

- O *Latch* Transparente:
  - Tem duas entradas, Enable (W) e D, e a(s) saída(s) Q (e  $\overline{Q}$ );
  - Quando W está em nível alto, o estado de D é transferido para Q;
  - Quando W está em nível baixo, Q não é alterado.

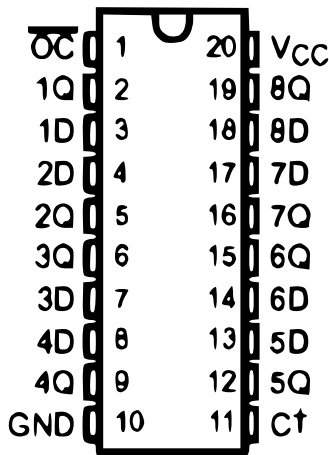
## Gated D Latch

- O *Latch* Transparente:
  - Tem duas entradas, Enable ( $W$ ) e  $D$ , e a(s) saída(s)  $Q$  (e  $\overline{Q}$ );
  - Quando  $W$  está em nível alto, o estado de  $D$  é transferido para  $Q$ ;
  - Quando  $W$  está em nível baixo,  $Q$  não é alterado.



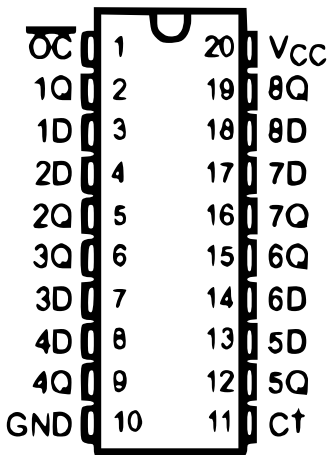
## Gated D Latch: 74xx373

- 74xx373: *Octal flip-flop D* com saídas 3-state



## Gated D Latch: 74xx373

- 74xx373: Octal flip-flop D com saídas 3-state



EN	D	Q - Saída
0	X	Não muda
1	0	Q=0
1	1	Q=1

X → Irrelevante



# Flip-flop

- Circuito que trabalha na **borda** de um sinal de entrada;
- Tipos de Flip-flops:
  - Flip-Flop SR (Set/Reset);
  - Flip-Flop D (Data);
  - Flip-Flop JK: equivalente ao Flip-Flop SR, mas também inverte o estado quando ambas as entradas estão ativas;
  - Flip-Flop T: inverte o estado quando a entrada está alta e ocorre um pulso de *clock*.

# Flip-flop

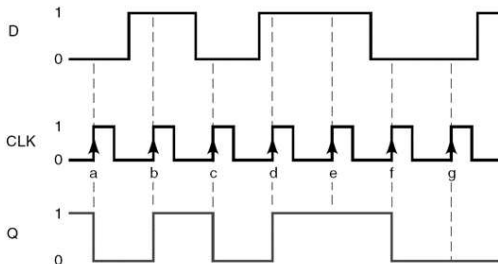
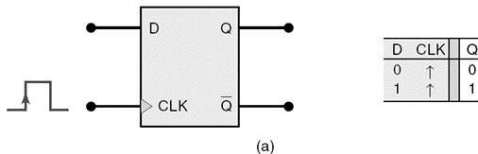
- Circuito que trabalha na **borda** de um sinal de entrada;
- Tipos de Flip-flops:
  - Flip-Flop SR (Set/Reset);
  - Flip-Flop D (Data);
  - Flip-Flop JK: equivalente ao Flip-Flop SR, mas também inverte o estado quando ambas as entradas estão ativas;
  - Flip-Flop T: inverte o estado quando a entrada está alta e ocorre um pulso de *clock*.

## Flip-flop D

- A saída muda para o valor da entrada na borda do clock;
- Utilizado para transferência de dados:

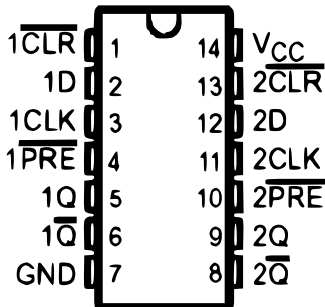
## Flip-flop D

- A saída muda para o valor da entrada na borda do clock;
- Utilizado para transferência de dados:



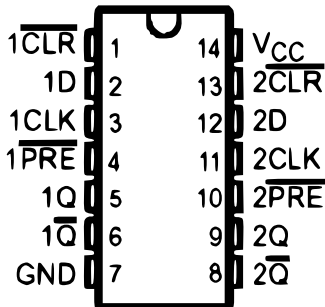
## Flip-flop D

- 74xx74;
- PRESET e CLEAR assíncronos;



## Flip-flop D

- 74xx74;
- PRESET e CLEAR assíncronos;



INPUTS				OUTPUTS	
$\overline{\text{PRE}}$	$\overline{\text{CLR}}$	CLK	D	Q	$\overline{\text{Q}}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H $\ddagger$	H $\ddagger$
H	H	$\uparrow$	H	H	L
H	H	$\uparrow$	L	L	H
H	H	L	X	Q <sub>0</sub>	$\overline{\text{Q}}_0$

$\ddagger$  This configuration is nonstable; that is, it does not persist when  $\overline{\text{PRE}}$  or  $\overline{\text{CLR}}$  returns to its inactive (high) level.

# Registrador

- Agrupamento de *flip-flops* utilizado para armazenamento e transferência de dados;
- Tanto a entrada como a saída podem ser em série ou paralelo.

# Registrador

- Agrupamento de *flip-flops* utilizado para armazenamento e transferência de dados;
- Tanto a entrada como a saída podem ser em série ou paralelo.

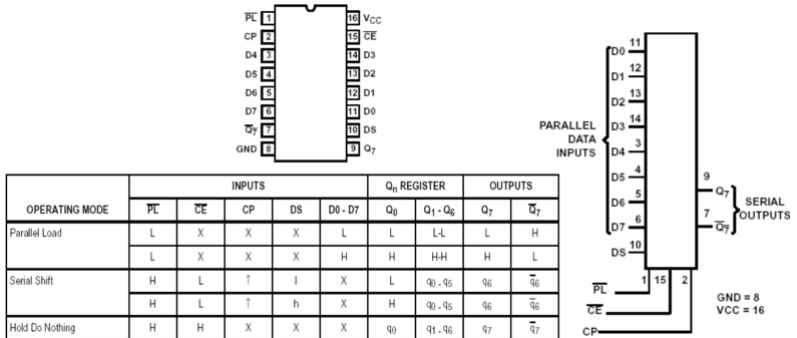


# Registrador

- Registrador de deslocamento 74xx165:
  - Conversor **paralelo/serial** de 8 bits.

# Registrador

- Registrador de deslocamento 74xx165:
  - Conversor **paralelo/serial** de 8 bits.



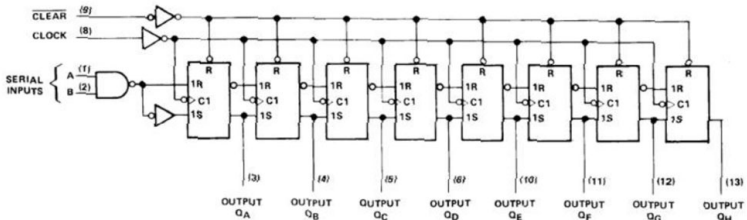
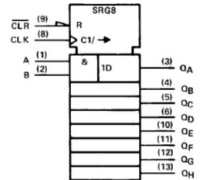
# Registrador

- Registrador de deslocamento 74xx164:
  - Conversor **serial/paralelo** de 8 bits com *clear* assíncrono:

# Registrador

- Registrador de deslocamento 74xx164:
  - Conversor **serial/paralelo** de 8 bits com *clear* assíncrono:


INPUTS				OUTPUTS			
CLR	CLOCK	A	B	Q <sub>A</sub>	Q <sub>B</sub>	...	Q <sub>H</sub>
L	X	X	X	L	L	L	L
H	L	X	X	Q <sub>A0</sub>	Q <sub>B0</sub>	Q <sub>H0</sub>	
H	↑	H	H	H	Q <sub>An</sub>	Q <sub>Gn</sub>	
H	↑	L	X	L	Q <sub>An</sub>	Q <sub>Gn</sub>	
H	↑	X	L	L	Q <sub>An</sub>	Q <sub>Gn</sub>	



# Memória RAM

- A memória RAM (volátil) pode ser dividida em:
  - DRAM (*Dynamic Random Access Memory*): Bits de dados são armazenados apenas com um transistor e um capacitor;
  - SRAM (*Static Random Access Memory*): Bits de dados são armazenados em *flip-flops*.
- Obviamente DRAMs são mais baratas de produzir, já que o conjunto de um transistor e um capacitor é muito mais simples e tem menos custo que um *flip-flop*. Tipicamente utilizadas como memória RAM de computadores pessoais;
- Porém, SRAMs normalmente são mais rápidas e gastam menos energia que as DRAMs;
- Além disto, as DRAMs necessitam de controladores, que "refresquem" (*refresh*) o estado dos capacitores, já que eles perdem sua carga a cada 2ms aproximadamente.

# Memória ROM

- A memória ROM (*Read Only Memory*) (não-volátil) pode ser dividida em:
  - PROM (*Programmable read-only memory*): Só pode ser gravada uma única vez, com um programador de PROM;
  - EPROM (*Erasable programmable read-only memory*): Pode ser regravada (número limitado), porém antes precisa ser apagada através de exposição à luz ultravioleta  ;
  - EEPROM (*Electrically erasable programmable read-only memory*): Pode ser regravada (número limitado), mas o processo é feito eletricamente. As memórias Flash são EEPROM modernas.
- Cada bit é armazenado através de um arranjo de transistores, que combinados permitem que o dado fique armazenado mesmo na ausência de alimentação;
- Alterar dados é lento e difícil, ou simplesmente não existe.

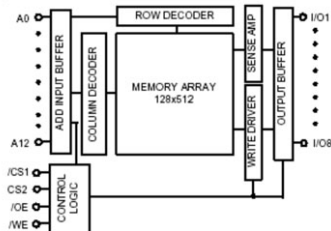
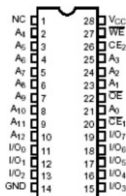
# Memória

- RAM Estática 6264:
  - 8k x 8 SRAM (8192 bytes);
  - Tempo de acesso: 12..150 ns.

# Memória

- RAM Estática 6264:
  - 8k × 8 SRAM (8192 bytes);
  - Tempo de acesso: 12..150 ns.

/CS1	CS2	/WE	/OE	MODE	I/O OPERATION
H	X	X	X	Standby	High-Z
X	L	X	X		High-Z
L	H	H	H	Output Disabled	High-Z
L	H	H	L	Read	Data Out
L	H	L	X	Write	Data In



Pin Name	Pin Function	Pin Name	Pin Function
/CS1	Chip Select 1	I/O1-I/O8	Data Input/Output
CS2	Chip Select 2	Vcc	Power(+5V)
/WE	Write Enable	Vss	Ground
/OE	Output Enable	NC	No Connect
Address Inputs			

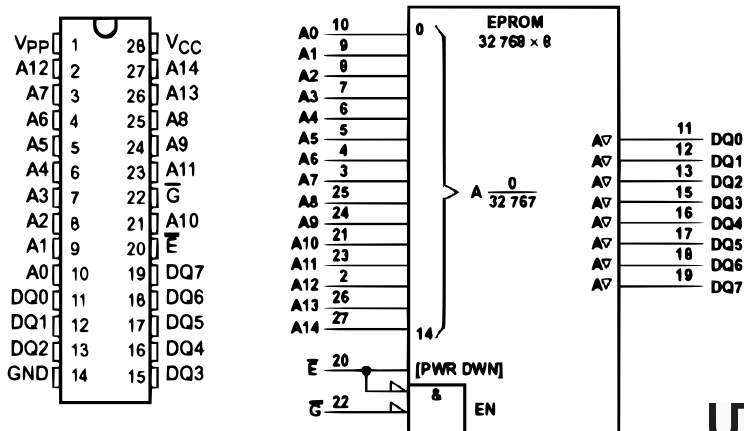


# Memória

- ROM Programável 27C256:
  - 32k x 8 EPROM (32768 bytes);
  - Tempos de acesso: 100..250 ns

# Memória

- ROM Programável 27C256:
  - 32k x 8 EPROM (32768 bytes);
  - Tempos de acesso: 100..250 ns



# Memória...



# Dúvidas?