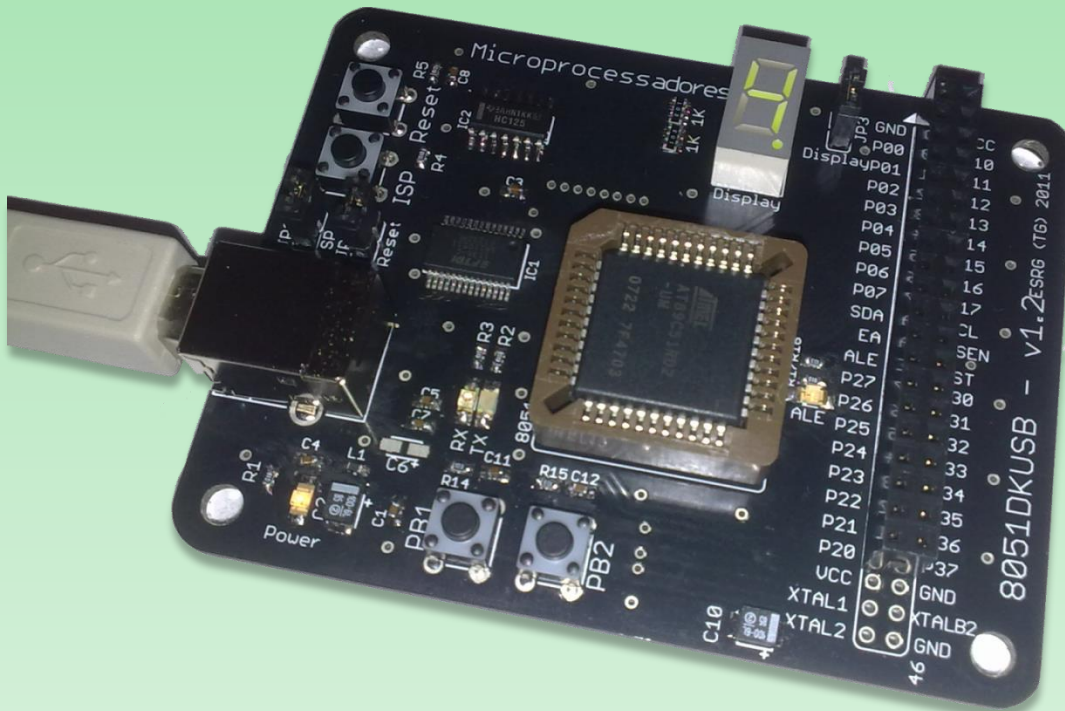
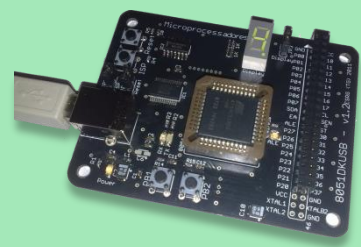


Mestrado Integrado em Eng. Engenharia de Telecomunicações e Informática



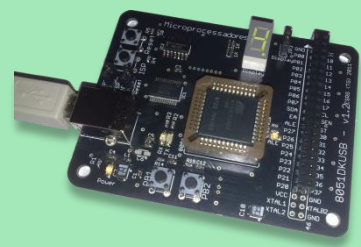
Revisões

Microcontroladores
2º Ano – A02



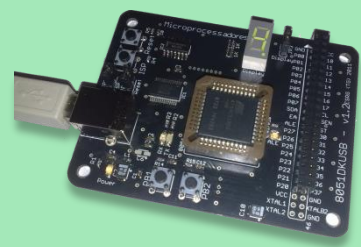
Revisão: Representação de números e códigos

- Representação de números:
 - Números decimais: *base 10*
 - O peso de cada dígito é 10 vezes maior que o dígito à sua direita.
 - $2200_{10} = 2 \times 10^3 + 2 \times 10^2 + 0 \times 10^1 + 0 \times 10^0$
 - Multiplicar por 10, 100 e 1000
 - Dividir por 10, 100 e 1000



Revisão: Representação de números e códigos

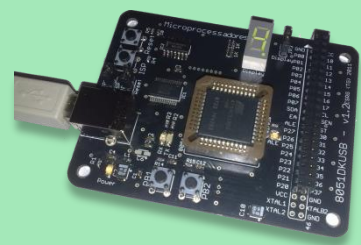
- Números binários: base 2
 - O peso de cada dígito é 2 vezes maior que o dígito à sua direita.
 - $1100_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
 $= 8 + 4 + 0 + 0$
 $= 12_{10}$
- Números octais: *base 8*
 - O peso de cada dígito é 8 vezes maior que o dígito à sua direita.
 - $2702_8 = 2 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 2 \times 8^0$
 $= 1024 + 448 + 0 + 2$
 $= 1474_{10}$
- Números hexadecimais: base 16
 - O peso de cada dígito é 16 vezes maior que o dígito à sua direita.
 - $2A0C_{16} = 2 \times 16^3 + 10 \times 16^2 + 0 \times 16^1 + 12 \times 16^0$
 $= 8192 + 2560 + 0 + 12$
 $= 10764_{10}$



Revisão: Representação de números e códigos

Table 3. Octal, Binary, and Hexadecimal Equivalents

Octal	Direct Binary	8-Bit Binary	Hexadecimal
000	000 000 000	0000 0000	\$00
001	000 000 001	0000 0001	\$01
002	000 000 010	0000 0010	\$02
003	000 000 011	0000 0011	\$03
004	000 000 100	0000 0100	\$04
005	000 000 101	0000 0101	\$05
006	000 000 110	0000 0110	\$06
007	000 000 111	0000 0111	\$07
010	000 001 000	0000 1000	\$08
011	000 001 001	0000 1001	\$09
012	000 001 010	0000 1010	\$0A
013	000 001 011	0000 1011	\$0B
014	000 001 100	0000 1100	\$0C
015	000 001 101	0000 1101	\$0D
016	000 001 110	0000 1110	\$0E
017	000 001 111	0000 1111	\$0F
101	001 000 001	0100 0001	\$41
125	001 010 101	0101 0101	\$55
252	010 101 010	1010 1010	\$AA
377	011 111 111	1111 1111	\$FF



Revisão: Representação de números e códigos

- Código ASCII

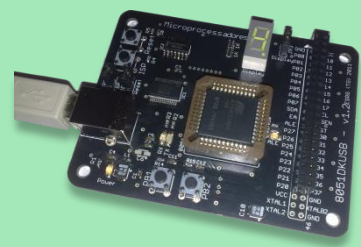
- Os computadores tem que tratar diferentes tipos de informação, tais como números e letras.
- Esta informação tem que ser codificada de modo a que os computadores possam trocar dados entre si e ainda serem capazes de os interpretar.
- O código mais usado é *American Standard Code for Information Interchange* (ASCII).
- O código ASCII codifica os caracteres (números, letras e símbolos, tais como '?') em códigos binários de *7-bits*.
 - Na prática o código ocupa um *byte* onde o *bit* mais significativo é nulo.
- Nota:
 - Actualmente a necessidade de trocar informação entre diferentes idiomas que exigem a utilização de muitos mais símbolos, o código ASCII é insuficiente. Por esse motivo a maioria das aplicações utiliza o *UNICODE* (~105 mil caracteres).

Revisão: Representação de números e códigos

Table 2. ASCII to Hexadecimal Conversion

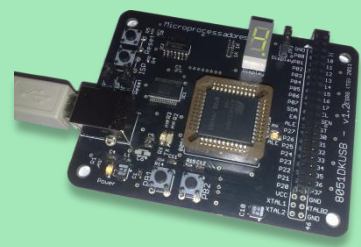
Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII
\$00	NUL	\$20	SP space	\$40	@	\$60	` grave
\$01	SOH	\$21	!	\$41	A	\$61	a
\$02	STX	\$22	"	\$42	B	\$62	b
\$03	ETX	\$23	#	\$43	C	\$63	c
\$04	EOT	\$24	\$	\$44	D	\$64	d
\$05	ENQ	\$25	%	\$45	E	\$65	e
\$06	ACK	\$26	&	\$46	F	\$66	f
\$07	BEL beep	\$27	' apost.	\$47	G	\$67	g
\$08	BS back sp	\$28	(\$48	H	\$68	h
\$09	HT tab	\$29)	\$49	I	\$69	i
\$0A	LF linefeed	\$2A	*	\$4A	J	\$6A	j
\$0B	VT	\$2B	+	\$4B	K	\$6B	k
\$0C	FF	\$2C	, comma	\$4C	L	\$6C	l
\$0D	CR return	\$2D	- dash	\$4D	M	\$6D	m

\$0E	SO	\$2E	. period	\$4E	N	\$6E	n
\$0F	SI	\$2F	/	\$4F	O	\$6F	o
\$10	DLE	\$30	0	\$50	P	\$70	p
\$11	DC1	\$31	1	\$51	Q	\$71	q
\$12	DC2	\$32	2	\$52	R	\$72	r
\$13	DC3	\$33	3	\$53	S	\$73	s
\$14	DC4	\$34	4	\$54	T	\$74	t
\$15	NAK	\$35	5	\$55	U	\$75	u
\$16	SYN	\$36	6	\$56	V	\$76	v
\$17	ETB	\$37	7	\$57	W	\$77	w
\$18	CAN	\$38	8	\$58	X	\$78	x
\$19	EM	\$39	9	\$59	Y	\$79	y
\$1A	SUB	\$3A	:	\$5A	Z	\$7A	z
\$1B	ESCAPE	\$3B	;	\$5B	[\$7B	{
\$1C	FS	\$3C	<	\$5C	\	\$7C	
\$1D	GS	\$3D	=	\$5D]	\$7D	}
\$1E	RS	\$3E	>	\$5E	^	\$7E	~
\$1F	US	\$3F	?	\$5F	_ under	\$7F	DEL delete



Revisão: Unidades de memória

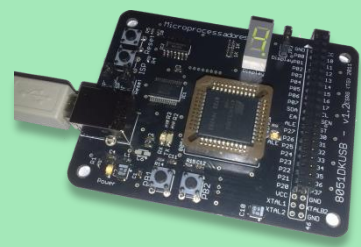
- Num computador a menor unidade de memória é o *bit* que pode armazenar os valores '0' e '1'.
- Os *bits* podem ser agrupados em
 - grupos de quatro *bits* formando um '*nibble*'.
 - grupos de oito *bits* formando um '*byte*'.
 - um *byte* tem dois *nibbles*.
 - grupos de 16 ou 32 *bits* formando uma '*word*'.
 - o tamanho de uma *word* varia de processador para processador.



Revisão:

- Prefixos:

- Dizemos que 1000 ohm = 1k ohm,
- Contudo, 1k *byte* são 1024 *bytes* e não 1000 *bytes* !!!
- Temos então que:
 - 1.000 : 1k \leftrightarrow 2^{10} : 1024₁₀ : 1k byte = 1KB
 - 1.000.000 : 1M \leftrightarrow 2^{20} : 1.048.576 bytes : 1M byte
 - 1.000.000.000 : 1G \leftrightarrow 2^{30} : 1.073.741.824 bytes : 1G byte

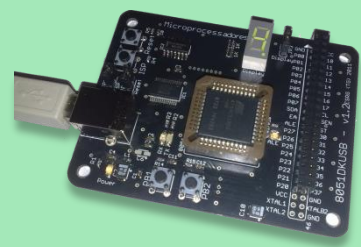


Problema

- Pretende-se desenvolver um programa que gere o resultado da equação de uma recta:

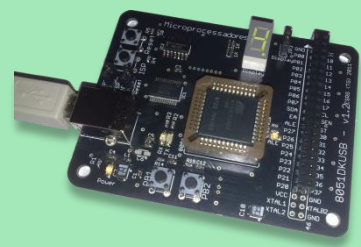
$$y=4*x+1$$

- Quais são as instruções?
 - Quais são as variáveis?
-
- *As instruções são comandos enviados ao CPU de modo a resolver a equação da recta, com base no valor da variável de entrada **x**. A equação é resolvida e o resultado é armazenado na variável de saída **y**.*
 - *Onde armazenar as variáveis de entrada e de saída?*
 - *Onde armazenar as instruções?*
 - *Qual o maior valor possível para **y**, sabendo que **x** tem 8-bit?*
 - *Se **y** estiver limitado a 8-bit, qual o maior valor de **x**?*



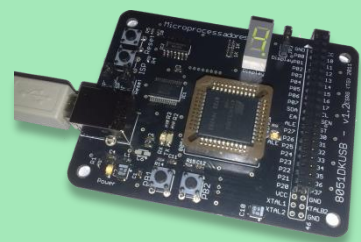
von Neumann

- Publicou um relatório incompleto sobre o primeiro computador (o ENIAC) “*First Draft of a Report on the EDVAC*”;
- Esse relatório contém os fundamentos de uma arquitectura, designada como arquitectura de von Neumann. No relatório propunha que as operações a executar (programa) fossem codificadas e armazenadas em memória. Na altura o ENIAC utilizava ligações eléctricas para definir as operações a executar;
- No mesmo relatório sugeriu ainda que os códigos (das operações a executar) utilizassem o sistema binário (ligado/desligado);
- Sugeriu ainda que o processamento dos códigos binários, que representam as operações, fossem processados sequencialmente, bit a bit: similar a **máquina de estados**.



Sistema computadorizado

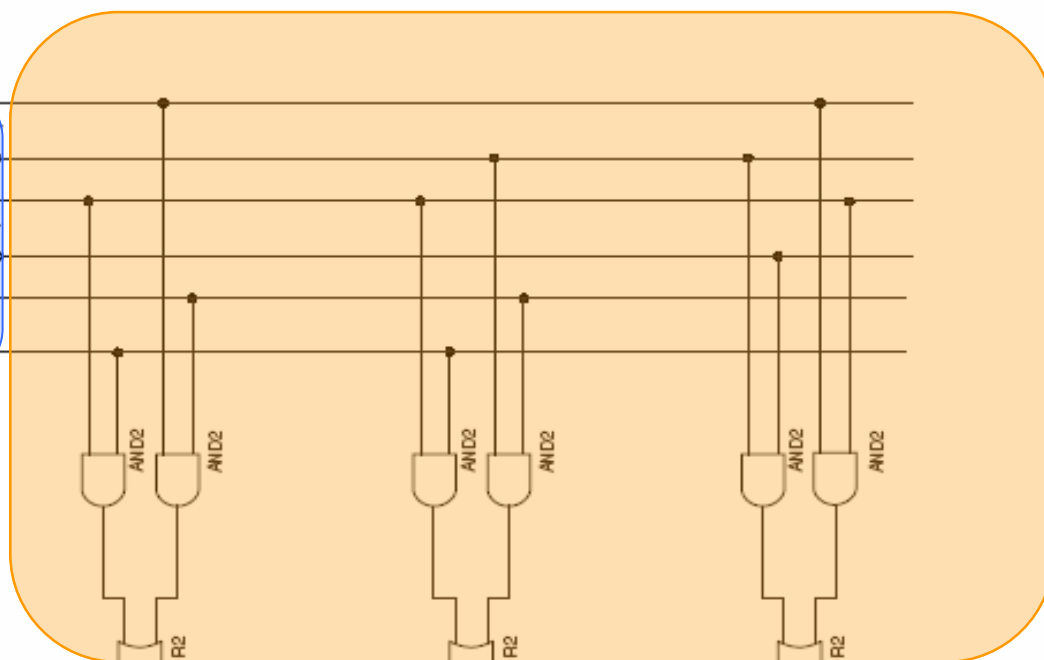
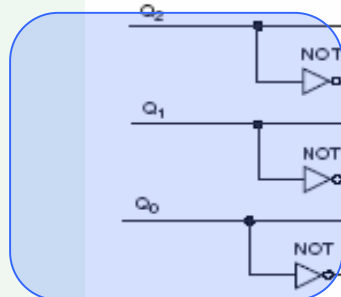
- Sistema capaz de executar um programa residente em memória;
- Principais blocos constituintes:
 - Memória
 - CPU – Unidade de Processamento Central
 - **ALU** – Unidade Lógica e Aritmética;
 - **CTU** – Unidade de Controlo
 - Periféricos:
 - Unidades de E/S (Entrada/Saída).



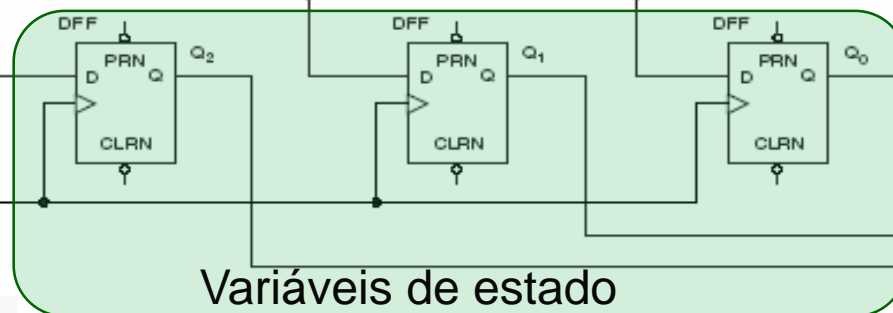
Revisão: O que são máquinas de estado ?

Circuito combinacional

Entradas/Saídas

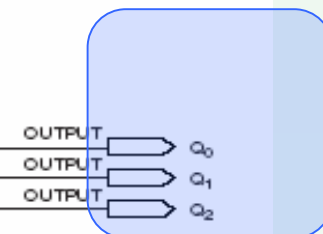


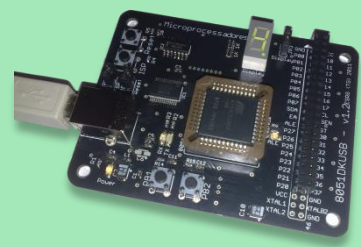
CLK INPUT



Variáveis de estado

Entradas/Saídas





CPU

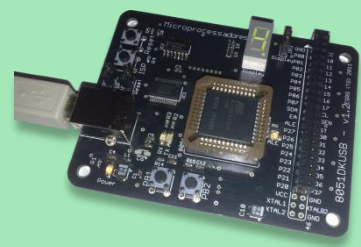
Unidade de Processamento Central:

➤ ALU

- Circuito lógico combinacional;
- Suporta um conjunto pré-definido de instruções aritméticas e lógicas, descritas na arquitectura do conjunto de instruções (ISA) do CPU.

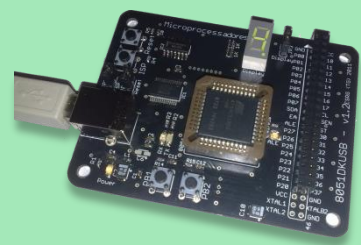
➤ CTU

- Circuito lógico sequencial;
- Realiza a sequência de operações necessárias à execução de cada uma das instruções especificadas na ISA.



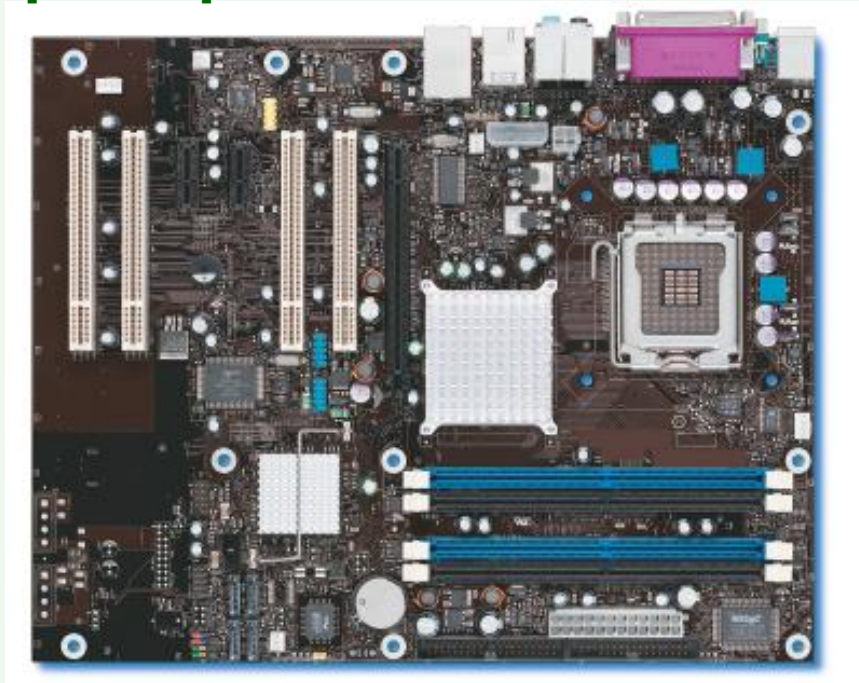
Micro...

- Microprocessador (μP)
 - Um único Circuito Integrado (CI) composto pelo CPU;
 - Exemplos: 8088, x386 (Pentium), PowerPC.
- Microcomputador
 - Computador onde o CPU é um μP ;
 - Exemplos: IBM PC, Apple Macintosh.
- Microcontrolador (μC)
 - CPU+Memória+I/O num único CI;
 - Exemplos: 4004, 8051, ATTiny, PIC16F84.

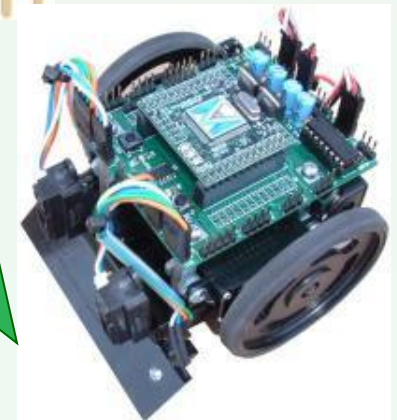
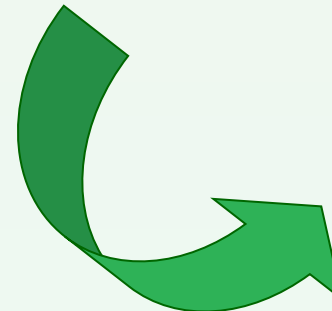
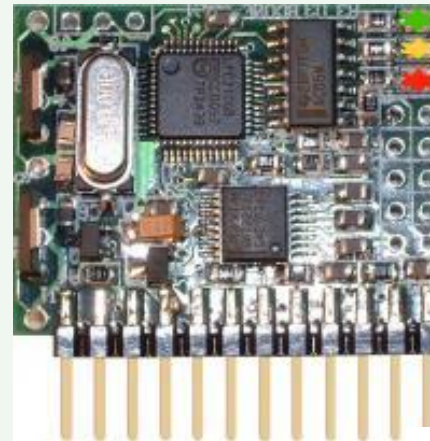


Microprocessador X Microcontrolador

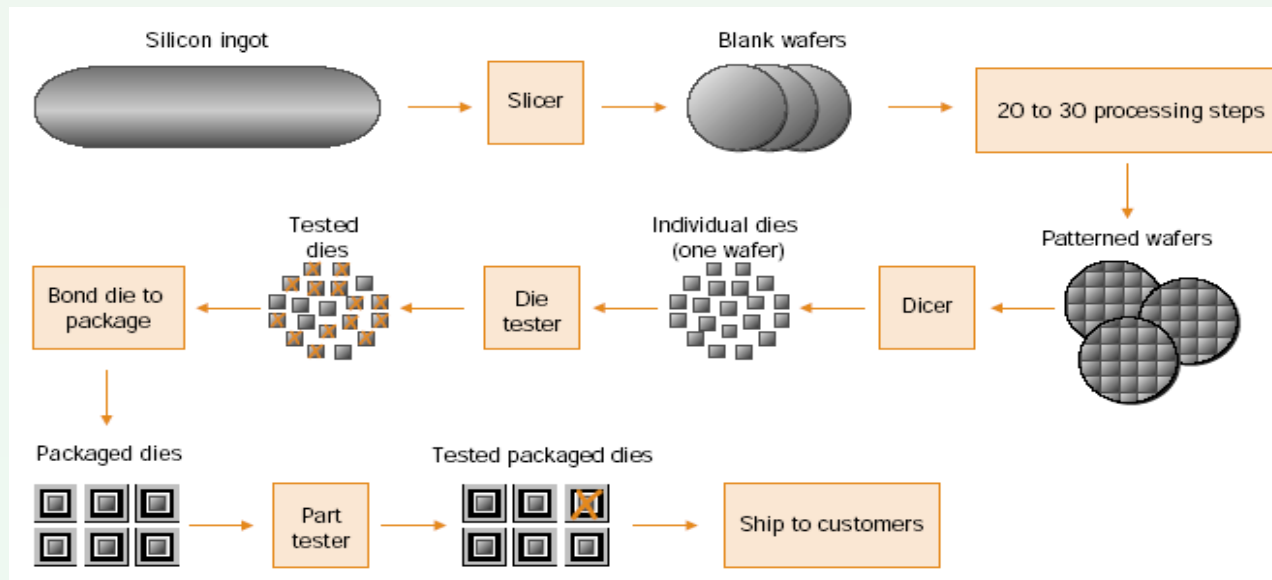
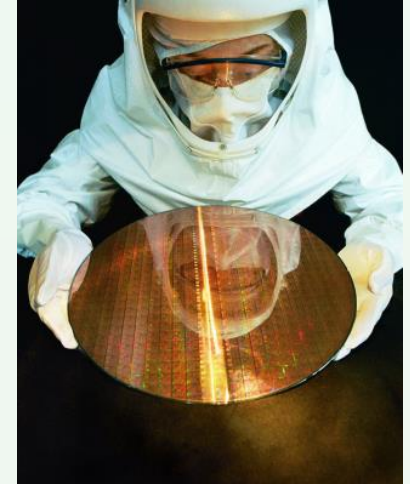
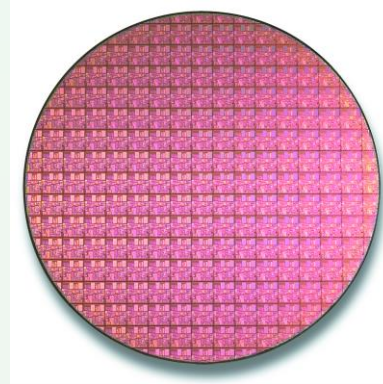
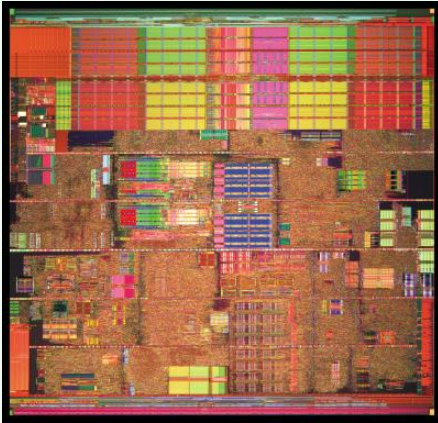
Sistema baseado em microprocessador
[Intel P4]



Sistema baseado em microcontrolador
[ARM]



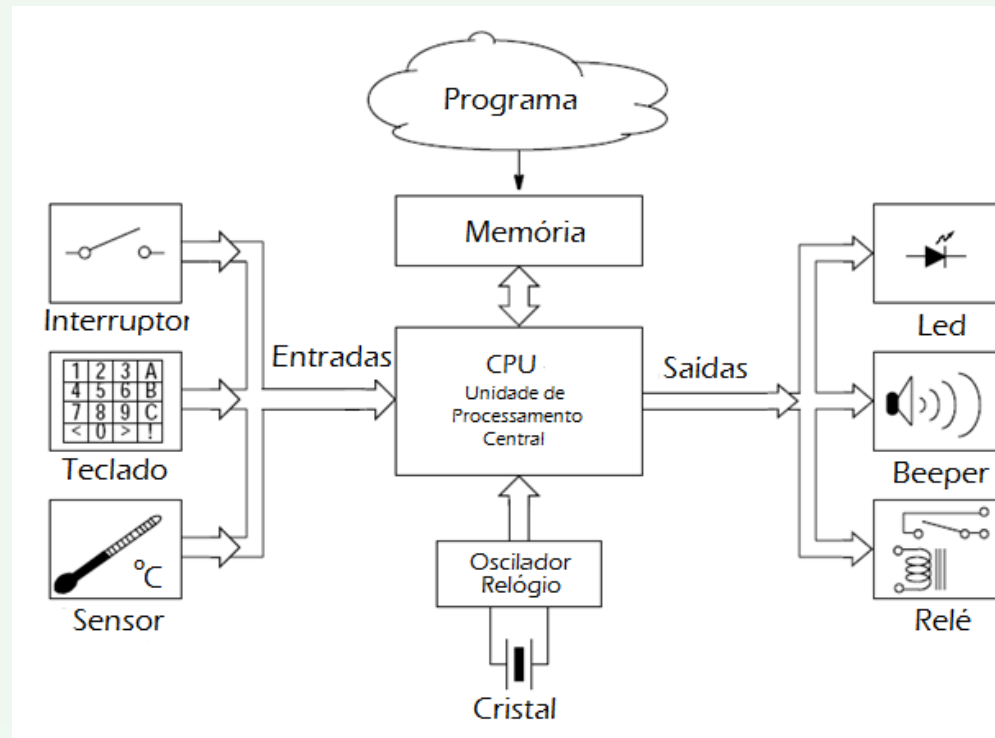
Processo de fabrico

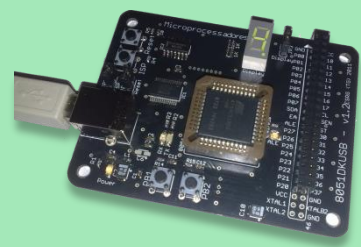




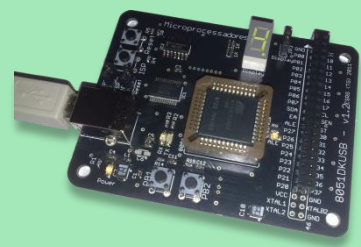
Arquitetura von Neumann

- O programa a executar sequencialmente está armazenado numa memória e é constituído por instruções que estão definidas no ISA e são suportadas pela sua Unidade de Processamento Central (CPU).

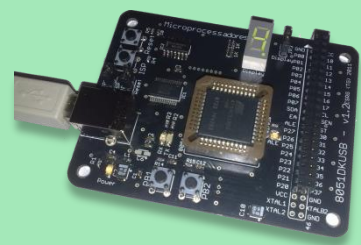




- O que distingue um microprocessador de um microcontrolador ?
 - Microprocessador
 - Este para ser usado precisa que outros componentes sejam adicionados externamente, tais como memória e periféricos de interface Entrada/Saída (E/S).
 - Microcontrolador
 - Desenhado de modo a ter todas as funcionalidades integradas num único circuito. Ou seja, dentro do chip são colocados todos os componentes/periféricos necessários.

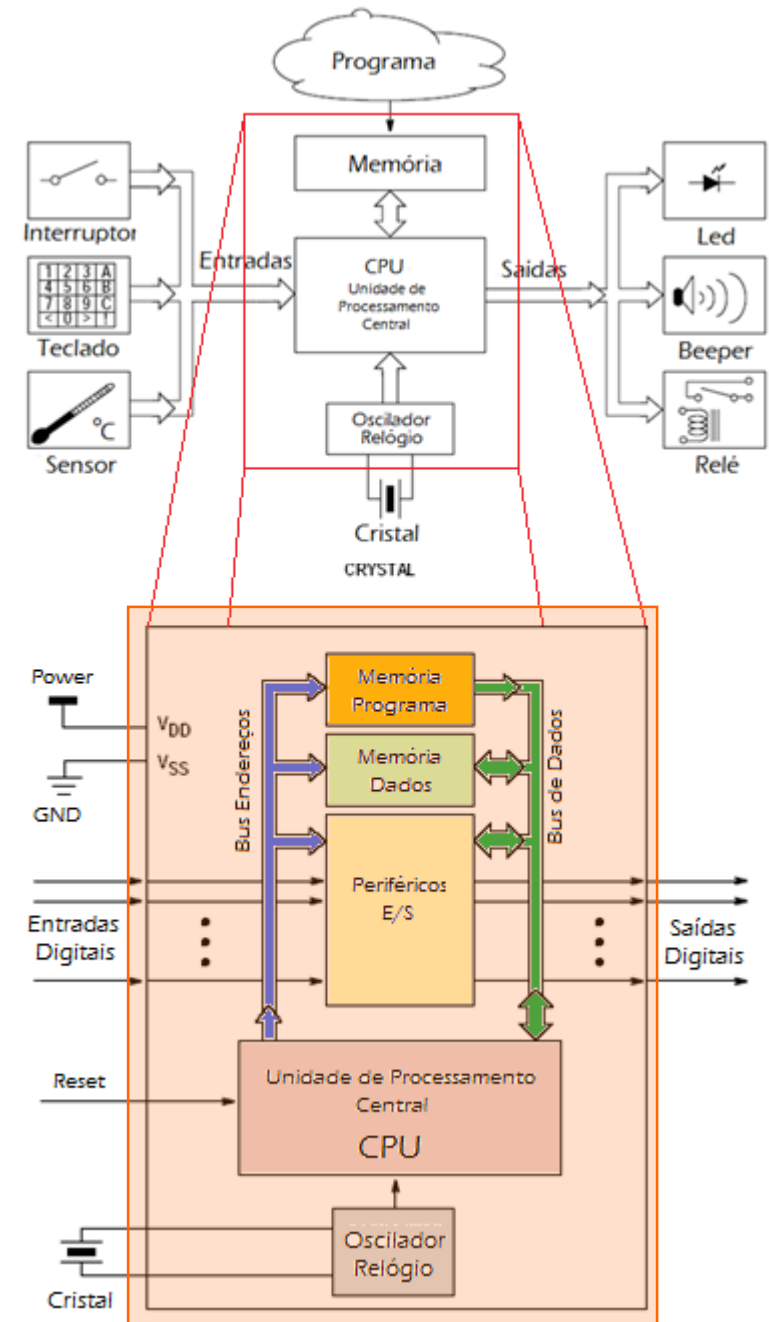


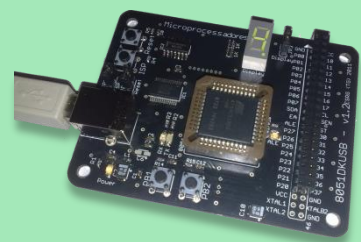
- **Componentes/Periféricos de um microcontrolador:**
 - CPU;
 - Diferentes tipos de memória: ROM (PROM, EPROM, FLASH), RAM, EEPROM;
 - Barramentos: endereços, dados e controlo;
 - Controlador de Comunicação: *ethernet*, CAN, I2C, SPI;
 - Unidades de temporização/contagem;
 - *Watchdog*;
 - Conversores A/D e D/A;
 - Portas de E/S digitais.



- O que é um microcontrolador ?

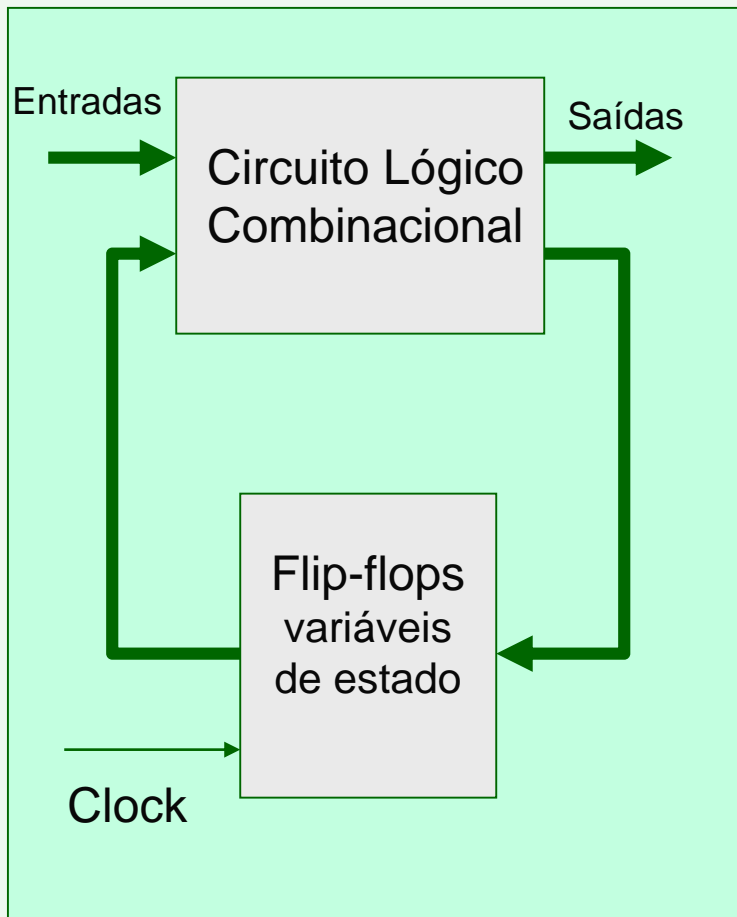
Microcontrolador



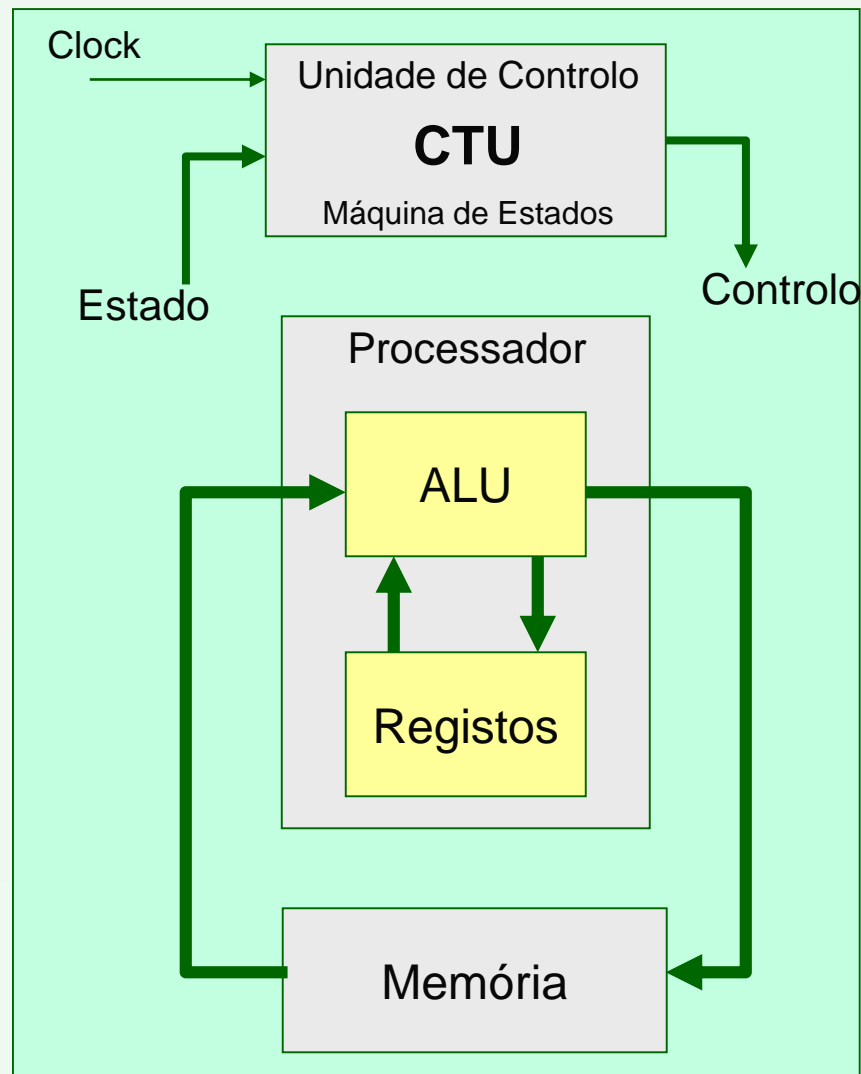


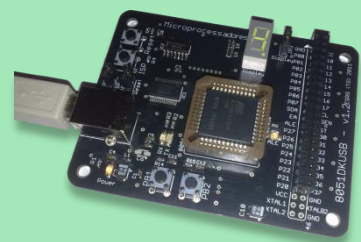
O que é um processador ?

Máquinas de estados

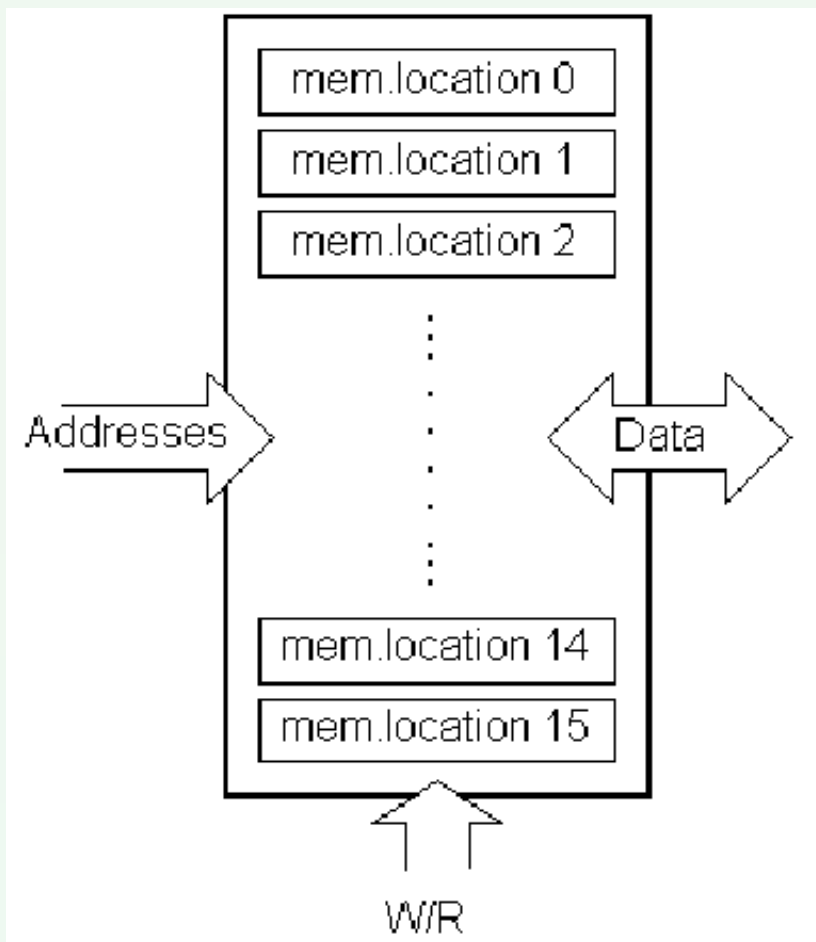


Modelo computacional elementar

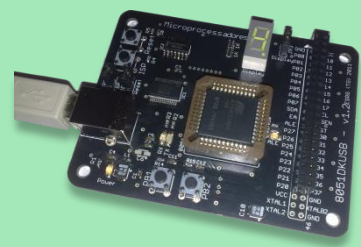




Unidade de Memória

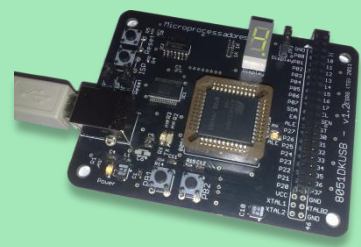


Exemplo de um modelo simplificado de uma unidade de memória.



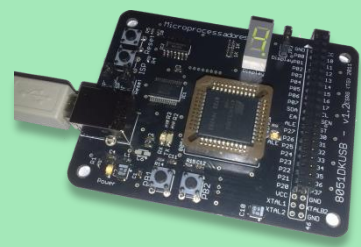
Revisão: O que é uma memória

- A memória de um computador pode ser vista como uma matriz de caixas de correio.
 - Cada caixa pode ser entendida como um invólucro onde guardamos as cartas (informação).
 - Num processador de oito *bits*, cada posição de memória pode ser vista como uma determinada caixa de correio que está dividida em oito prateleiras.
 - Ao contrário de uma caixa de correio real, onde podemos colocar sempre mais uma carta, neste caso só podemos guardar simultaneamente oito cartas.



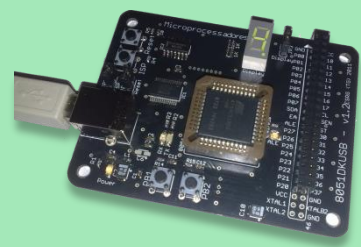
Revisões

- Em cada instante cada uma das prateleiras na caixa de correio pode estar vazia ou preenchida.
- Cada caixa de correio tem um único endereço, logo podemos guardar informação e retirá-la mais tarde.
- No caso de um processador com 10 linhas de endereço teríamos 1024 posições diferentes de memória.
- Quando o processador pretende ler uma posição de memória, este coloca uma combinação única de 1s e 0s nas linhas de endereço de forma a identificar a posição de memória pretendida.
 - Num processador definimos duas características o tamanho típico dos dados (8-bits, 16-bits, ...) e o tamanho das linhas de endereço que nos diz quantos *bytes* o processador consegue distinguir (endereçar).



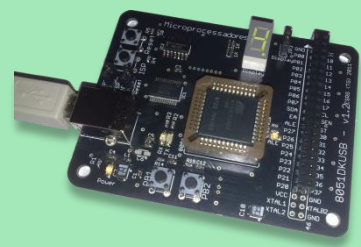
Revisões

- Um sistema computadorizado usa diferentes tipos de informação, requerendo diversos tipos de memória (quanto ao armazenamento dos dados):
 - Memória volátil:
 - Resultados temporários de variáveis são usados apenas naquele momento, não tendo que ser preservado quando o sistema é desligado. Nestes casos usa-se uma memória volátil para armazenar a informação.
 - Memória não-volátil:
 - Por exemplo, as instruções que controlam o processador não podem ser perdidas quando o sistema é desligado, logo terá que ser armazenada numa memória não-volátil.



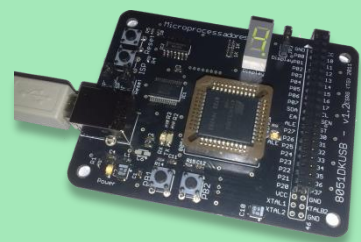
Revisão: Memórias

- Tipos de memórias (quanto a tecnologia):
 - *Random Access Memory* (RAM).
 - É um tipo de memória volátil cujas posições podem ser acedidas em qualquer ordem.
 - *Read-Only Memory* (ROM).
 - É um tipo de memória não-volátil. A informação armazenada é 'gravada' durante o fábriço do circuito integrado.
 - *Programmable ROM* (PROM).
 - É semelhante a ROM, mas pode ser programada uma vez após o circuito integrado ter sido feito.



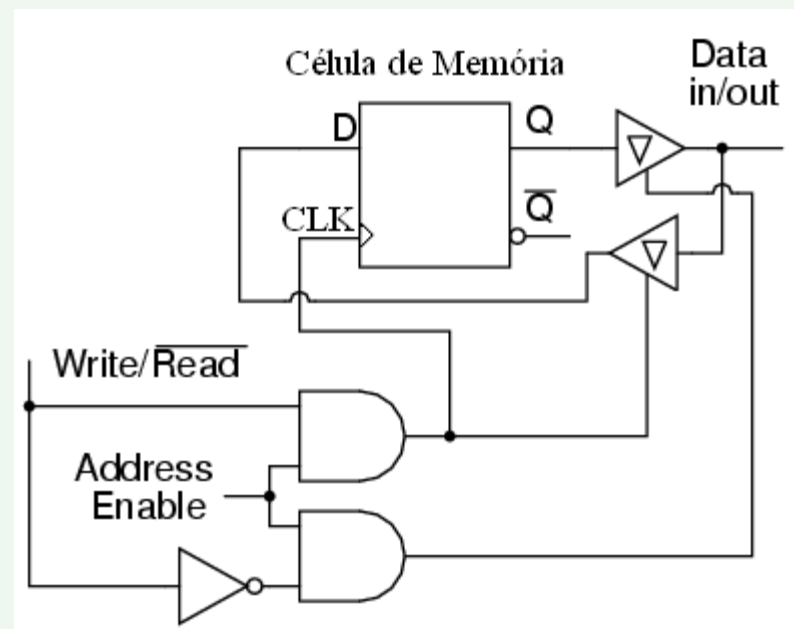
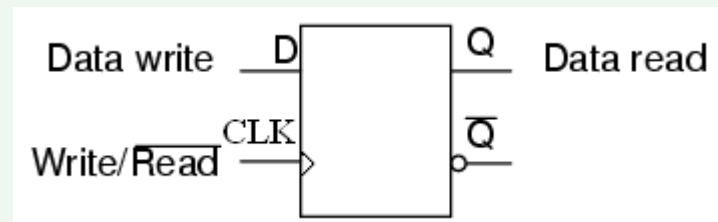
Revisão: Memórias

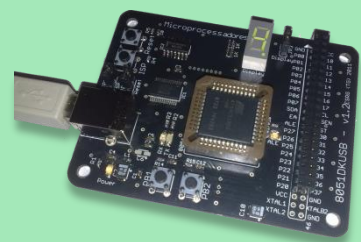
- *Erasable PROM* (EPROM).
 - Semelhante a ROM, mas a informação pode ser apagada e reprogramada tantas vezes quanto necessária.
 - O conteúdo é apagado através de luz ultravioleta.
- *Electrically erasable PROM* (EEPROM).
 - Semelhante a EPROM, mas o conteúdo pode ser apagado electricamente e gravado com o integrado inserido no circuito final.



Memória – Static RAM

- Numa *Static* RAM o *flip-flop* tipo-D pode ser utilizado para armazenar 1-bit de informação. A entrada D do *flip-flop* serve para gravar o bit no *flip-flop* e a saída Q do *flip-flop* para ler o bit armazenado.
- Utilizando dois *buffers* de 3 estados, podemos conectar a entrada “data write” e a saída “data read” a uma só linha de dados, fazendo com que os *buffers* ou conectem a saída Q à linha de dados (Read) ou conectem a entrada D à linha de dados (Write) ou se mantenham ambos num estado de alta impedância, o que significa que tanto a entrada D como a saída Q estão desconectadas da linha de dados.
- Notar que NANDs, NOTs e *buffers* são implementados com MOSFETs (tecnologia CMOS). A *Static* RAM já não é utilizada (espaço, consumo e dissipação de calor). Na *dynamic* RAM utilizam-se condensadores (carregado ou descarregado) para armazenar a informação binária, de modo prevenir as perdas é feito um *refresh* periódico aos condensadores. A tecnologia *flash* utiliza a gate isolada do MOSFET para implementar os condensadores que armazenam a informação.

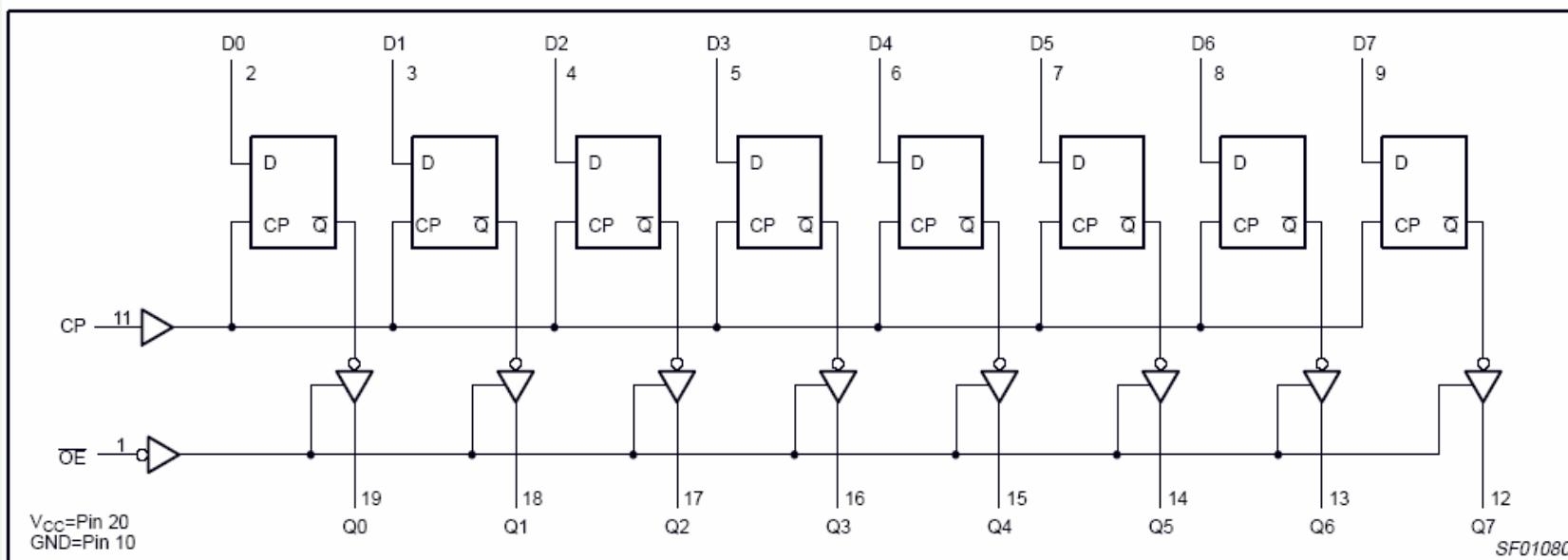




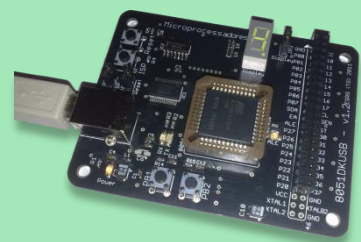
Registro de 8-bit

- Utilizando oito células de memória obtemos um registro de 8-bit. Notar que um registro pode ser visto como um único endereço de uma memória com 8-bit de dados. Relembrar que para implementar 1-byte de memória (ou um registro de 8-bit) poderemos utilizar um 74F574.

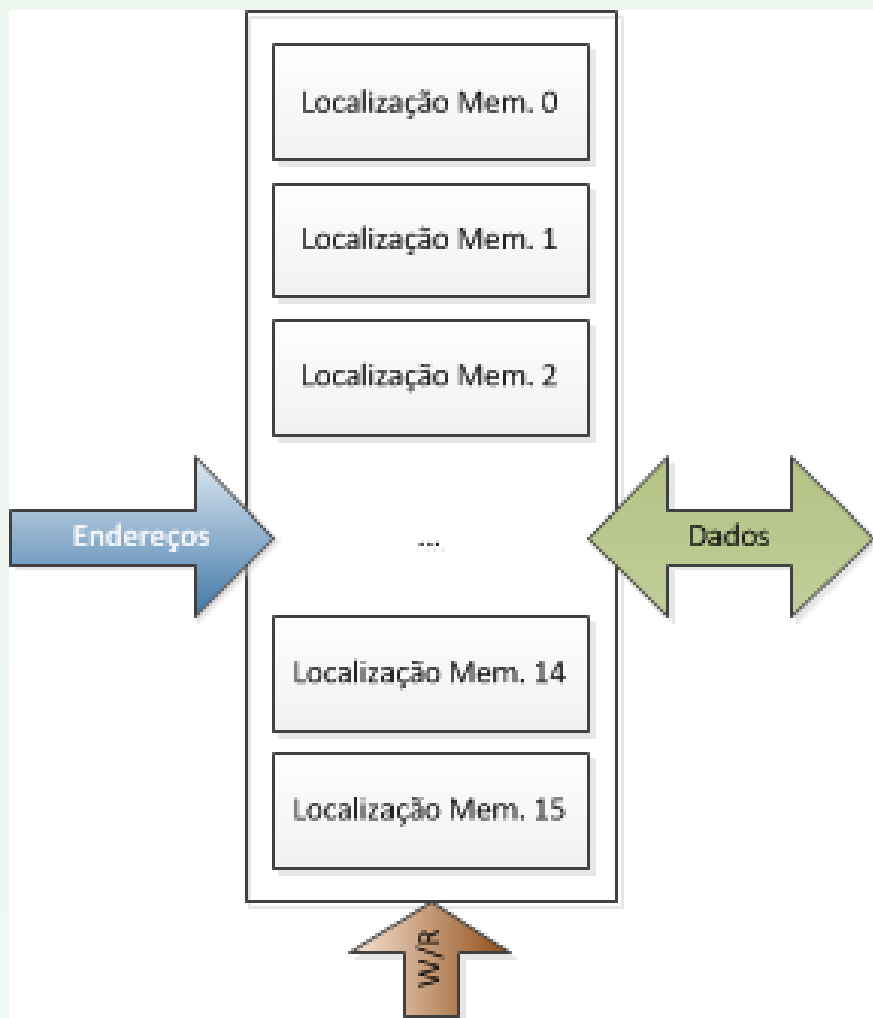
LOGIC DIAGRAM – 74F574 Philips Semiconductors



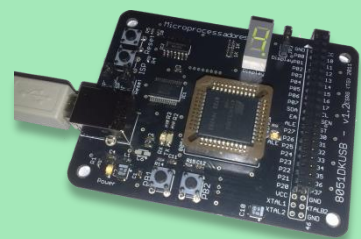
- A única diferença para a célula de memória anterior prende-se com a utilização da saída !Q que é utilizada para minimizar o nº de MOSFETs. Utilizando apenas este circuito integrado não é possível utilizar apenas um barramento de dados. O sinal CP é a entrada CLK (Write Data) e o sinal OE é usado como Read Data.



Unidade de Memória



Exemplo de um modelo simplificado de uma unidade de memória.

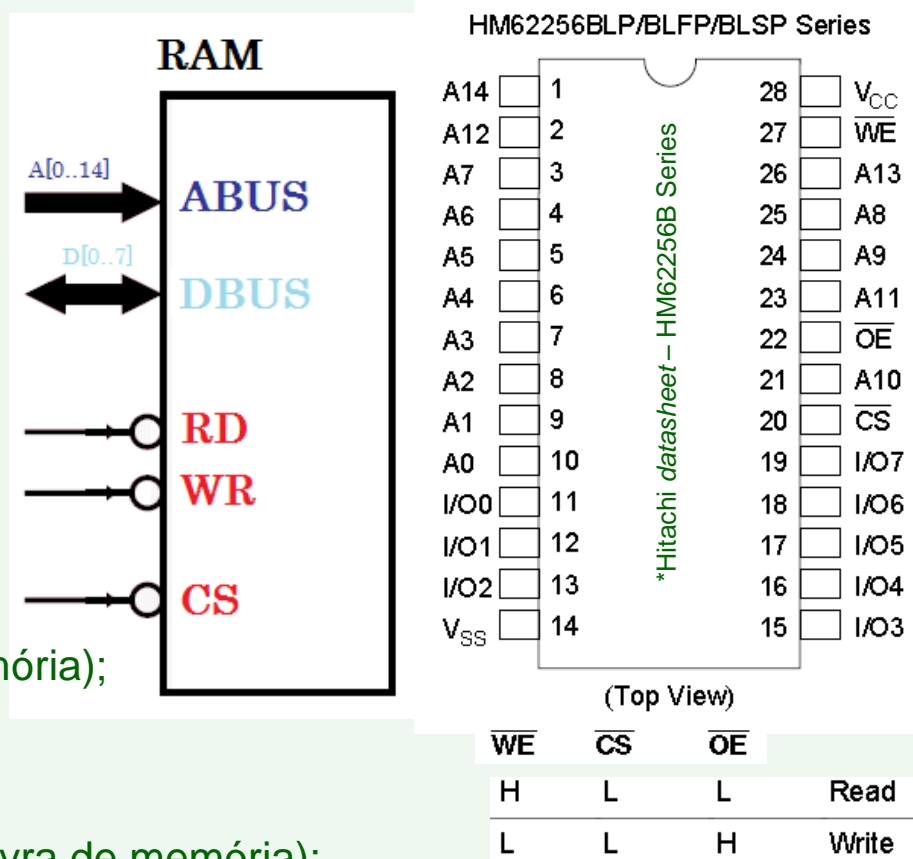


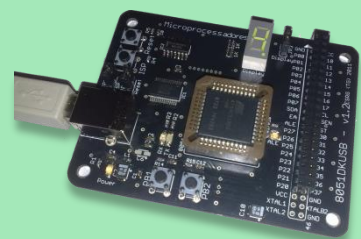
Leitura/Escrita Memória

- Barramentos:
 - Endereços (ABUS);
 - Dados (DBUS);
 - Controlo (RD, WR e CS).

Operação de Leitura

- Recebe:
 - Sinal de activação (CS);
 - ABUS (n linhas=tamanho da memória);
 - Sinal /RD para leitura.
- Devolve:
 - DBUS (n linhas=tamanho da palavra de memória);
 - Em DBUS é colocada a palavra que se encontra armazenada no endereço, fornecido por ABUS.





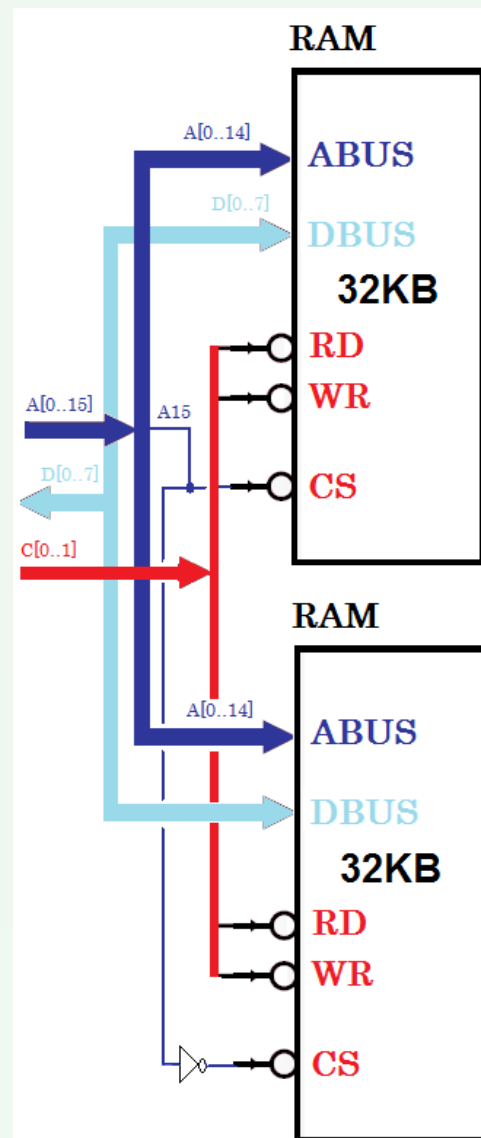
Leitura/Escrita Memória

Operação de Escrita

- Recebe:
 - Sinal de activação (CS);
 - ABUS (n linhas=tamanho da memória);
 - DBUS (n linhas=tamanho da palavra de memória);
 - Sinal /WR para escrita.
- Devolve:
 - Não devolve nada.
- Operação:
 - Armazena a palavra presente em DBUS no endereço presente em ABUS.

Operação de Leitura

- Recebe:
 - Sinal de activação (CS);
 - ABUS (n linhas=tamanho da memória);
 - Sinal RD para leitura.
- Devolve:
 - DBUS palavra armazenada na memória.
- Operação:
 - Coloca em DBUS a palavra armazenada no endereço presente em ABUS.



RAM - Exemplos

Memória 64Kb Não Volátil

- Tecnologia Ferroelétrica
- Funcionamento idêntico a RAM:
 - Sinal de activação (\overline{CE});
 - **ABUS** (n linhas=tamanho da memória);
 - **DBUS** (n linhas=tamanho da palavra de memória);
 - Sinal \overline{WE} para escrita;
 - Sinal \overline{OE} para leitura.
- Armazena dados durante 10 anos
- Operação escrita:
 - Colocar linha \overline{WE} a zero;
 - Colocar em ABUS endereço a escrever;
 - Colocar linha \overline{CE} a zero (a FRAM faz *latch* do endereço);
 - Colocar em DBUS o dado a escrever;
 - Coloca \overline{CE} a um;
 - Colocar \overline{WE} a um.

FM1608

64Kb Byte-wide FRAM Memory

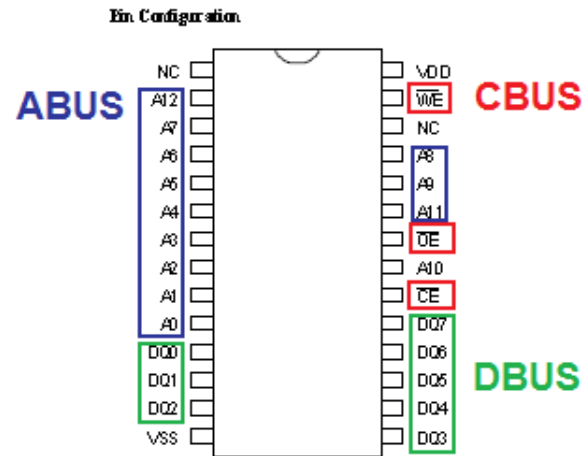
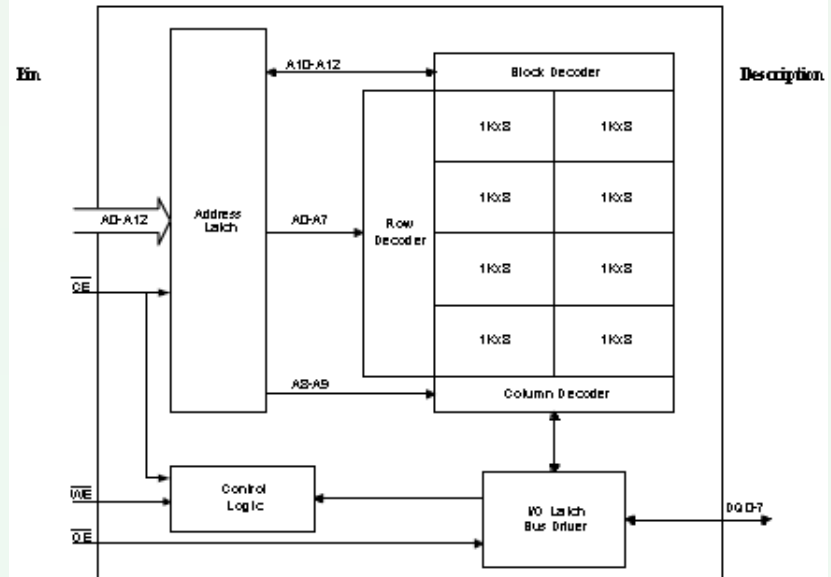
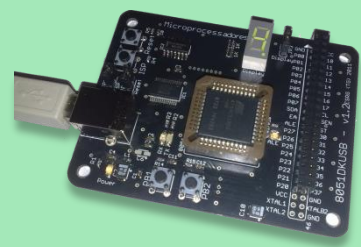


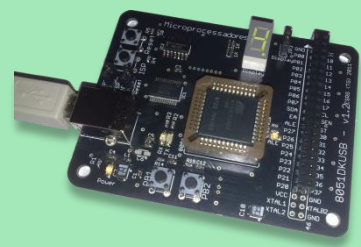
Figure 1. Block Diagram





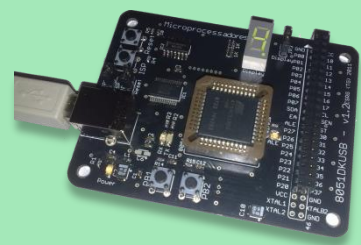
Exercício

- Pretende-se utilizar vários *chips* FM1608 para implementar um espaço de memória de 64KB.
 - Quantos CI FM1608 necessita?
 - Faça o mapeamento para cada memória;
 - Obtenha as equações lógicas dos sinais de /CE de cada memória;
 - Implemente o circuito lógico que gera os sinais.

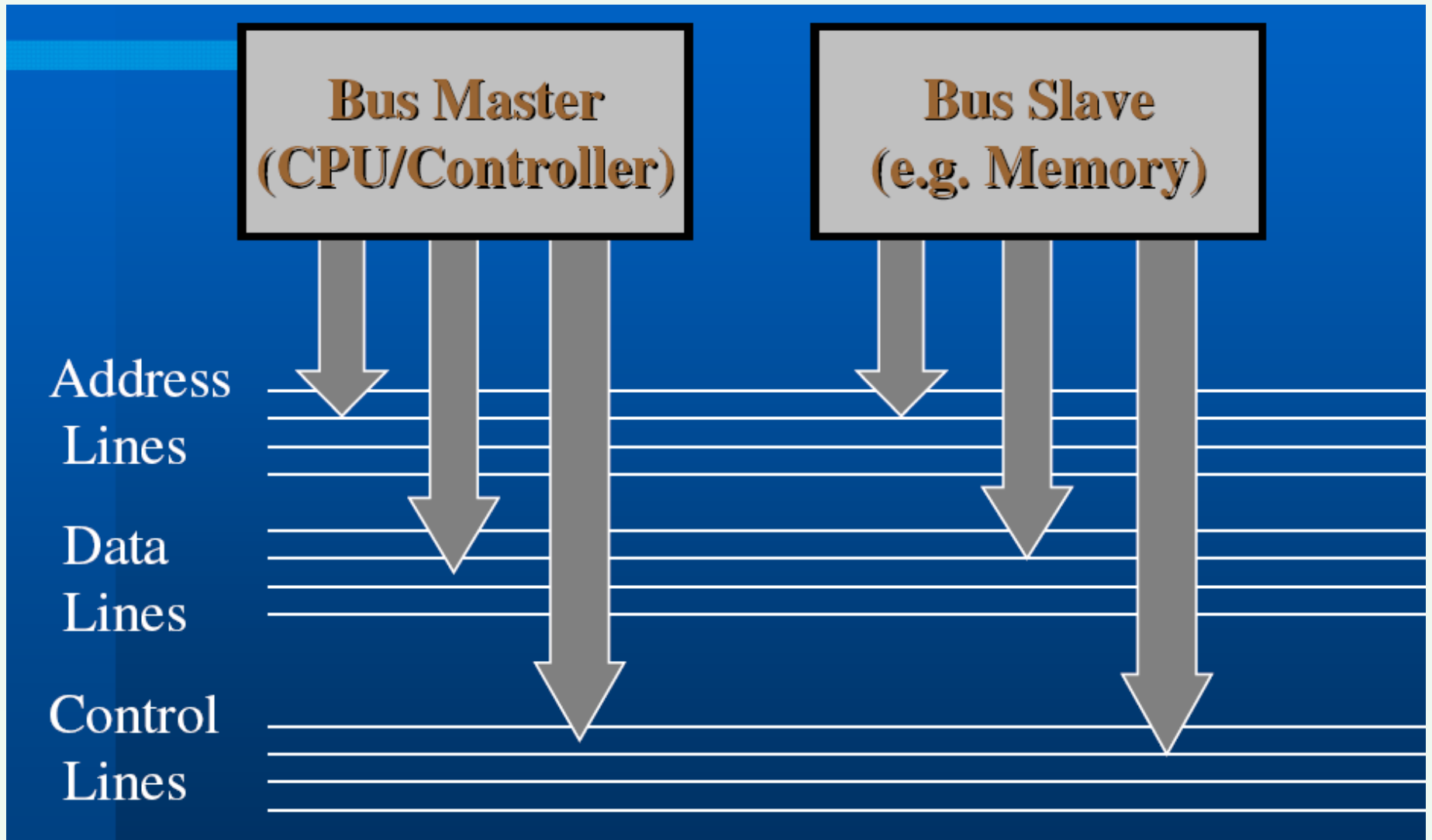


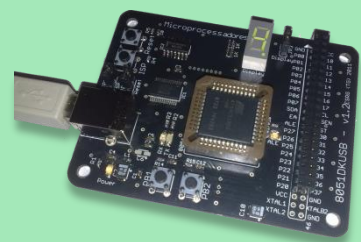
Barramentos

- O processador de um computador comunica com outros módulos do computador através de um dispositivo chamado barramento (bus).
- Existem várias arquiteturas de barramento disponíveis no mercado, tais como PCI; cPCI; VME ...
- Todas as arquiteturas incluem um barramento de controlo, um barramento de dados e um barramento de endereços.
- Similarmente os microcontroladores utilizam também uma arquitetura de barramento para comunicarem com todos os módulos existentes dentro do chip. A arquitetura pode variar de micro para micro, mas incluem os três barramentos referidos.



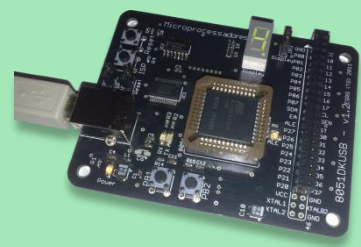
Barramentos





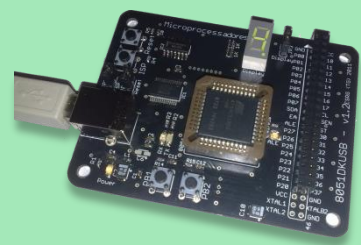
Sequência de Escrita (Write)

- O *master* coloca o endereço da memória, onde os dados devem ser escritos, no barramento de endereços e qualifica-os usando as linhas de controlo;
- O *master* sinaliza, usando as linhas de controlo, que esta é uma operação de escrita;
- O *master* coloca os dados a serem escritos no barramento de dados e qualifica-os utilizando as linhas de controlo;
- Após a qualificação do endereço pelo *master*, o *slave* compara o endereço no barramento de endereços com o seu próprio endereço. Se a operação de **Write** lhe é destinada, ele adquire os dados e sinaliza usando as linhas de controlo que terminou a operação (faz o *acknowledge* dos dados).



Sequência de leitura (Read)

- O *master* coloca o endereço de memória de onde pretende ler os dados no bus e qualifica-o usando as linhas de controlo;
- O *master* sinaliza, usando as linhas de controlo, que se trata de uma sequência de leitura;
- O *master* sinaliza que está pronto a receber os dados usando as linhas de controlo;
- O *slave* compara o endereço no barramento com o seu próprio endereço, após o *master* o ter qualificado. Se a leitura se refere a ele, ele coloca os dados no barramento e sinaliza, usando as linhas de controlo que terminou (*acknowledge* dos dados). No final o *master* realiza o *Latch* dos dados.



Transferência Paralelo

- Num barramento paralelo todos os bits de um byte ou word (palavra: 16-bit ou 32-bit) são transferidos simultaneamente;
- Se tivermos um barramento com um byte de tamanho (8 linhas) e uma frequência de 1MHz então temos uma velocidade de barramento de 1Mbytes/seg;
- Num barramento há transferência de dados em paralelo (oposta à transferência de dados em série que iremos analisar posteriormente).