

1. Sistemas binários:

- ✓ Circuitos combinatórios;
- ✓ Circuitos sequenciais;
- ✓ Representação de números;
- ✓ Notação em complemento para 2;
- ✓ Soma e subtração;
- ✓ Grandes números.

2. O Mundo Binário

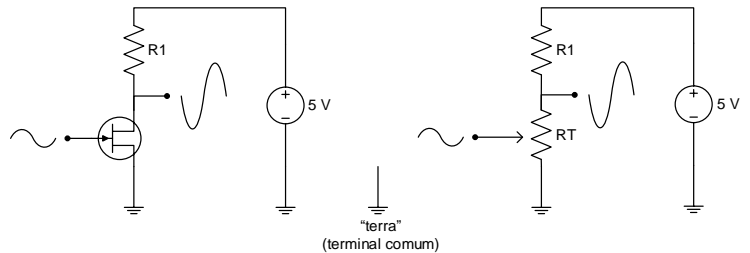
Circuitos eletrônicos analógicos – Amplificador analógico

Ondas sinusoides → Sinais **analógicos**

1Hz corresponde a um ciclo completo (até voltar o padrão) por cada segunda.

As **variações** dos sinais de saída:

- São de maior amplitude do que os sinais de entrada (devido à amplificação);
- Estão invertidas em relação ao sinal de entrada, uma vez que, quando o sinal de entrada sobe, RT (e a tensão de saída desce).

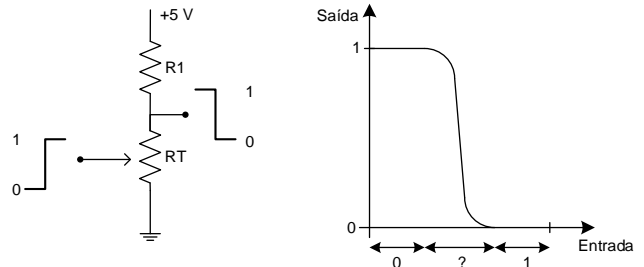


Circuitos eletrônicos digitais

Amplificador saturado

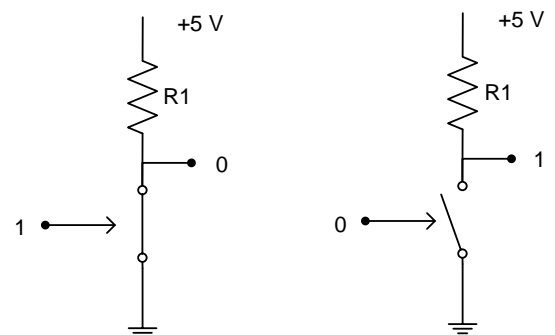
A zona de transição 0 → 1 (amplificador saturado) – é conhecida como **margem de ruído**:

- ↳ Estão sujeitos ao ruído;
- ↳ Requerem/gastam energia.




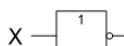




Inversor (negação)

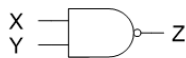





O **Inversor** é encarado como um sistema digital (ou binário) puro, com apenas dois estados possíveis (0 e 1) e em que se assume que o transístor se comporta como um interrupto comandado pelo sinal de entrada, uma vez que a sua resistência varia entre um valor nulo (entrada com valor 1, interruptor fechado, saída com valor zero) ou infinito (entrada com valor 0, interruptor aberto, saída com valor 1).



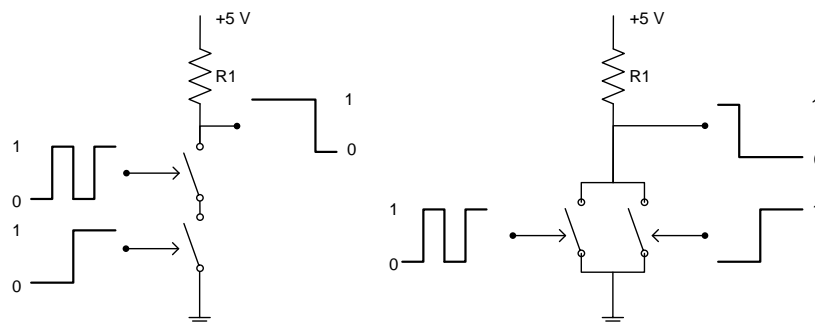
Portas lógicas (gates) básicas

- A “bolinha” tem a simbologia de negação.
- AND – as duas têm que ser verdade, para a terceira ser falsa.
- OR – as duas têm que ser falsas, para a terceira ser verdadeira.
- NAND – AND seguido de um NOT – só quando as duas são verdade, a terceira é falsa.
- NOR – as duas têm que ser falsas, para a terceira ser verdadeira.
- XOR – a saída só é verdade quando apenas uma é verdade.

PORTA	SÍMBOLOS		FUNÇÃO	TABELA DE VERDADE															
NOT			$Z = \bar{X}$	<table><tr><th>X</th><th>Z</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	X	Z	0	1	1	0									
X	Z																		
0	1																		
1	0																		
AND			$Z = X \cdot Y$	<table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X	Y	Z	0	0	0	0	1	0	1	0	0	1	1	1
X	Y	Z																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OR			$Z = X + Y$	<table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X	Y	Z	0	0	0	0	1	1	1	0	1	1	1	1
X	Y	Z																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	

PORTA	SÍMBOLOS	FUNÇÃO	TABELA DE VERDADE			
NAND			$Z = \overline{X \cdot Y}$	X	Y	Z
				0	0	1
				0	1	1
				1	0	1
				1	1	0
NOR			$Z = \overline{X + Y}$	X	Y	Z
				0	0	1
				0	1	0
				1	0	0
				1	1	0
XOR			$Z = X \oplus Y$	X	Y	Z
				0	0	0
				0	1	1
				1	0	1
				1	1	0

NAND e NOR



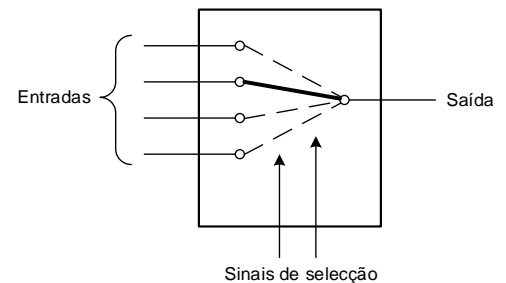
Essencial

- Os circuitos eletrónicos digitais são circuitos analógicos que usam apenas duas games de valores de tensão, situadas nos extremos da gama total de valores possíveis. Qualquer valor na gama baixo é um 0, e qualquer valor na gama alta (perto do valor de tensão de alimentação) é 1;
- A transição de 0 para 1 e vice-versa gasta tempo e energia. Isto limita a frequência máxima de operações de um circuito digital e por conseguinte de um computador;
- Uma porta lógica é um circuito destes cujo valor de saída depende de uma ou mais entradas. A porta NOT troca a entrada (se 1 dá 0 e vice-versa). A saída de uma porta AND só é um se ambas as entradas forem 1. A porta OR tem 1 na saída desde que pelo menos uma das entradas tenha 1;
- Uma tabela de verdade é uma tabela que indica o valor da(s) saídas(s) para cada combinação dos valores de entradas. Um diagrama temporal permite ver as variações dos sinais de saída ao longo do tempo em função das variações dos sinais de entrada.

Multiplexer

Um **multiplexer** permite escolher entre uma de várias entradas e transportar o seu valor para uma saída, sob controlo de um ou mais sinais de seleção.

O número de entradas é normalmente uma potência 2, por n sinais de seleção permitem seleccionar 2^n hipóteses diferentes, normalmente, numeradas entre 0 e 2^n-1 .



Número sinais de selecção	Número de combinações possíveis		Game de numeração das combinações		
			Mínimo	Máximo	
4	2^4	16	0	15	2^4-1

Notas:

- Um sinal escolhe duas opções – 0 ou 1.
- Um bit é um sinal de 0 a 1.
- 1 byte são 8 bits.

O bit de menor peso / o que varia mais rapidamente é o da direita.

Um dígito hexadecimal corresponde a 4 bits; A correspondência entre binário e hexadecimal é direta.

Base (decimal) 5: 1,2,3,4,5,10,11,12,13,14,15,20,...

0001 – necessita de 1 bit

1111 – necessita de 4 bits

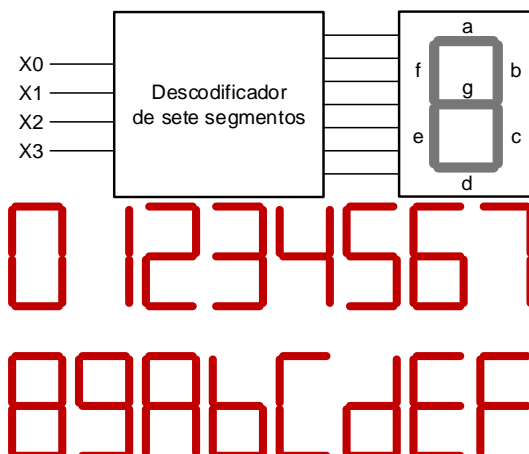
7AF5H – base hexadecimal

Decimal	Binário	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Descodificadores

Um **descodificador** é um circuito combinatório que, para cada uma das 2^n combinações, das n variáveis de entrada, produz uma combinação das p variáveis de saída com a restrição $n < p$. Os descodificadores usam-se quando apenas um subconjunto das 2^p combinações são interessantes, pelo que em vez de usar p variáveis para o fazer, se codificam as combinações interessantes usando menos variáveis (*bits*) para o fazer.

Um exemplo de descodificador é o que produz os sinais necessários a um mostrador (*display*) de sete segmentos a partir de 4 bits.



Essencial

- Um circuito combinatório é aquele em que nenhuma entrada de uma porta lógica dependa, diretamente ou indiretamente da sua saída. Assim, as saídas dos circuitos dependem apenas das suas entradas, sem realimentações.
- Os multiplexers permitem escolher uma de n entradas para reproduzir na sua saída, sob controlo de k sinais de seleção em que $n=2^k$. Os multiplexers mais frequentes têm 2,4,8 a 16 entradas, com 1,2,3,4 sinais de seleção.
- Os decodificadores têm n saídas, todas a 1 exceto uma, a indicada por k sinais de seleção em que $n=2^k$.

Circuitos sequenciais

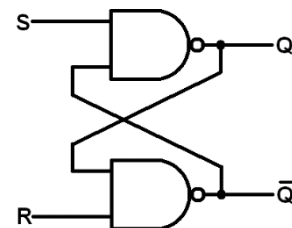
Ao contrário dos circuitos **combinatórios**, em que as saídas dependem apenas dos valores das entradas, nos circuitos **sequenciais** o valor das saídas pode depender não apenas das entradas, mas também dos valores anteriores das saídas.

Uma combinação dos valores das saídas do circuito designa-se **estado**. Quando as entradas do circuito mudam o seu valor, os valores das saídas podem mudar em consequência, dizendo-se nessa altura em que houve uma **transição** de estado.

Elementos biestáveis são elementos com apenas uma saída (um bit, dois estados possíveis) e com capacidade de memória, isto é, manter a saída mesmo que a entrada mude.

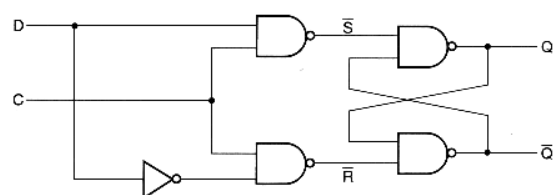
Trinco SR

Um **trinco** (*latch*) de uma porta tem a propriedade de permitir à porta fechar (encolhendo a lingueta) quando se empurra mas depois prender e já não deixar abrir quando se tenta fazer o movimento inverso (puxar). É preciso atuar noutro local para permitir abrir a porta novamente.



Trinco D

O trinco **SR** permite memorizar 0 ou 1 na saída Q , mas precisa de dois sinais separados para poder mudar de estado e não tem controlo sobre mudar ou não a saída quando as entradas mudam. O trinco **D** permite resolver estes dois problemas.



(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

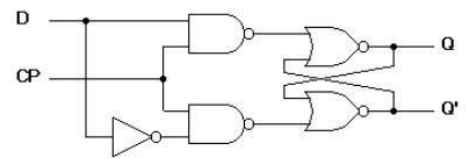
(b) Function table

Quando a saída Q é igual à entrada D diz-se que o trinco está **transparente**.

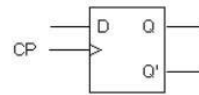
Báscula (*flip-flop*) D (ativa no flanco ascendente)

Uma **báscula** é um circuito constituído por dois trincos em que quando um está transparente outro está a memorizar, de forma a que nunca se verifique uma situação em que a saída dependa exclusivamente da entrada, sendo os instantes possíveis de transição da saída comandadas por um sinal de controlo.

A báscula D só pode mudar a saída quando o sinal C transita de valor. Tal como no trinco D, a saída Q memoriza o valor que a entrada D tem, mas a diferença é que a báscula tira uma espécie de fotografia à entrada D na altura da transição de C, e mantém esse valor mesmo que depois a entrada D varie, ao passo que o trinco acompanha as variações da entrada D enquanto C estiver ativo (e não apenas na transição).



(a) Logic diagram with NAND gates

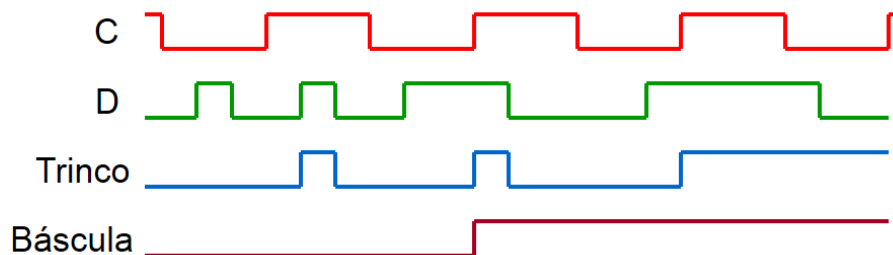


(b) Graphical symbol

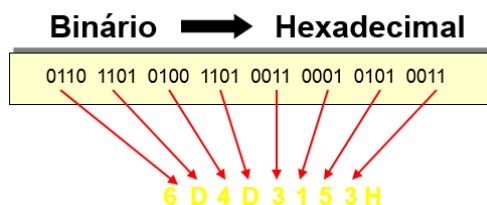
Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

(c) Transition table

Clocked D flip-flop



Representação de números



Soma (binário e hexadecimal)

0 1 0 0 1 1 1 0	transporte
0 1 1 0 1 0 1 1	operando A
+ 0 1 0 0 0 1 1 0	operando B
1 0 1 1 0 0 0 1	resultado
0 1	transporte
6 B H	operando A
+ 4 6 H	operando B
B 1 H	resultado



Gama de números

Com N bits consegue-se representar números inteiros

0 a 2^N-1 (só > 0) ou -2^N-1 a $+(2^N-1-1)$

Exemplo: 8 bits:


0 a 255 (só > 0) ou -128 a +127

Sem sinal (só > 0)		Com sinal	
1111 1111	255	0111 1111	+127
1111 1110	254	0111 1110	+126
...
1000 0010	130	0000 0010	2
1000 0001	129	0000 0001	1
1000 0000	128	0000 0000	0
0111 1111	127	1111 1111	-1
0111 1110	126	1111 1110	-2
...
0000 0001	1	1000 0001	-127
0000 0000	0	1000 0000	-128

Complemento para dois

→ Começa-se da direita a copiar os 0s. Depois copia-se o primeiro 1. Seguidamente trocam-se os 0 para 1 e vice-versa.

→ Note-se que se o bit mais à esquerda for 0 o número é positivo, caso contrário é 1.

0 1 0 1 1 1 0 0 número (5CH)

 1 0 1 0 0 1 0 0 complemento para 2 (A4H = -5CH)

Na **extensão de sinal**, se:

- O número for positivo, acrescentam-se mais zeros à sua esquerda;
- O número for negativo, acrescentam-se mais uns à sua esquerda.

O Excesso ou **overflow** acontece quando ao somar dois números o número de bits não é suficiente para representar o novo valor.

0 1 0 1 1 1 1 transporte
 0 1 0 1 1 1 0 1 operando A
 + 0 1 0 1 0 1 1 1 operando B

 1 0 1 1 0 1 0 0 soma

Oops! Resultado negativo!!!

Potências de 2

Símbolo	Lê-se	Equivalente	Valor binário	Valor decimal	Valor decimal aproximado
K	Kilo	1024	2^{10}	1 024	10^3
M	Mega	1024 K	2^{20}	1 048 576	10^6
G	Giga	1024 M	2^{30}	1 073 741 824	10^9
T	Tera	1024 G	2^{40}	1 099 511 627 776	10^{12}

N	2^N (decimal)	K (1024)	2^N (hexadecimal)
0	1		1
1	2		2
2	4		4
3	8		8
4	16		10H
5	32		20H
6	64		40H
7	128		80H
8	256		100H
9	512		200H
10	1024	1 K	400H
11	2048	2 K	800H
12	4096	4 K	1000H
13	8192	8 K	2000H
14	16384	16K	4000H
15	32768	32K	8000H
16	65536	64K	10000H

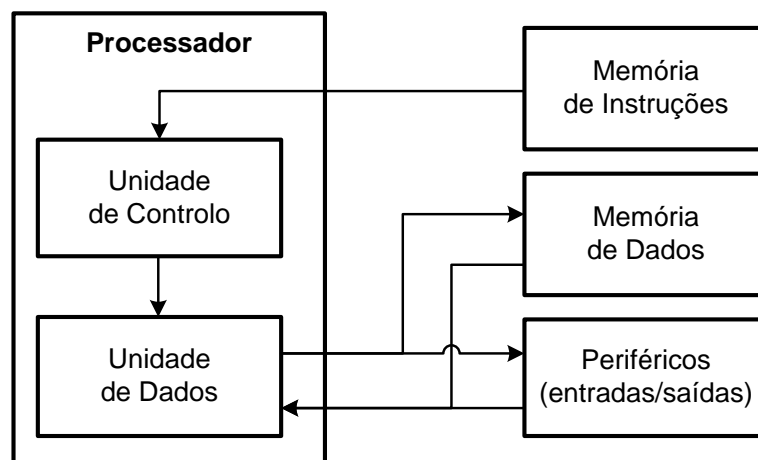
Cálculo de potências de 2

Potência 2	Decomposição	Ou seja...	Resultado
2^{20}	$2^{16} * 2^4$	64K * 16	1M
2^{20}	$2^{10} * 2^{10}$	1K * 1K	1M
2^{12}	$2^{10} * 2^2$	1K * 4	4K
2^{14}	$2^{16} / 2^2$	64K / 4	16K
2^{27}	$2^{20} * 2^7$	1M * 128	128M
2^{30}	$2^{20} * 2^{10}$	1M * 1K	1G

3. O meu primeiro computador

Componentes básicos de um computador:

- Processador – coordena tudo e executa as operações definidas pelo programa e inclui duas unidades fundamentais:
 - Unidade de dados – faz o processamento, executando as operações (soma, subtracções, etc.) sobre os dados;
 - Unidade de controlo – interpreta as instruções do programa e controla tudo (não apenas a unidade de dados, mas também os acessos às memórias e aos periféricos).
- Memória:
 - De instruções – armazena as instruções do programa a executar;
 - De dados – armazena não apenas os dados de entrada do programa, mas também os resultados de saída e mesmo os resultados parciais, intermédios;
- Periféricos – permitem efectuar a entrada e saída de informação no computador.



A diferença entre ROM (Read Only Memory) e RAM (Random Access Memory) é que enquanto a primeira só pode ser escrita uma vez e o seu conteúdo nunca se perde quando desligado, a segunda pode ser escrita quantas vezes forem necessárias e é volátil.

Tanto as ROMs e as RAMs são constituídas por um conjunto linear de células de memória, seleccionadas por um endereço.

Nota: O nome WR, significa "Write", se for 0 está ativo e escreve, se for 1, está inactivo e lê.

(fig 3.2 pag 116)

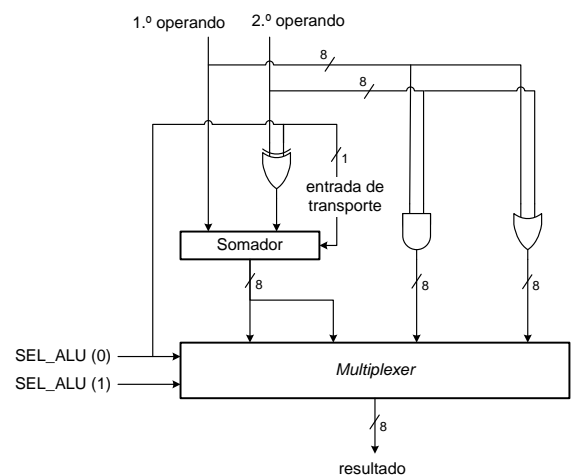
Conceitos básicos numa RAM:

- Capacidade em células – é sempre uma potência de base 2.
- Largura de cada célula, em bits – quantos bits consegue cada célula armazenar de cada vez. Tipicamente é uma potência de 2 múltipla de 8, ou seja, um número inteiro de bytes (valores típicos: 8,16,23,64 e 128).
- Capacidade em bytes – produto da capacidade em células pela largura da célula em bytes. Se uma RAM tiver 16 células e cada célula tiver 16 bits (2 bytes), a capacidade da RAM será de 32 bytes).
- Endereço – Número da célula a que se pretende aceder, começando em 0 e terminando em N-1, em que N é a capacidade (em célula) da RAM. O número de bits necessários para o endereço depende de N. São necessários 4 bits para poder representar 16 endereços diferentes. Se a capacidade fosse 256 células, seriam necessários 8 bits. No caso geral, o endereço necessita de um número de bits igual ao logaritmo de base 2 (em células) da RAM.

Em "Assembly" sempre que tiver [] mexe com a memória de dados.

A existência de um registo dentro da unidade de dados do processador é fundamental. Sem ele, as operações com dois operandos só seriam possíveis se as memórias permitissem ler duas células e escrever outra ao mesmo tempo.

A **ALU** (Unidade Aritmética e Lógica) permite implementar diversas operações com dois operandos.



Para além das operações da ALU, a unidade de dados tem de saber transferir dados entre o registo e a memória de dados.

A largura em bits da unidade de dados (e todos os seus recursos internos, incluindo o registo) deve ser igual à memória e define a gama de números que o processador consegue lidar (com n bits, o processador consegue lidar com números em complemento para 2 entre -2^{n-1} e $+2^{n-1}-1$).

Os processadores com uma largura de 8 e 16 bits usam-se para aplicações específicas de baixo custo e sem necessidade de grande capacidade de cálculo. Os processadores de 32 e 64 bits usam-se em computadores de utilização genérica de médio e alto desempenho.

A RTL (Register Transfer Language) é uma notação simbólica que permite descrever de forma compacta as operações básicas que envolvem transferência de dados entre elementos de memorização (células de memória e registo), bem como operações sobre esses dados.

- A unidade de controlo inclui um registo (**PC**- Program Counter) que contém o endereço (na memória de instruções) da instrução a ser executada. Cada instrução inclui directamente os sinais de controlo e uma constante (seja um dado, seja um endereço), pelo que é semelhante a uma microinstrução numa unidade de controlo microprogramada.
- O PC é um contador com carregamento paralelo. Normalmente, conta sequencialmente, avançando em cada ciclo de relógio. No entanto, se a instrução assim o especificar, pode saltar para um endereço diferente, carregando a constante da instrução no PC. Os saltos podem ser condicionais (dependem de uma condição) ou incondicionais (saltam sempre). As condições normalmente são muito simples ($A=0$ ou $A<0$).
- Em cada ciclo de relógio é executada uma nova instrução. Quando o relógio inicia um novo ciclo, o PC muda e as acções preparadas durante a instrução anterior (escritas no registo A, por exemplo) ocorrem nessa altura. Assumem-se registos que mudam o estado num dos flancos do relógio.
- A definição da largura de um processador é fundamental. Não só defini a gama de números inteiros como pode trabalhar directamente como tem implicações no número de bits dos endereços de memórias.

- A memória de dados deve ter a mesma largura que o processador. Embora a capacidade da memória de dados possa ser arbitrariamente grande, acaba por ter de ser limitada pelo facto de os endereços desta memória terem de circular pelo mesmo caminho que os dados. Logo, tipicamente o número de bits de endereço da memória de dados não é superior à largura do processador.
- A largura da memória de instruções pode ser diferente da largura do processador.
- A unidade de controlo tem a seu cargo a gestão dos sinais de controlo, gestão do pc e da sua actualização, através dos saltos condicionais e incondicionais, e a interacção com a memória de instruções.
- A unidade de dados lida sobretudo com o processamento dos dados, nomeadamente através da AL e do registo A, e com a interacção com a memória de dados.
- Cada instrução no PEPE-8 demora apenas um ciclo de relógio a ser executada. Durante a execução de uma instrução, os sinais necessários para a operação em causa (escrita no registo A, por exemplo) são preparados. Quando o relógio inicia um novo ciclo, o PC muda as acções preparadas durante a instrução anterior ocorrem nessa altura. Assumem-se registos disparados pelos flancos do relógio (edge- triggered).
- A linguagem assembly é específica de cada processador e permite codificar uma longa lista de valores de sinais num só número, o opcode (código de operação). É uma notação mais compacta, mais simples e reduz os erros de programação.
- Ao executar um programa, o processador vai lendo cada instrução e o respectivo opcode tem de ser decodificado e convertido na combinação de sinais de controlo que lhe corresponde, o que é conseguido com uma ROM de decodificação.
- Os valores que possam variar para uma mesma instrução (valor constante, por exemplo), não podem ser incluídos no opcode e têm de constar, ao lado deste, em cada instrução.
- O assembler converte o programa assembly (texto) em números binários, instruções que o processador consegue executar directamente.