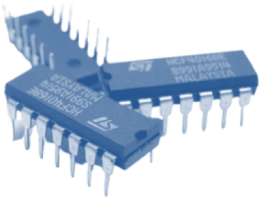
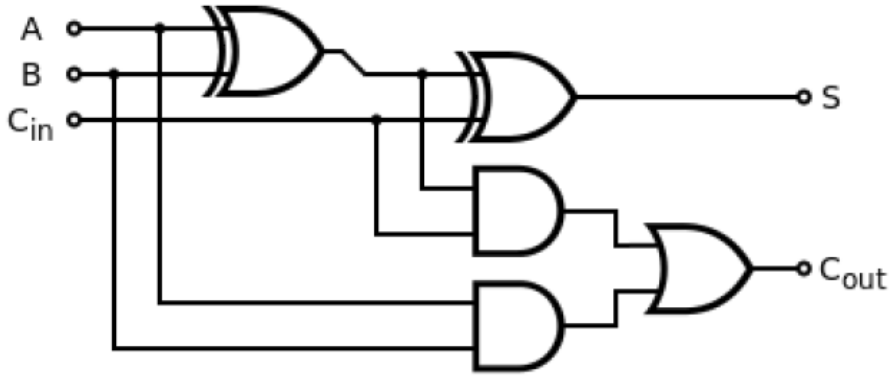
TÉCNICO LISBOA

TÉCNICO LISBOA

Sistemas Digitais



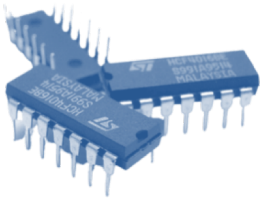


Circuitos Combinatórios

Prof. António Grilo

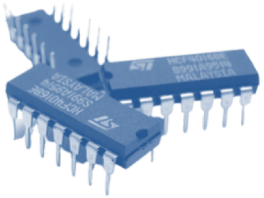
TÉCNICO LISBOA

Sumário



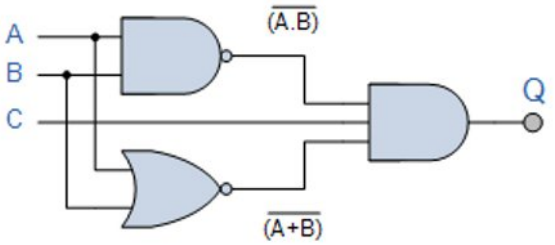
- Noção de Circuito Combinatório
  - Definição
- Tempo de Propagação
  - Tempos de propagação numa porta lógica
  - Diagrama temporal
  - Noção de caminho crítico
- Circuitos Combinatórios Simples
  - Codificadores
  - Descodificadores
  - Multiplexers
  - Demultiplexers
- Circuitos Aritméticos
  - Somadores elementares
    - Circuito Ripple-Carry
    - Representação de números com sinal
    - Operações com números com sinal
  - Subtractores
  - Comparadores
  - Unidade Lógica e Aritmética
  - Bits de estado/Flags

TÉCNICO LISBOA



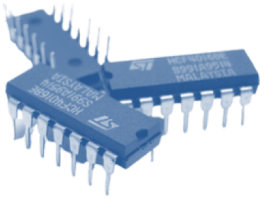
Noção de Circuito Combinatório

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

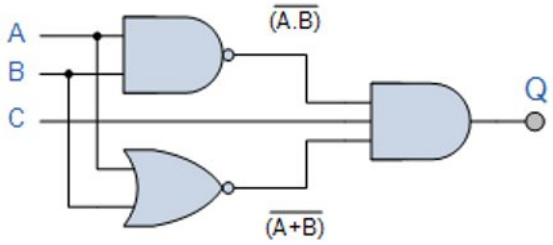


C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Noção de Circuito Combinatório

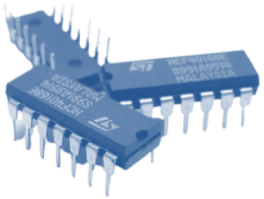


- Circuito Combinatório:
  - A saída é uma função que depende apenas da entrada atual;
  - Definido através de:
    - **Função Booleana** – Ex: Q= $(\overline{A.B}).(\overline{A+B}).C$
    - **Diagrama lógico**
    - **Tabela de verdade**

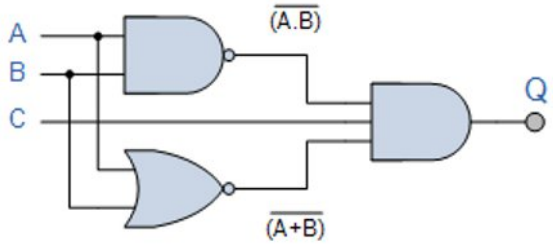


C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Noção de Circuito Combinatório



- Circuito Combinatório:
  - A saída é uma função que depende apenas da entrada atual;

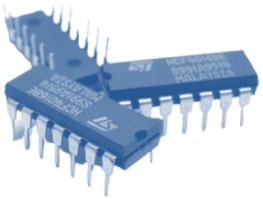


  - Definido em contraste com a noção de **circuito sequencial**, em que a saída depende não só da entrada atual, mas também do valores anteriores dessa entrada…

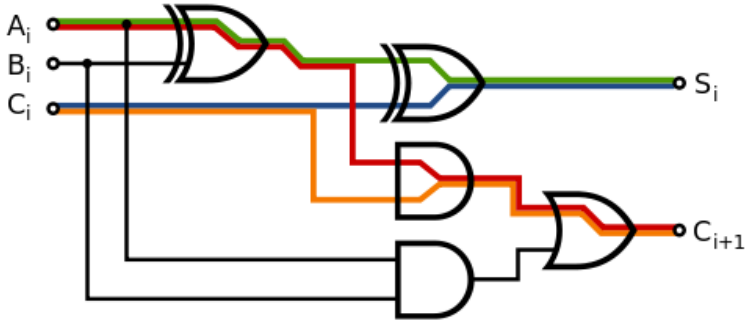
    i.e., circuitos sequenciais têm "efeito de memória", enquanto que um circuito combinatório não.

    Veremos em breve…

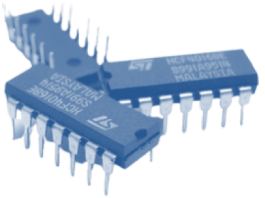
TÉCNICO LISBOA



Tempo de Propagação

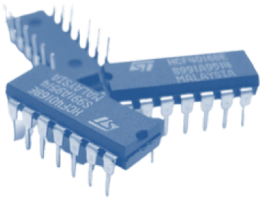


Tempos de Propagação

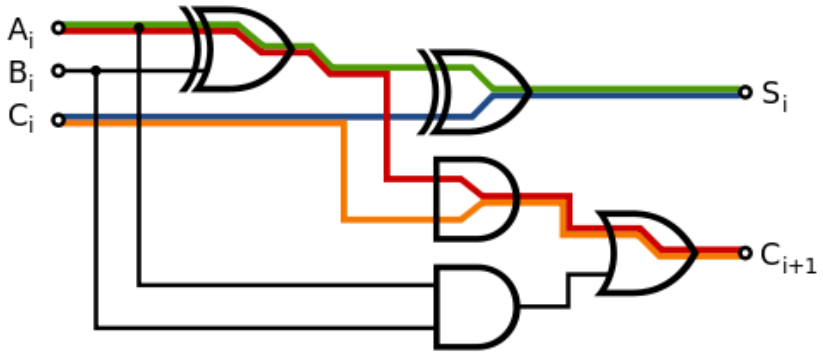


- Tempo de Propagação – analogia com o movimento ondulatório

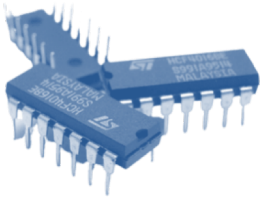
Noção de Circuito Combinatório



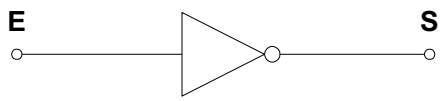
- Circuito Combinatório:
  - Até ao momento, tem-se assumido um modelo **ideal** dos circuitos lógicos, em que a saída muda instantaneamente face aos valores na entrada do circuito.
  - Na realidade, todos os circuitos caracterizam-se por um certo tempo de propagação, entre as entradas e as saídas, e que depende no número e complexidade das portas lógicas envolvidas:

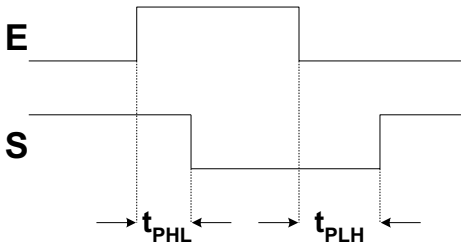


Tempos de Propagação

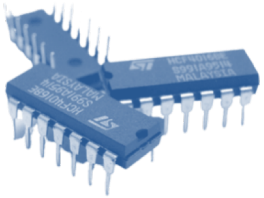


- Tempo de Propagação:
  - Corresponde ao intervalo de tempo necessário para que uma alteração na entrada se propague até à saída de uma determinada porta lógica ou circuito combinatório.
    - t_{PHL} - Tempo de propagação de H para L na saída, desde a variação da entrada.
    - t_{PLH} - Tempo de propagação de L para H na saída, desde a variação da entrada.





Tempos de Propagação



- Tempo de Propagação:
  - Exemplo (para TTL LS):
    - Valores Típicos: 8 a 10 ns
    - Valores Máximos: 15 a 20 ns
  - **ATENÇÃO:** Em geral, os tempos de propagação aumentam com o número de entradas ligadas à saída da porta lógica (fan-out).
  - Na determinação do atraso máximo na propagação de um sinal através de um circuito combinatório consideram-se, sempre, os valores máximos.

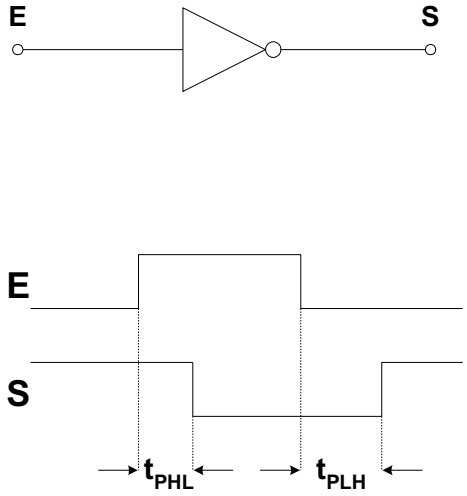
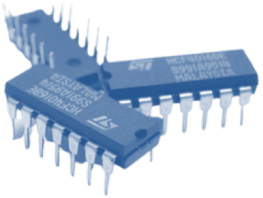
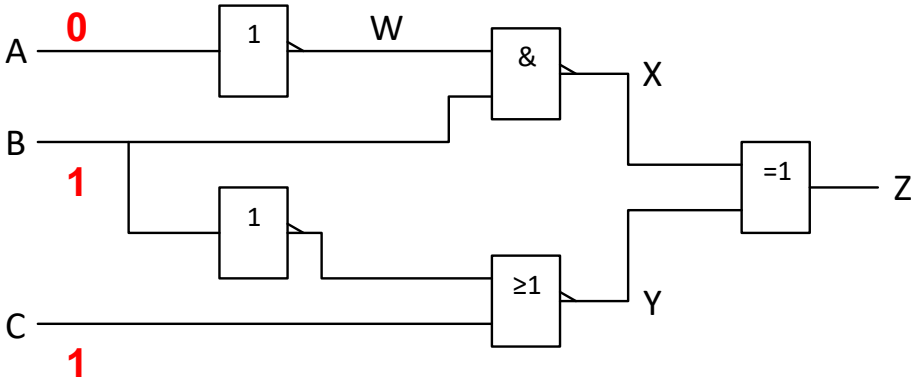


Diagrama Temporal

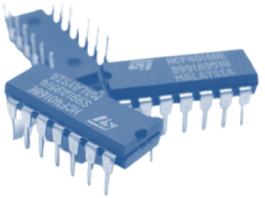


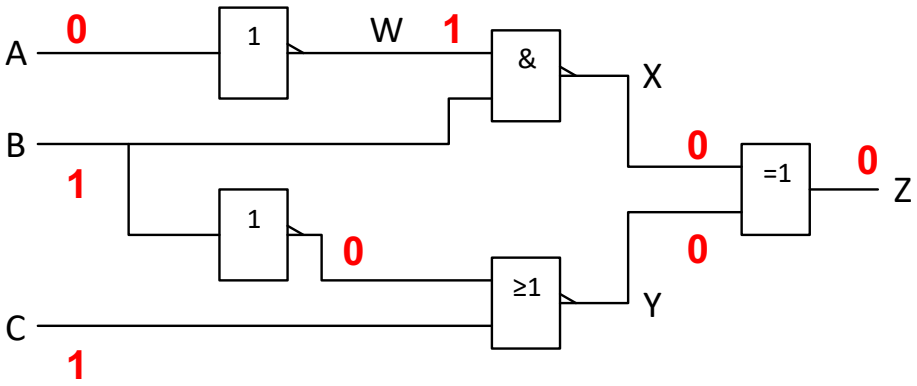


	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Estado inicial

Diagrama Temporal





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Estado inicial

A 0

B 1

C 1

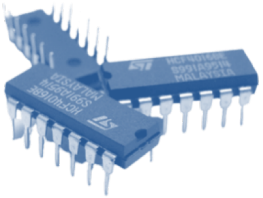
W 1

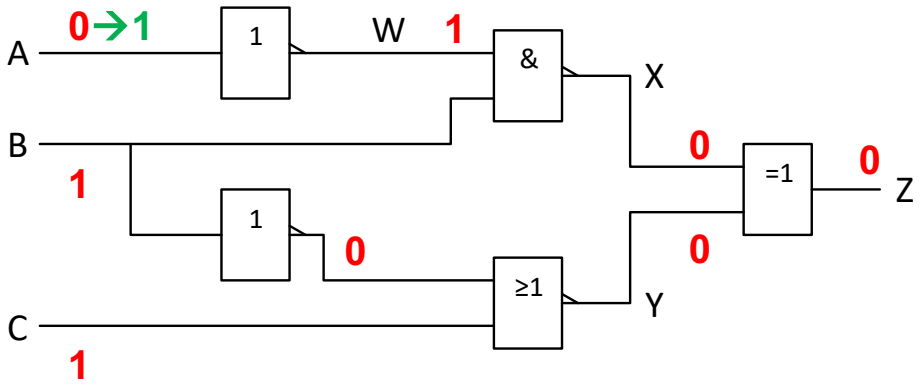
X 0

Y 0

Z 0

Diagrama Temporal

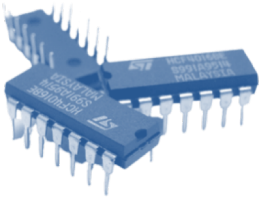


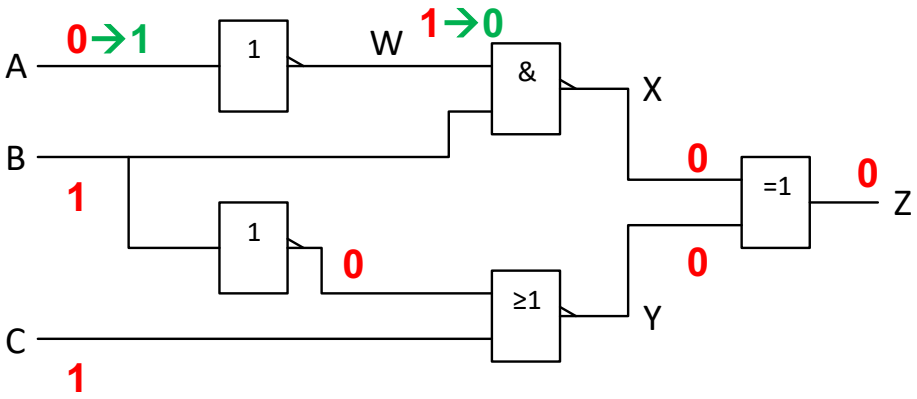


	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Transição da Entrada A

Diagrama Temporal

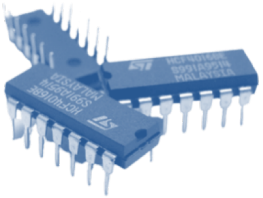


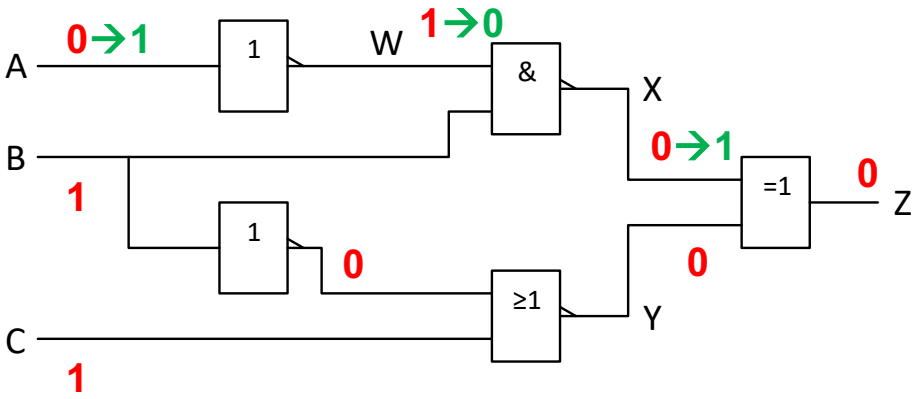


	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Transição da Entrada A

Diagrama Temporal





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Transição da Entrada A

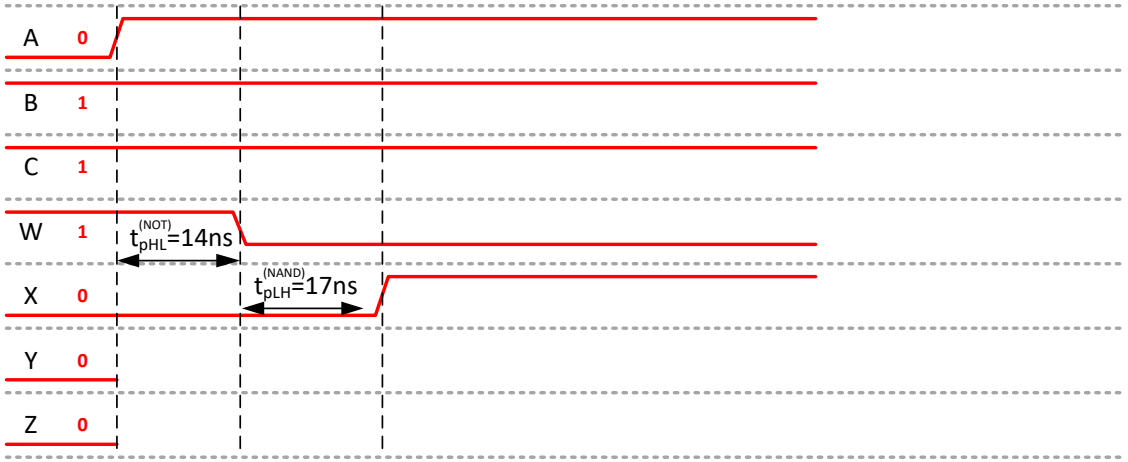
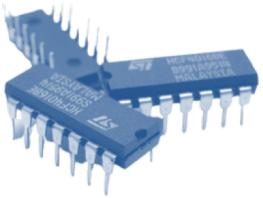
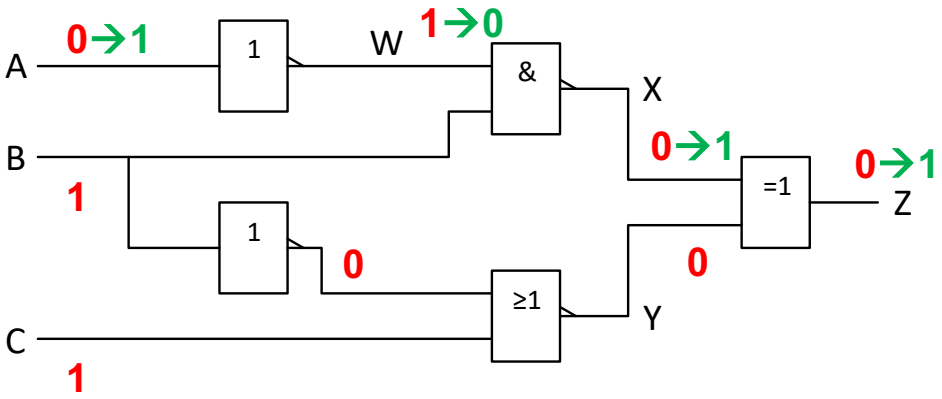


Diagrama Temporal





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Transição da Entrada A

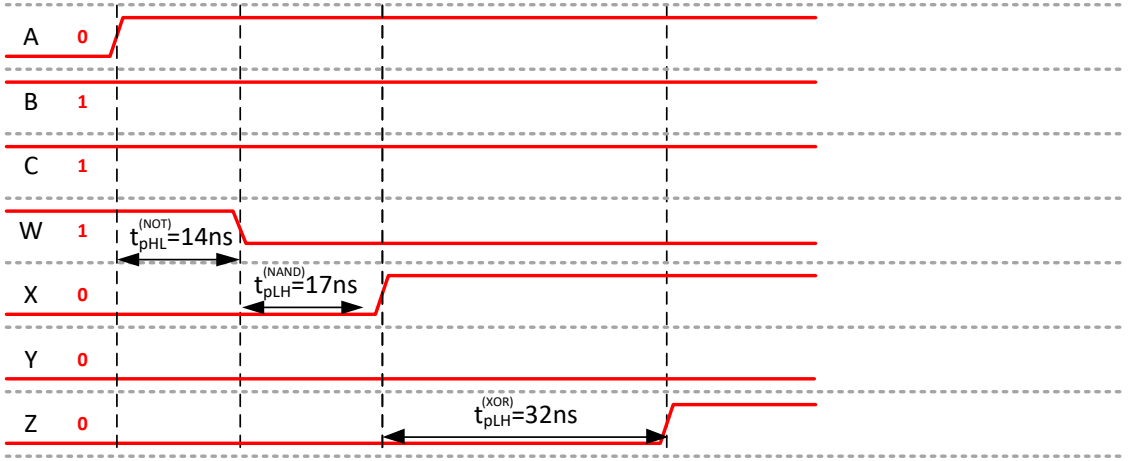
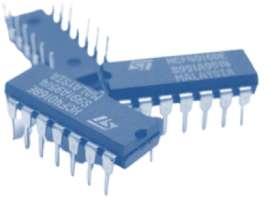
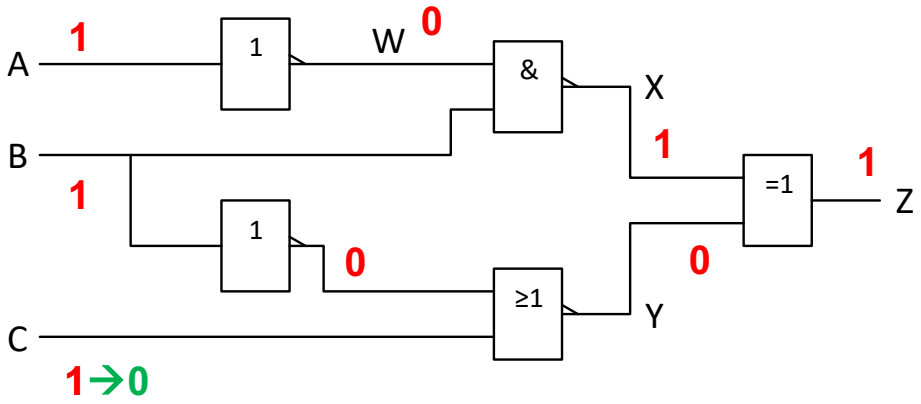


Diagrama Temporal





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Transição da Entrada C

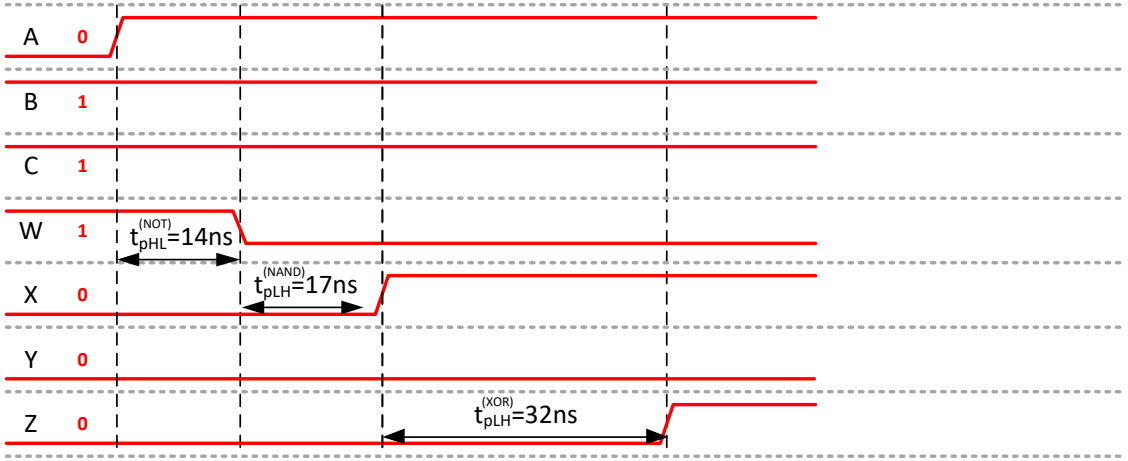
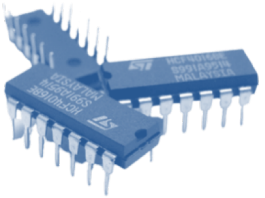
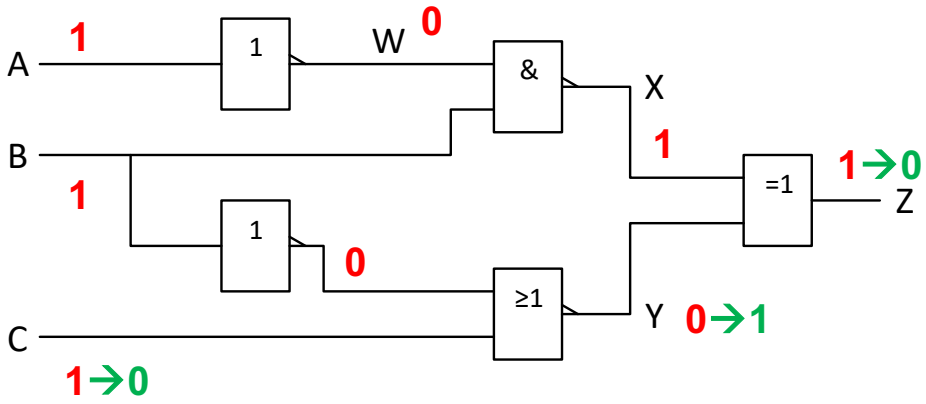


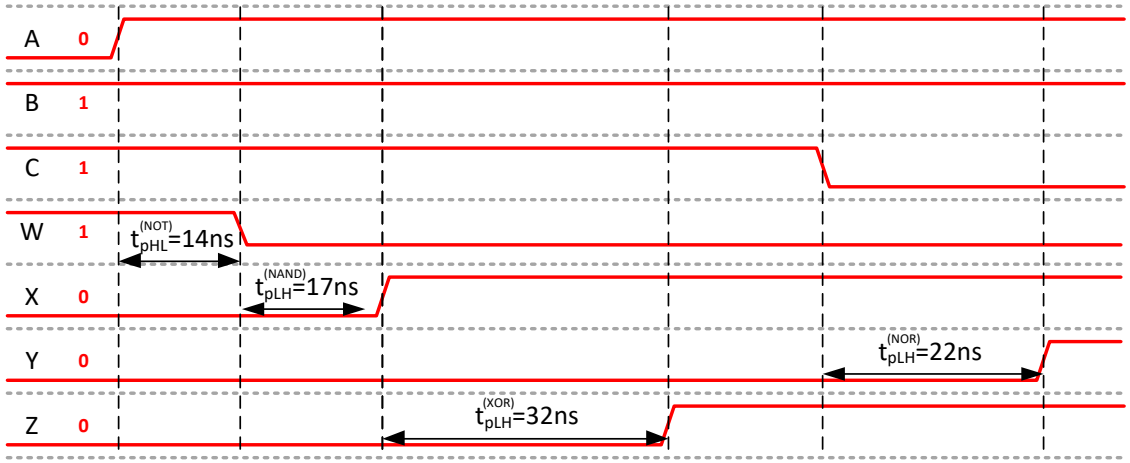
Diagrama Temporal



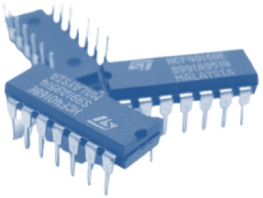


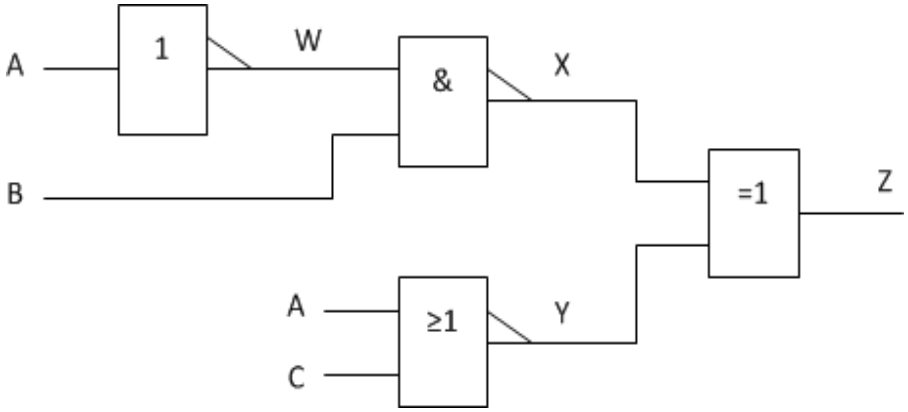
	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

Transição da Entrada C



Cálculo do Atraso Máximo





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

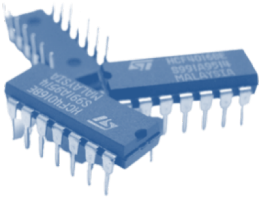
- O atraso máximo de um dado circuito é calculado como: $t_p = \max\{ t_{pLH} ; t_{pHL} \}$

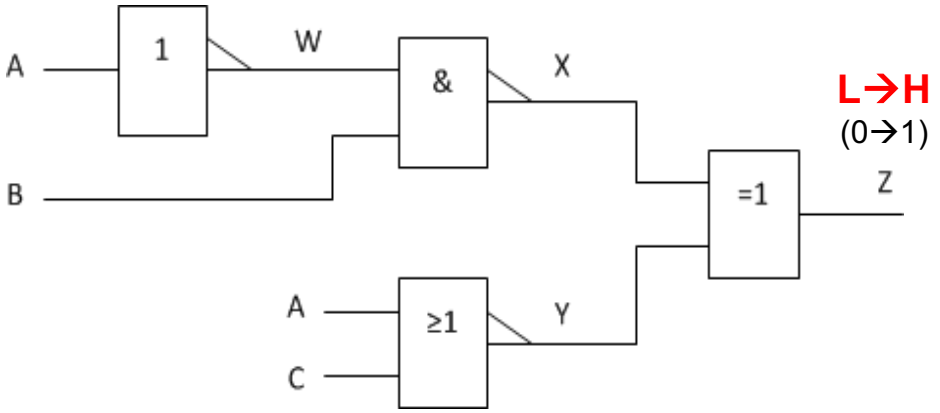
  em que:

  t_{pLH} - máximo tempo de propagação de uma qualquer entrada para a saída que leva a saída a transitar de Low para High

  t_{pHL} - máximo tempo de propagação de uma qualquer entrada para a saída que leva a saída a transitar de High para Low

Cálculo do Atraso Máximo



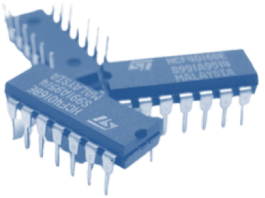


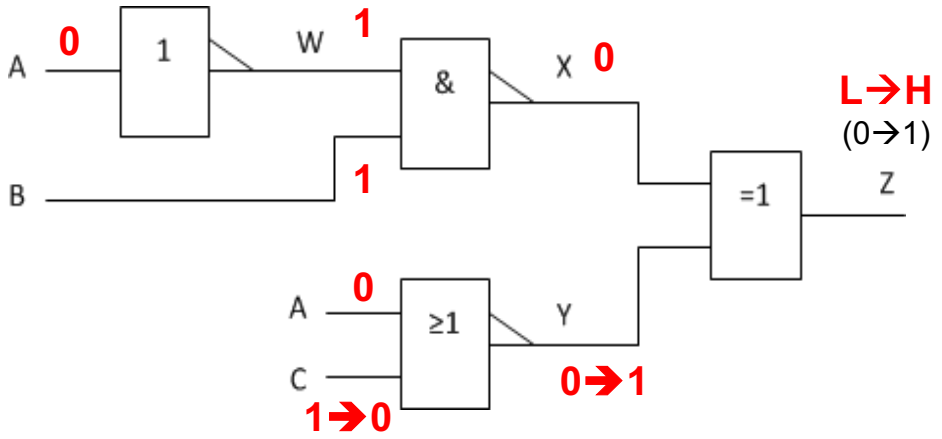
	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis... E depois escolhendo o pior:

1. X=0 , Y=0 → 1
2. X=0 → 1 , Y=0
3. X=1 , Y=1 → 0
4. X=1 → 0 , Y=1

Cálculo do Atraso Máximo





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis...
  E depois escolhendo o pior:

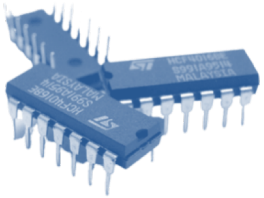
1. X=0 , Y=0 → 1
2. X=0 → 1 , Y=0
3. X=1 , Y=1 → 0
4. X=1 → 0 , Y=1

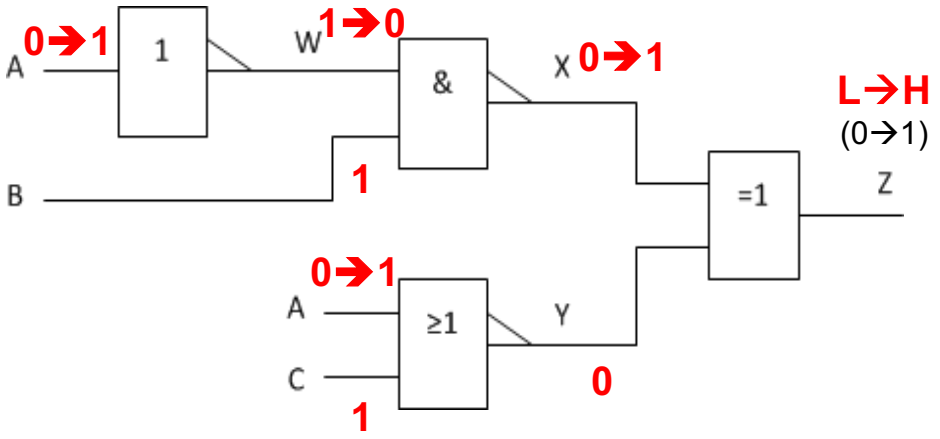
X=0 → B=1, W=1 , A=0

Logo C transita 1 → 0

$t_{pLH} = t_{pLH}(NOR) + t_{pLH}(XOR) = 54ns$

Cálculo do Atraso Máximo





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis... E depois escolhendo o pior:

1. X=0 , Y=0 → 1
2. X=0 → 1 , Y=0
3. X=1 , Y=1 → 0
4. X=1 → 0 , Y=1

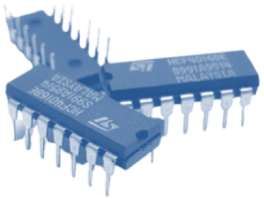
O pior caso corresponde à transição vir da porta NOT:

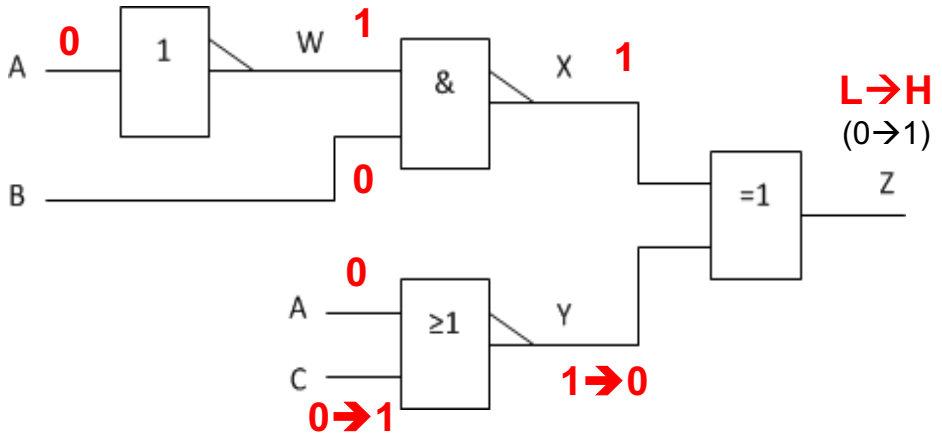
A transita 0 → 1

$t_{pLH} = t_{pHL}(NOT) + t_{pLH}(NAND) + t_{pLH}(XOR)$

$= 14+17+32=63ns$

Cálculo do Atraso Máximo





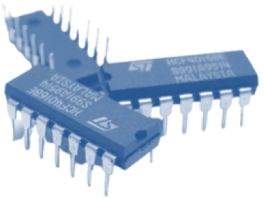
	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

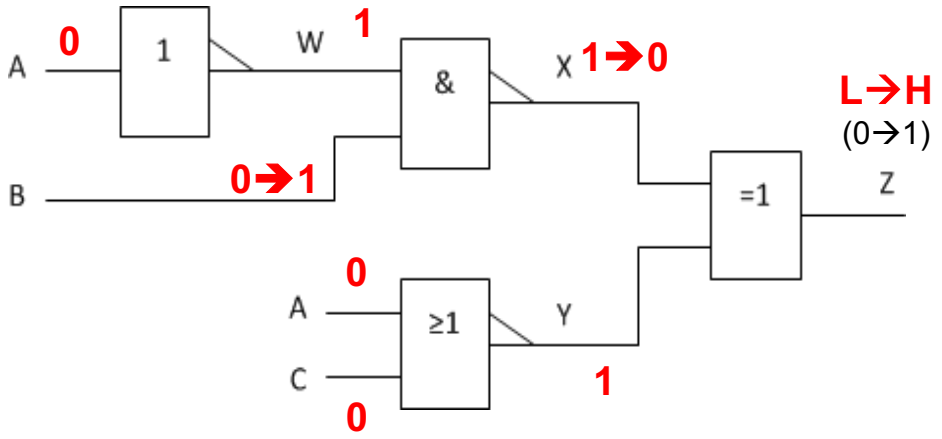
- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis... E depois escolhendo o pior:

1. X=0 , Y=0 → 1
2. X=0 → 1 , Y=0
3. X=1 , Y=1 → 0
4. X=1 → 0 , Y=1

$t_{pLH} = t_{pHL}(NOR) + t_{pLH}(XOR) =$
$= 24+32=56ns$

Cálculo do Atraso Máximo





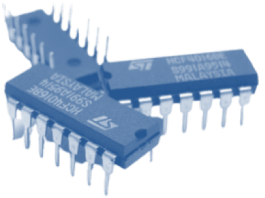
	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

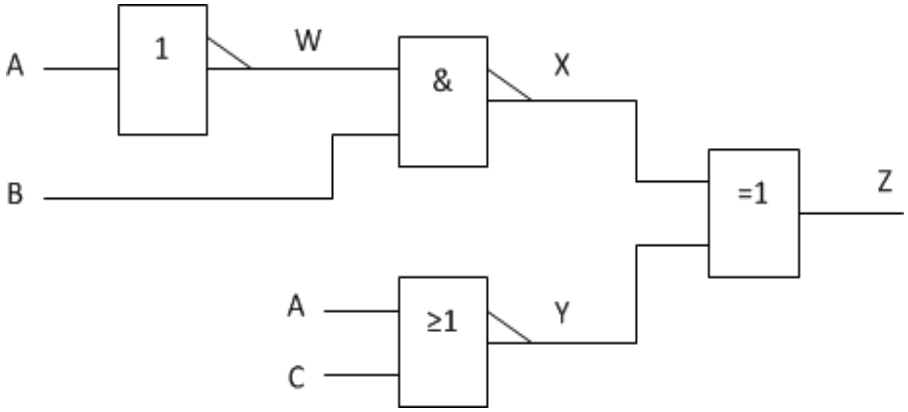
- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis... E depois escolhendo o pior:

1. X=0 , Y=0 → 1
2. X=0 → 1 , Y=0
3. X=1 , Y=1 → 0
4. X=1 → 0 , Y=1

$t_{pLH} = t_{pHL}(NAND) + t_{pLH}(XOR) =$
$= 15+32=47ns$

Cálculo do Atraso Máximo





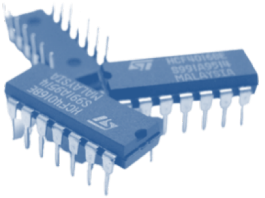
	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

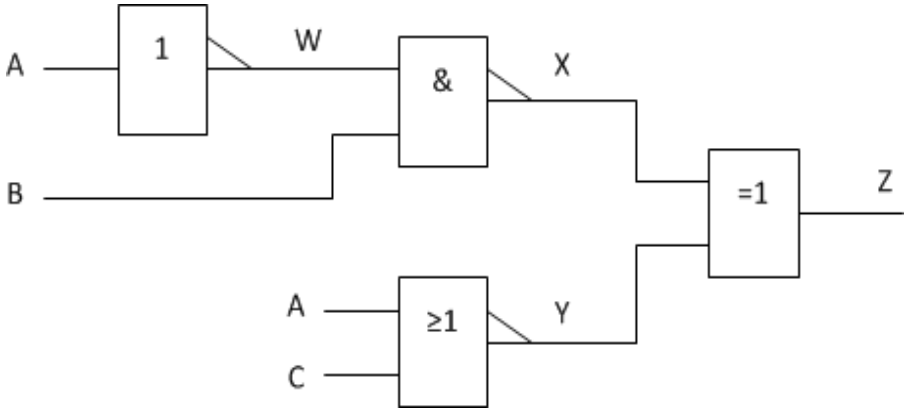
- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis... E depois escolhendo o pior:
  1. X=0 , Y=0 → 1 ➞ t_{pLH} = 54ns
  2. X=0 → 1 , Y=0 ➞ t_{pLH} = 63ns
  3. X=1 , Y=1 → 0 ➞ t_{pLH} = 56ns
  4. X=1 → 0 , Y=1 ➞ t_{pLH} = 47ns

Tempo máximo de propagação

$t_{pLH} = 63\ ns$

Cálculo do Atraso Máximo





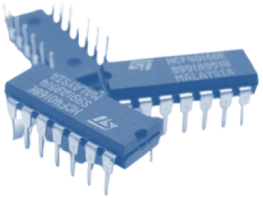
	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

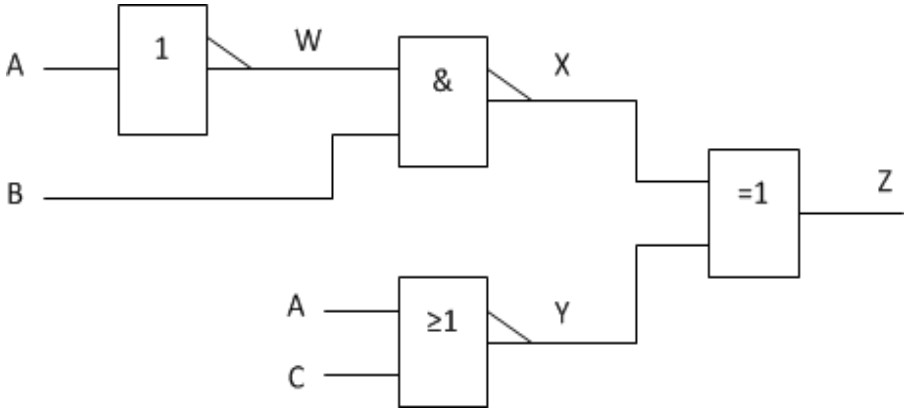
- O cálculo do tempo de propagação t_{pLH} é calculado verificando todos os casos possíveis... E depois escolhendo o pior:

1. X=0 , Y=0 → 1
2. X=0 → 1 , Y=0
3. X=1 , Y=1 → 0
4. X=1 → 0 , Y=1

- **Ver todos os casos possíveis...**
- **verificar qual é o pior!!!**

Cálculo do Atraso Máximo





	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	15ns	17ns
NOR	24ns	22ns
XOR	30ns	32ns

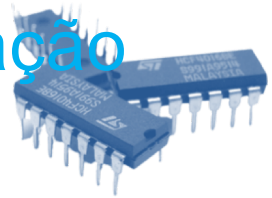
- O atraso máximo de um dado circuito é calculado como: $t_p = \max\{ t_{pLH} ; t_{pHL} \}$

  em que:

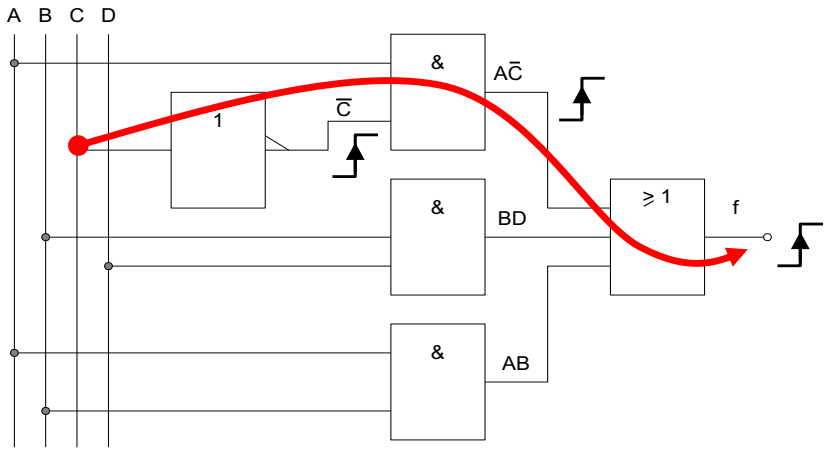
  t_{pLH} - máximo tempo de propagação de uma qualquer entrada para a saída que leva a saída a transitar de Low para High

  t_{pHL} - máximo tempo de propagação de uma qualquer entrada para a saída que leva a saída a transitar de High para Low

Cálculo do Caminho com Atraso de Propagação Máximo



- Exemplo:



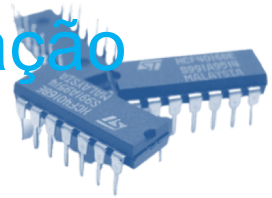
O caminho de atraso máximo é activado quando C comuta e A=1, B.D=0 e A.B=0.

$$t_{PLHtotal} = t_{PLHnot} + t_{PLHand} + t_{PLHor}$$

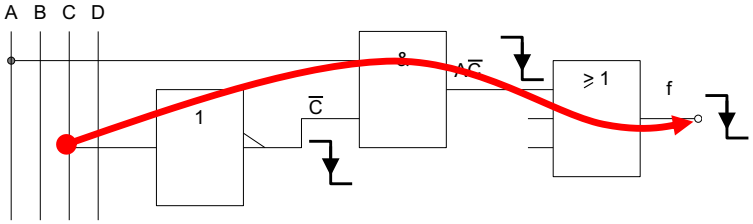
ou

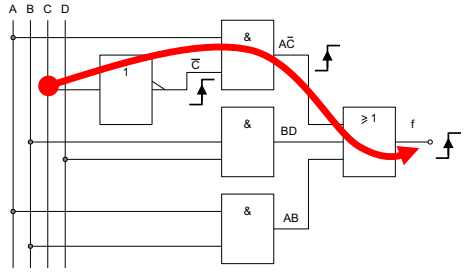
$$t_{PHLtotal} = t_{PHLnot} + t_{PHLand} + t_{PHLor}$$

Cálculo do Caminho com Atraso de Propagação Máximo



- Exemplo (cont.):



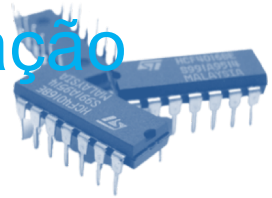


$$t_{P\max} = \max(14ns + 17ns + 24ns; 15ns + 20ns + 22ns)$$

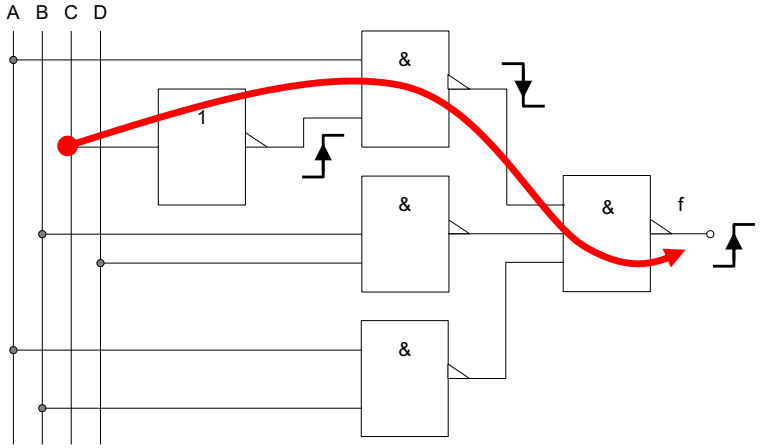
$$= \max(55ns; 57ns) = 57ns$$

	t_{PHL}	t_{PLH}
NOT	14ns	15ns
AND	17ns	20ns
OR	24ns	22ns

Cálculo do Caminho com Atraso de Propagação Máximo **(com NANDs)**



- Exemplo:



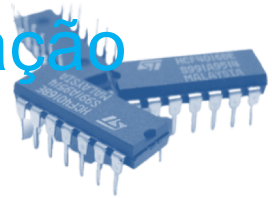
O caminho de atraso máximo é activado quando C comuta e A=1, $\overline{B.D}$ =1 e $\overline{A.B}$ =1.

$$t_{PLHtotal} = t_{PLHnot} + t_{PHLnand1} + t_{PLHnand2}$$

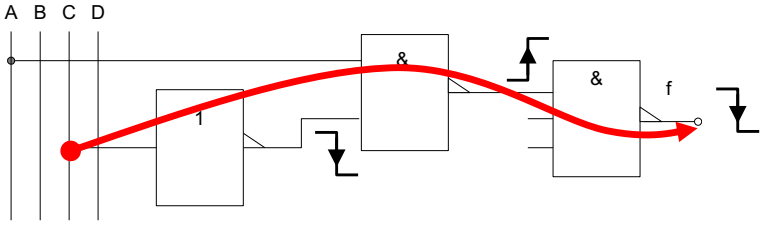
ou

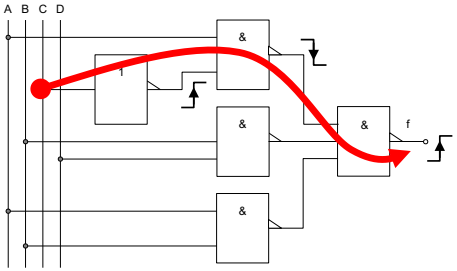
$$t_{PHLtotal} = t_{PHLnot} + t_{PLHnand1} + t_{PHLnand2}$$

Cálculo do Caminho com Atraso de Propagação Máximo **(com NANDs)**



- Exemplo (cont.)



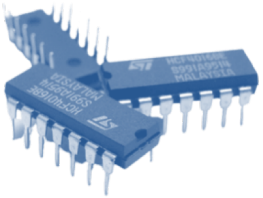


	t_{PHL}	t_{PLH}
NOT	14ns	15ns
NAND	17ns	16ns

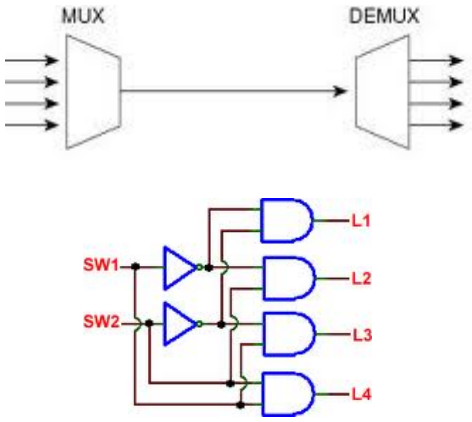
$$t_{P\max} = \max(14ns + 16ns + 17ns\,;\,15ns + 17ns + 16ns)$$

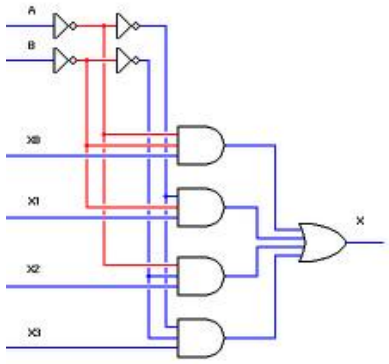
$$= \max(47ns\,;\,48ns) = 48ns$$

TÉCNICO LISBOA

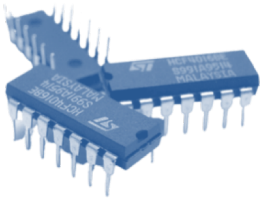


Circuitos combinatórios simples





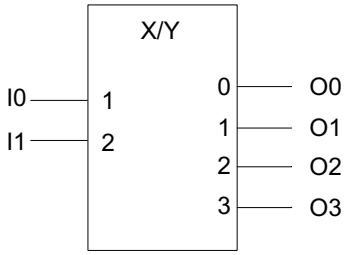
Descodificador



- Descodificador (em inglês, Decoder)
  - O descodificador binário é um circuito combinatório que permite, perante uma combinação de entradas, activar uma e só uma saída.

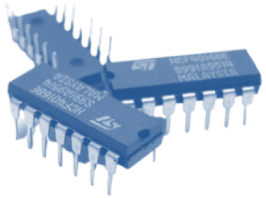
I1	I0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

DESCODIFICADOR 2:4



  - No símbolo do componente, o índice dos sinais de entrada/saída permite identificar claramente as saídas e o "peso" de cada um dos sinais de entrada.

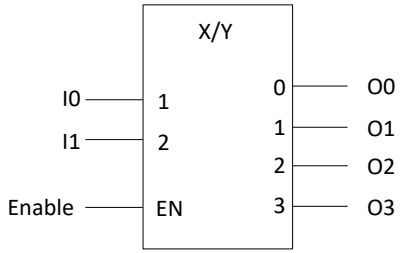
Descodificador



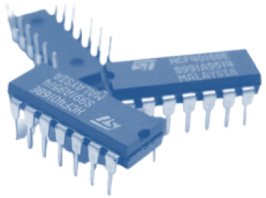
- Descodificador com entrada de ativação (*Enable*):
  - A entrada de **ENABLE** permite, quando ativa (neste caso, a "1"), que o descodificador funcione normalmente. Quando não ativa, inibe o seu funcionamento fazendo com que todas as saídas fiquem inativas (neste caso, todas a "0").

EN	I1	I0	O0	O1	O2	O3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0

DESCODIFICADOR 2:4



Descodificador

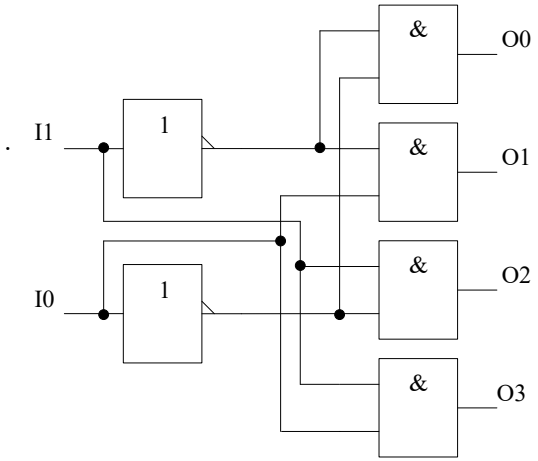


- Descodificador: estrutura interna
  - A figura representa a estrutura interna de um descodificador binário de 2 entradas.
  - Cada saída representa uma das combinações possíveis das entradas

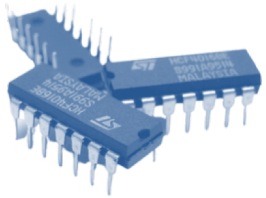
I1	I0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$O_0 = \overline{I_1}.\overline{I_0} \qquad O_2 = I_1.\overline{I_0}$$

$$O_1 = \overline{I_1}.I_0 \qquad O_3 = I_1.I_0$$

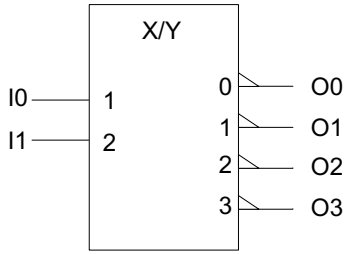


Descodificador

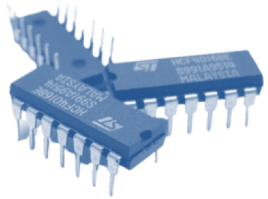


- Descodificador com saídas ativas a zero
  - No símbolo do componente, o Δ na saída indica que esta é ativa a “0”, i.e., a saída selecionada tem um “0” e as outras têm um “1”.
    (funciona como se tivesse um inversor na saída)

I1	I0	O0	O1	O2	O3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

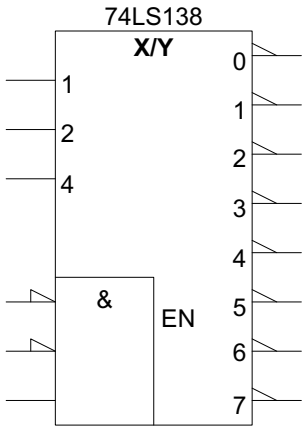


Descodificador

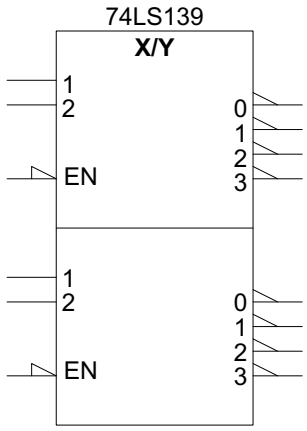


- Descodificadores: exemplos de componentes

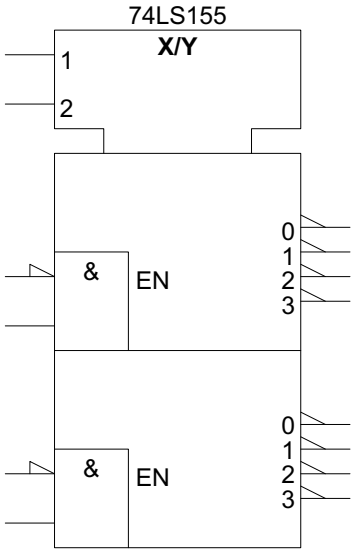
DESCODIFICADOR 3:8



DUAL DECODER 2:4

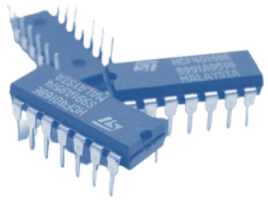


DUAL DECODER 2:4



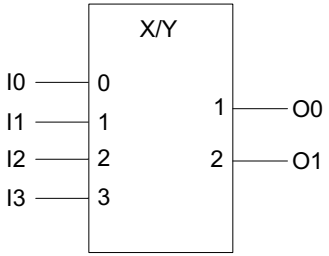
- Nos 3 exemplos os sinais de saída são ativos a zero.
- No 138 o Enable é um AND de 3 entradas, 2 delas negadas.
  No 139 o Enable é ativo a zero.
  No 155 o Enable é um AND de 2 entradas, 1 delas negada.

Codificador



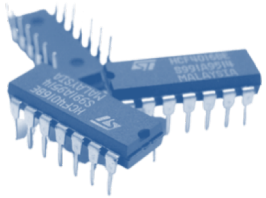
- Codificador (em inglês, encoder):
  - O codificador binário é um circuito combinatório que indica qual das entradas possíveis é que está ativa (neste caso, a “1”).

I3	I2	I1	I0	O1	O0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



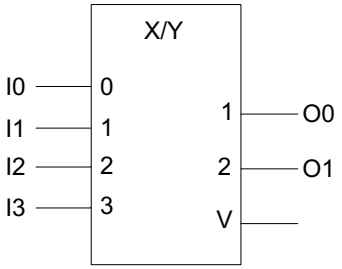
  - Nesta versão simples, o codificador só considera 4 das 16 combinações possíveis de entrada.
  - O circuito não distingue a situação de todas as entradas estarem a “0”.
  - O circuito não distingue as situações em que estão a “1” mais do que uma entrada.

Codificador



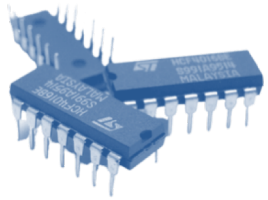
- Codificador de prioridade:
  - A saída V indica se existe pelo menos uma entrada activa (a “1”).

I3	I2	I1	I0	O1	O0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1



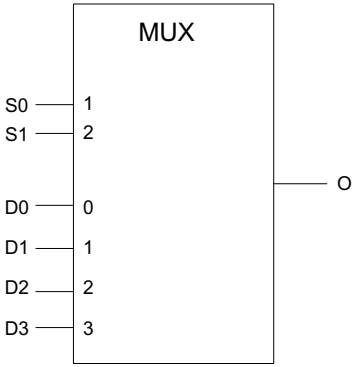
  - As entradas deste codificador têm uma ordem de prioridades: em caso de mais de uma entrada ativa (a “1”) é considerada a de maior prioridade.
  - A entrada I3 é a de maior prioridade, seguida da I2, da I1, e a I0 é a de menor prioridade.

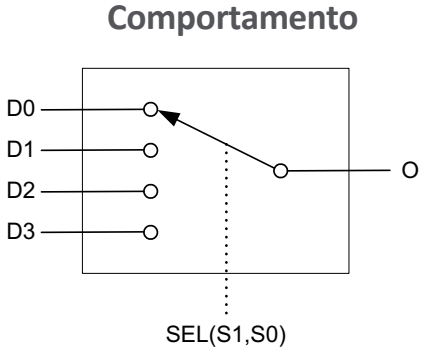
Multiplexer



- Multiplexer:
  - O multiplexer é um circuito combinatório que permite, através da especificação dos sinais de seleção, encaminhar uma das N entradas de dados para a saída.

    Exemplo: multiplexer 4:1

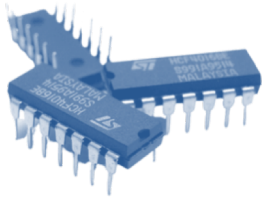




S1	S0	O
0	0	D0
0	1	D1
1	0	D2
1	1	D3

  - As entradas de seleção determinam a entrada de dados cujo valor é colocado na saída.

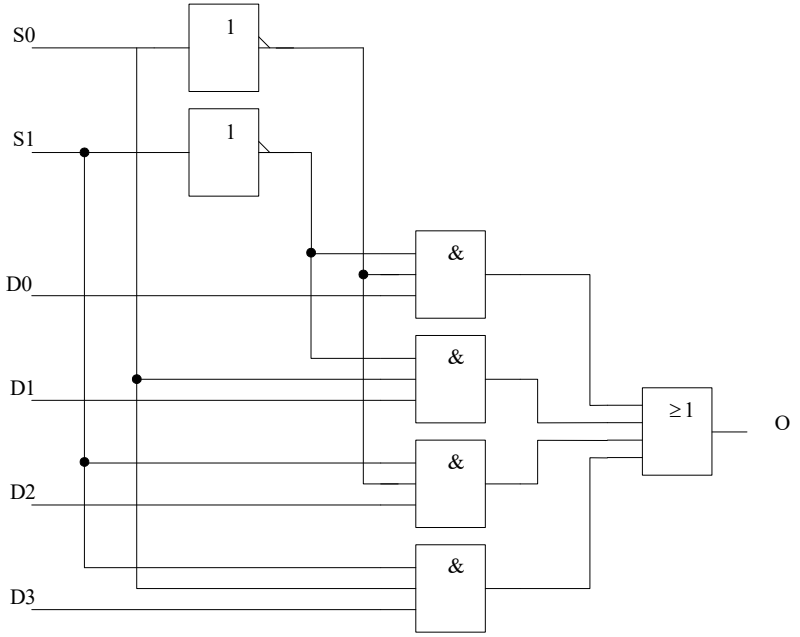
Multiplexer



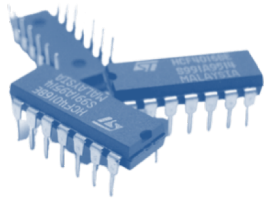
- Multiplexer: estrutura interna

S1	S0	O
0	0	D0
0	1	D1
1	0	D2
1	1	D3

$$O = D_0 . \overline{S_1} . \overline{S_0} + D_1 . \overline{S_1} . S_0 + D_2 . S_1 . \overline{S_0} + D_3 . S_1 . S_0$$

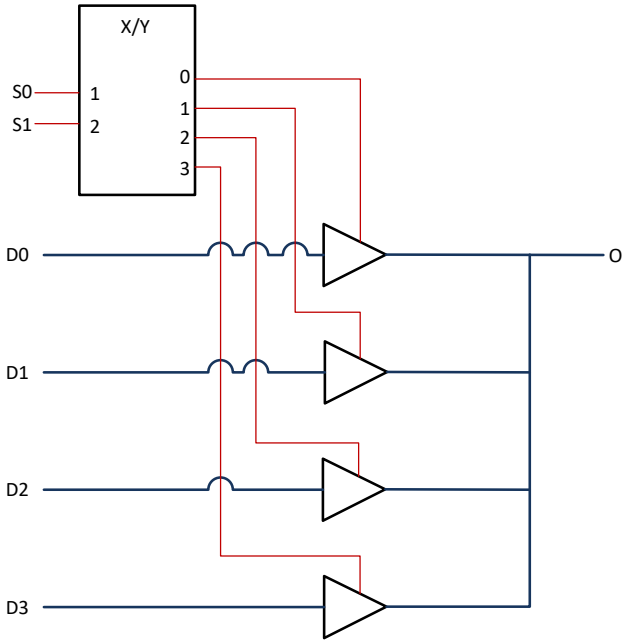


Multiplexer

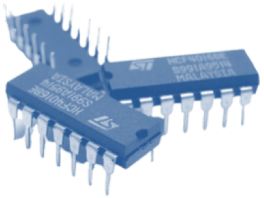


- Multiplexer: estrutura interna alternativa

S1	S0	O
0	0	D0
0	1	D1
1	0	D2
1	1	D3

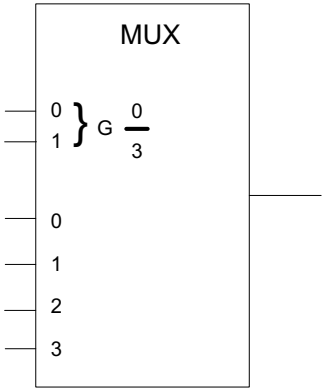


Multiplexer

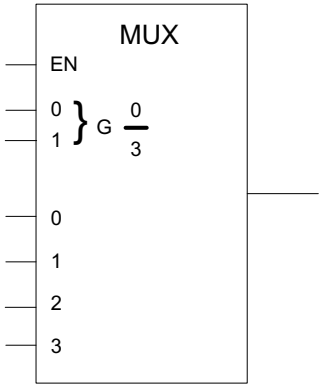


- Multiplexer: simbologia

MULTIPLEXER 4:1 simples

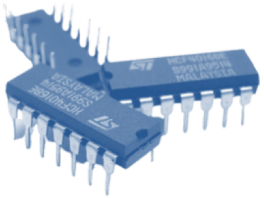


MULTIPLEXER 4:1 com enable

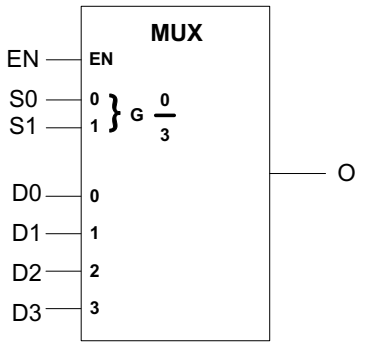


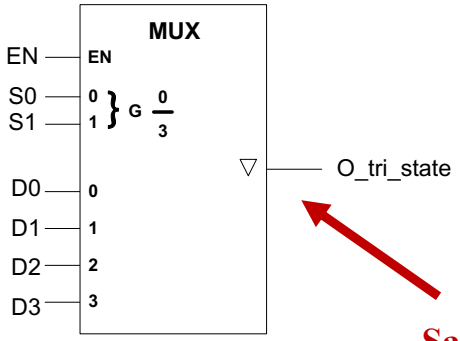
EN	S1	S0	O
1	0	0	D0
1	0	1	D1
1	1	0	D2
1	1	1	D3
0	X	X	0

Multiplexer



- Multiplexer: saída tri-state

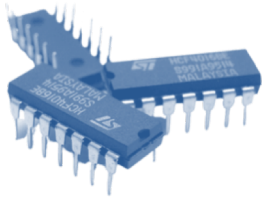




Saída tri-state

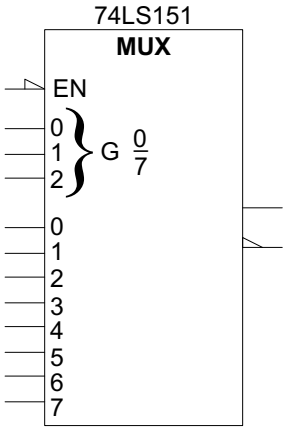
S1	S0	EN	O	O_tri_state
-	-	0	0	Z
0	0	1	D0	D0
0	1	1	D1	D1
1	0	1	D2	D2
1	1	1	D3	D3

Multiplexer

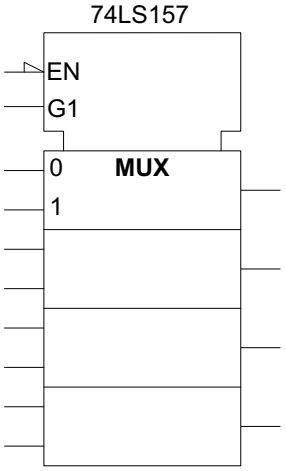


- Multiplexers: exemplos de componentes

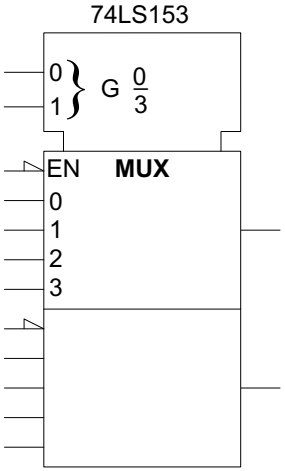
MUX 8:1



QUAD MUX 2:1

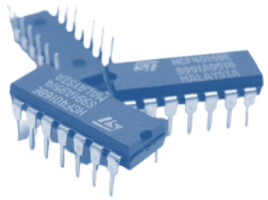


DUAL MUX 4:1



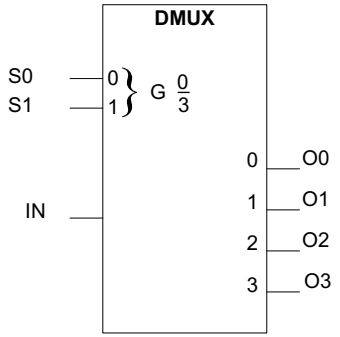
- Nos 3 exemplos, os sinais de Enable são activos a zero (a activação do funcionamento normal do componente acontece quando EN=0).
- O 74151 tem uma saída suplementar que é a negação da outra.

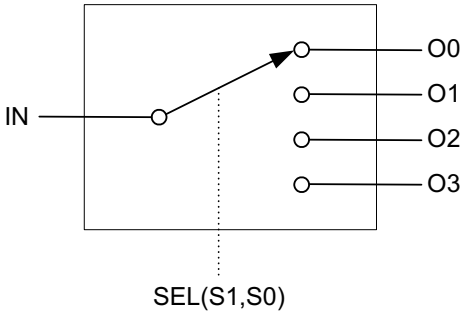
Demultiplexer



- Demultiplexer:
  - O demultiplexer é um circuito combinatório que permite, através da especificação dos sinais de seleção, encaminhar a entrada para uma das N saídas.

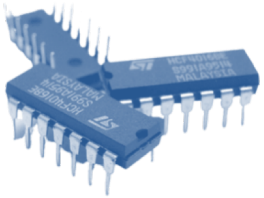
    Exemplo: Demultiplexer 1:4





S1	S0	O0	O1	O2	O3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN

Demultiplexer

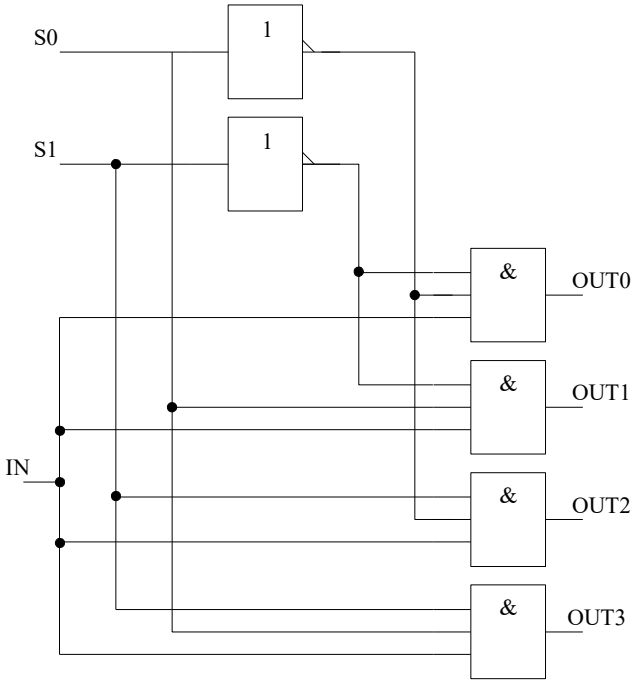


- Demultiplexer: estrutura interna

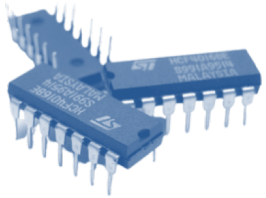
S1	S0	O0	O1	O2	O3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN

$$O_0 = IN.\overline{S_1}.\overline{S_0} \qquad O_2 = IN.S_1.\overline{S_0}$$

$$O_1 = IN.\overline{S_1}.S_0 \qquad O_3 = IN.S_1.S_0$$

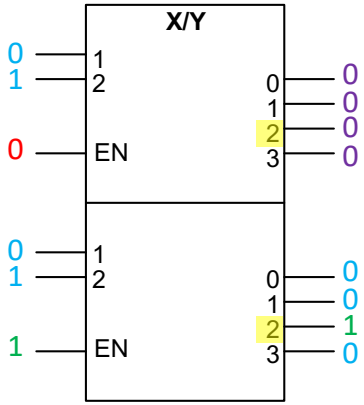


Demultiplexer

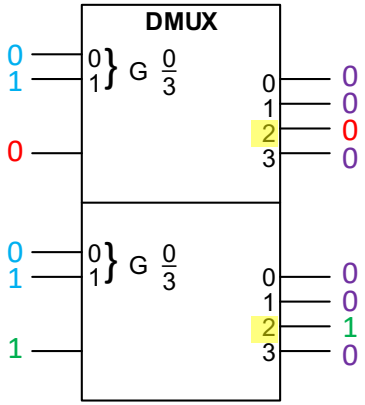


- Demultiplexeres e Descodificadores:

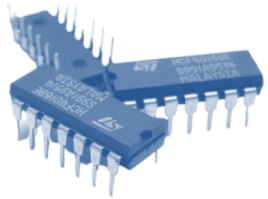
DUAL DECODER 2:4



DUAL DMUX 1:4



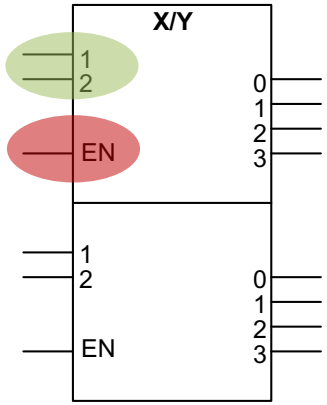
Demultiplexer



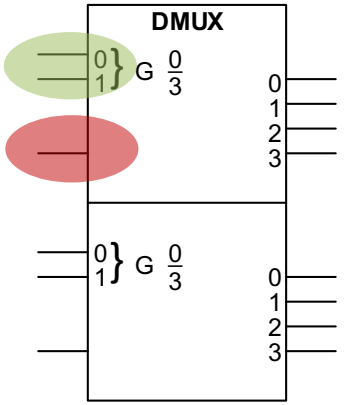
- Demultiplexeres e Descodificadores:
  - Um **descodificador** com *enable* é equivalente a um **demultiplexer**, sendo as entradas de dados do primeiro as entradas de selecção do segundo e a entrada de *enable* do primeiro a entrada de dados do segundo.

    Nota: os 2 símbolos abaixo referem a mesma funcionalidade do circuito.

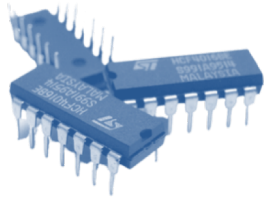
DUAL DECODER 2:4



DUAL DMUX 1:4

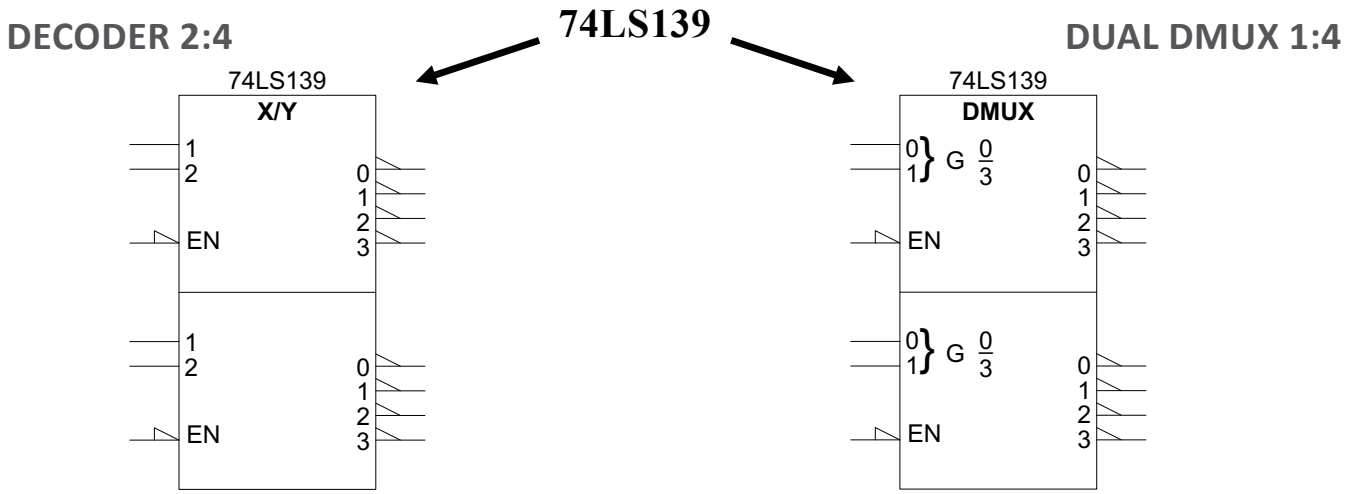


Demultiplexer

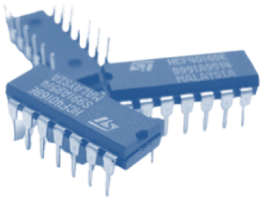


- Demultiplexeres e Descodificadores:
  - Um **descodificador** com *enable* é equivalente a um **demultiplexer**, sendo as entradas de dados do primeiro as entradas de seleção do segundo e a entrada de *enable* do primeiro a entrada de dados do segundo.

Exemplo:



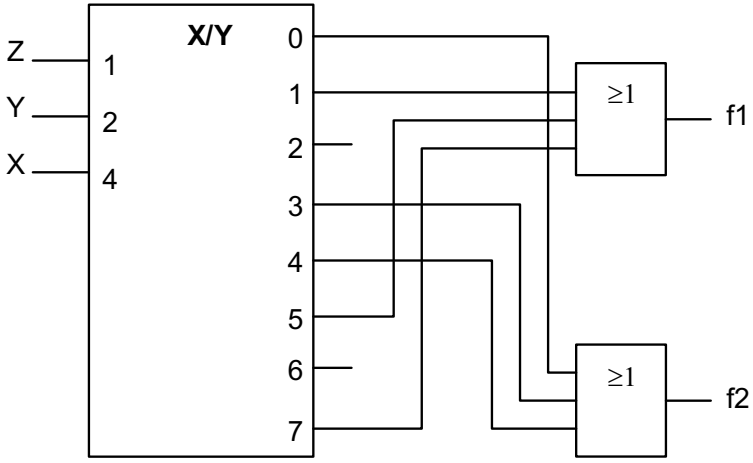
Aplicações



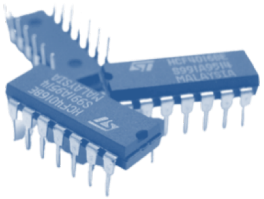
- Descodificadores: aplicações (f. combinatórias)
  - Realização de funções combinatórias de 3 variáveis com decoders 3:8

$$f1(X,Y,Z) = \sum m(1,5,7)$$

$$f2(X,Y,Z) = \sum m(0,3,4)$$



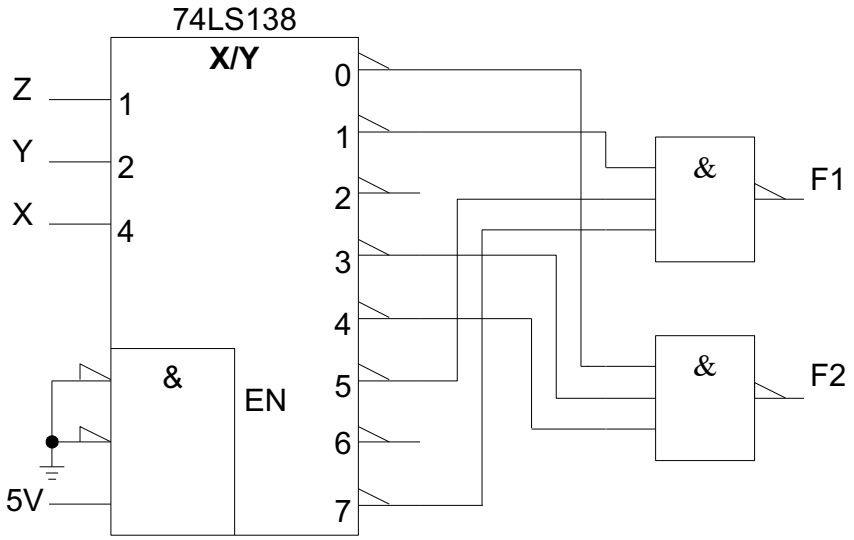
Aplicações



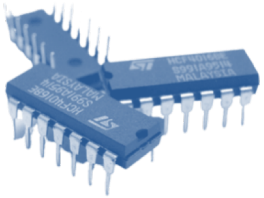
- Descodificadores: aplicações (f. combinatórias)
  - Realização de funções combinatórias de 3 variáveis com decoders 3:8 com saídas ativas a 0.

$f1(X,Y,Z) = \sum m(1,5,7)$

$f2(X,Y,Z) = \sum m(0,3,4)$



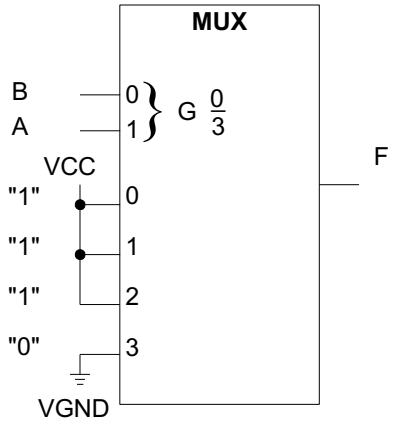
Aplicações



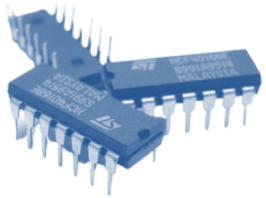
- Multiplexers: aplicações (f. combinatórias)
  - Exemplo de realização de funções combinatórias de 2 variáveis com MUX 4:1

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

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



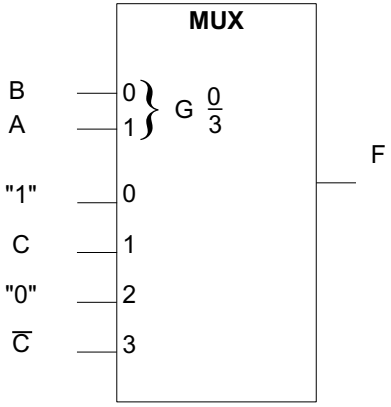
Aplicações



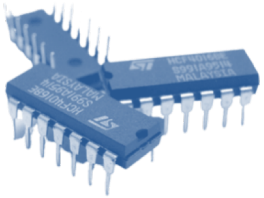
- Multiplexers: aplicações (f. combinatórias)
  - Exemplo de realização de funções combinatórias de 3 variáveis com MUX 4:1

$$F = \overline{A}\,\overline{B} + \overline{A}\,C + AB\overline{C}$$

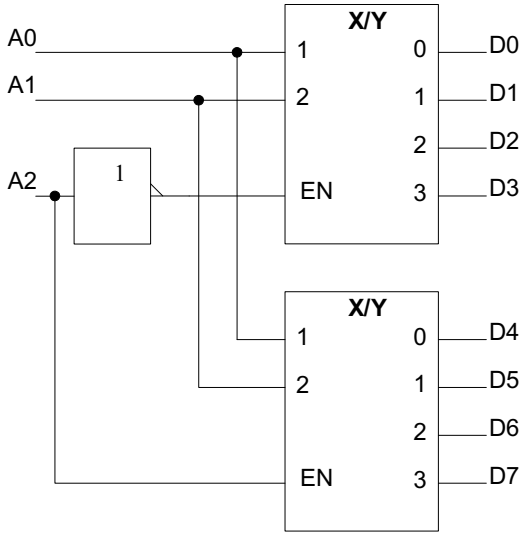
A	B	C	F	
0	0	0	1	$F = 1$
0	0	1	1	
0	1	0	0	$F = C$
0	1	1	1	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = \overline{C}$
1	1	1	0	



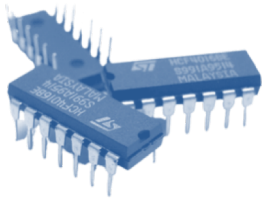
Aplicações



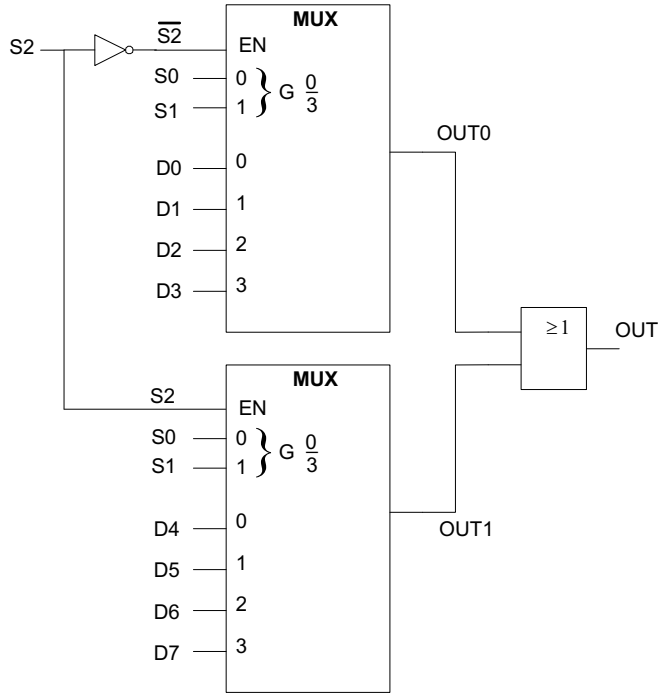
- Descodificadores: aplicações (descodificação)
  - Exemplo de realização de um DECODER 3:8 tendo por base 2 DECODERs 2:4



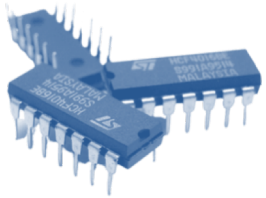
Aplicações



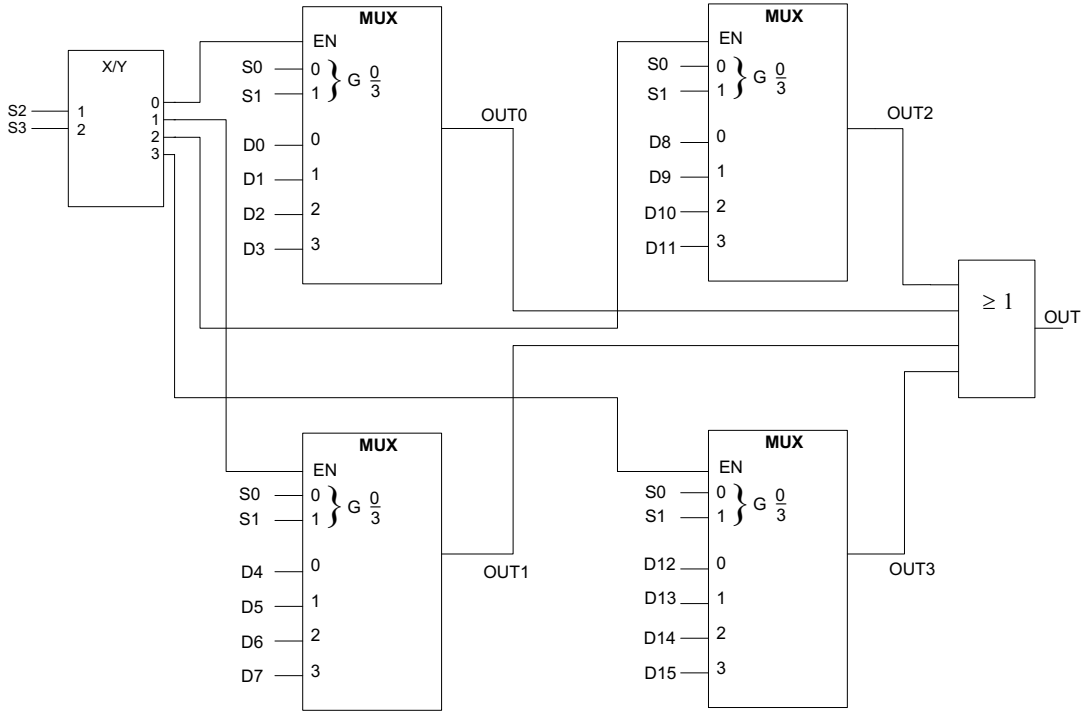
- Multiplexers: aplicações (multiplexagem)
  - Exemplo de realização de um MUX 8:1 tendo por base 2 MUXs 4:1



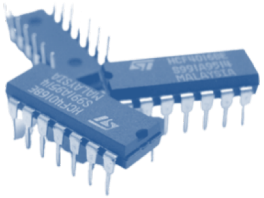
Aplicações



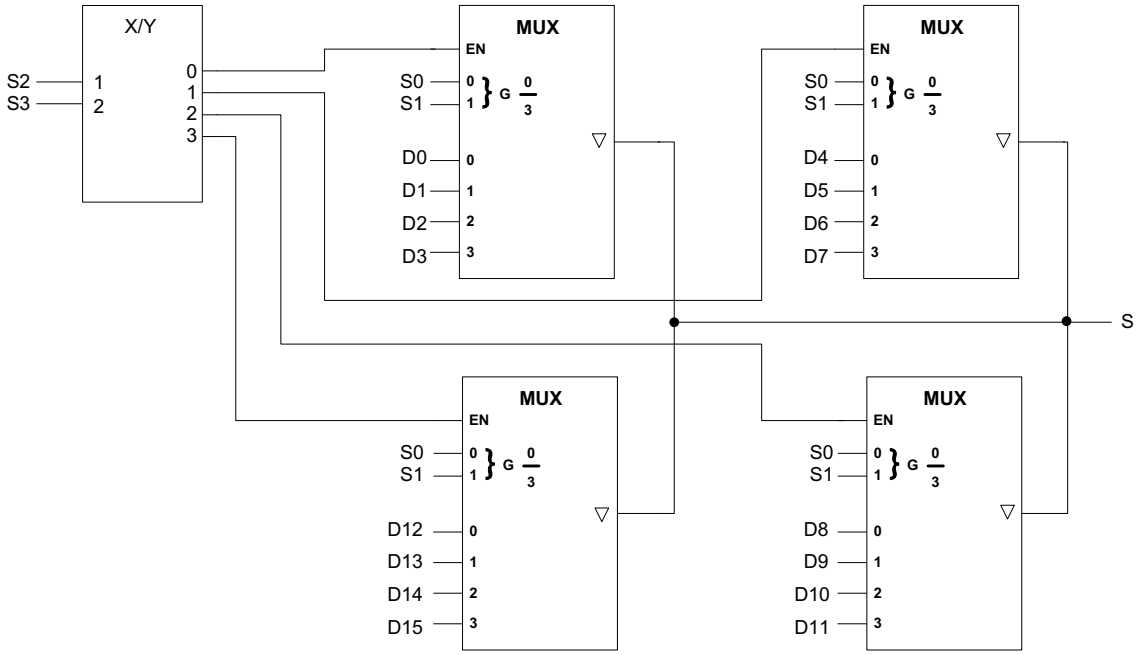
- Multiplexers: aplicações (multiplexagem)
  - Exemplo de realização de um MUX 16:1 tendo por base 4 MUXs 4:1



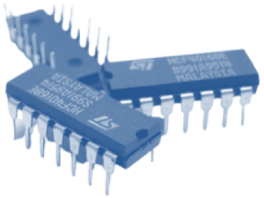
Aplicações



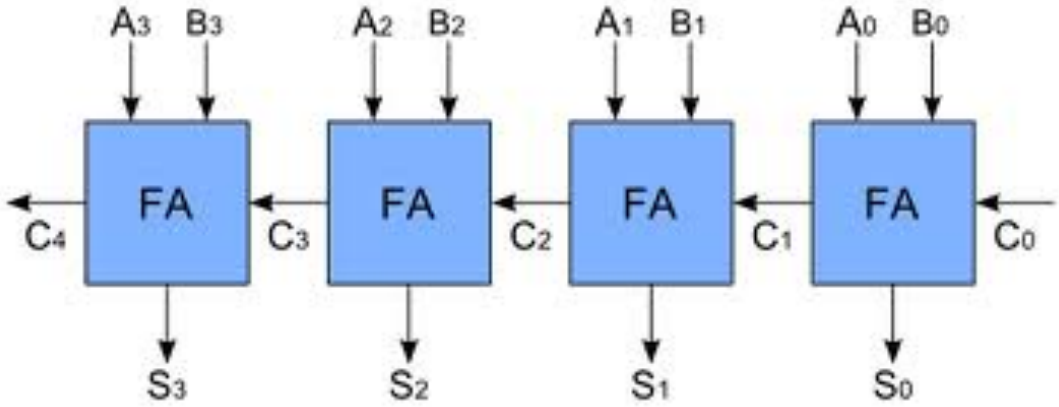
- Multiplexers: aplicações (multiplexagem)
  - Exemplo de realização de um MUX 16:1 tendo por base 4 MUXs 4:1 tri-state



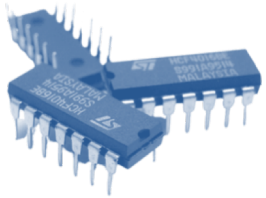
TÉCNICO LISBOA



Circuitos Aritméticos

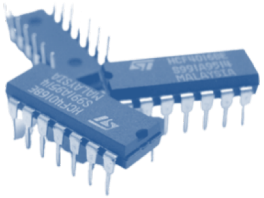


TÉCNICO LISBOA



Somadores elementares

Somadores

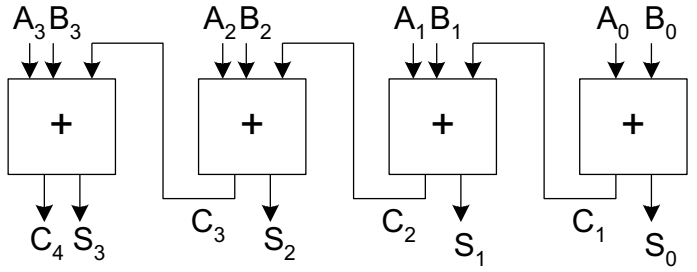


- Circuito para soma aritmética
  - Exemplo: Somador de 2 números de 4 bits cada.

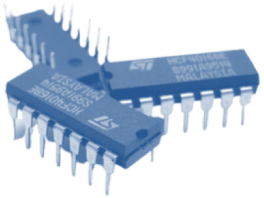
		0	1	1	0 ← Carry
	7		0	1	1 1
+	2	+	0	0	1 0
	9		1	0	0 1

  - A estrutura mais simples resolve 1 bit de cada vez:

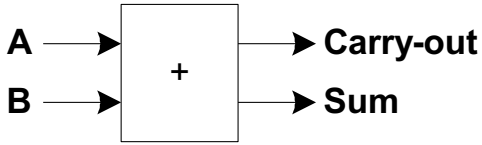
C4	C3	C2	C1	
	A3	A2	A1	A0
+	B3	B2	B1	B0
C4	S3	S2	S1	S0



Somadores

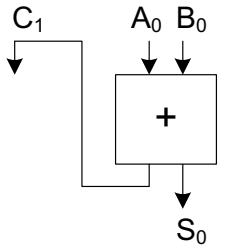


- Circuito semi-somador
  - O circuito semi-somador (em inglês, half-adder) soma 2 bits de entrada (sem transporte anterior) e produz 1 bit da soma e 1 bit de transporte.

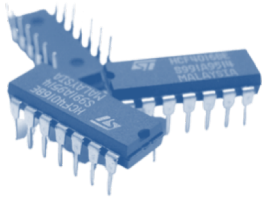


  - Corresponde p.ex. ao 1º passo do algoritmo de soma: soma os 2 bits de menor peso e obtém 1 bit S0 da soma e o transporte C1 para o passo seguinte.

C4	C3	C2	C1	
	A3	A2	A1	A0
+	B3	B2	B1	B0
C4	S3	S2	S1	S0



Somadores

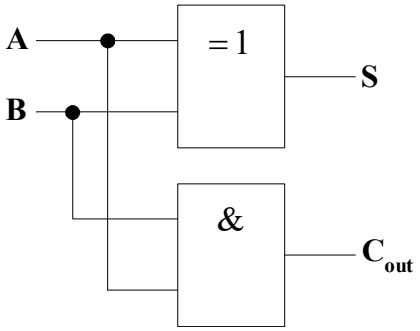


- Circuito semi-somador

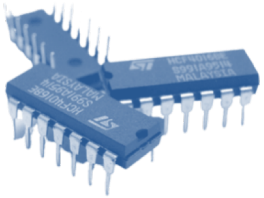
A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$C_{out} = A \cdot B$$

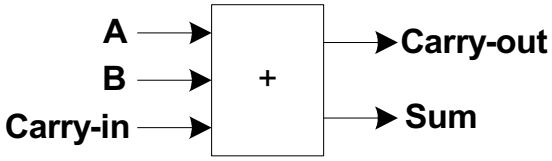
$$S = A \oplus B$$



Somadores

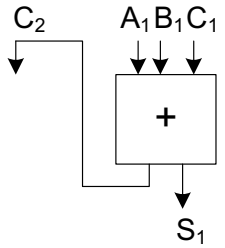


- Circuito somador completo
  - O circuito somador completo (em inglês, full-adder) soma 3 bits de entrada (incluindo o transporte anterior) e produz 1 bit da soma e 1 bit de transporte.

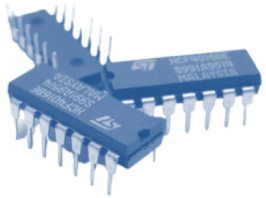


  - Por exemplo, no 2º passo: soma 3 bits A1 e B1 e o transporte C1 do passo anterior, e obtém 1 bit S1 da soma e o transporte C2 para o passo seguinte.

C4	C3	C2	C1	
	A3	A2	A1	A0
+	B3	B2	B1	B0
C4	S3	S2	S1	S0



Somadores



- Somador completo

A	B	C_{in}	C_{out}	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

S:

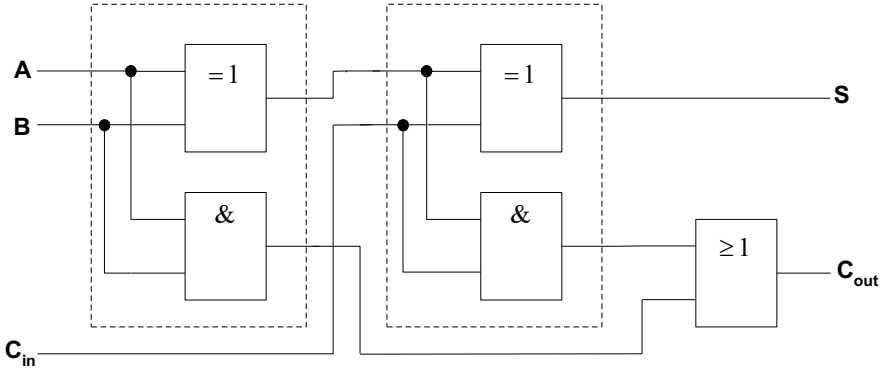
C_{in} \ A B	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = \overline{C}_{in} \cdot \overline{A} \cdot B + \overline{C}_{in} \cdot A \cdot \overline{B} + C_{in} \cdot \overline{A} \cdot \overline{B} + C_{in} \cdot A \cdot B = A \oplus B \oplus C_{in}$$

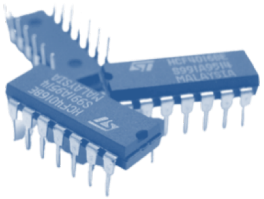
C_{out}:

C_{in} \ A B	00	01	11	10
0	0	0	1	0
1	0	1	1	1

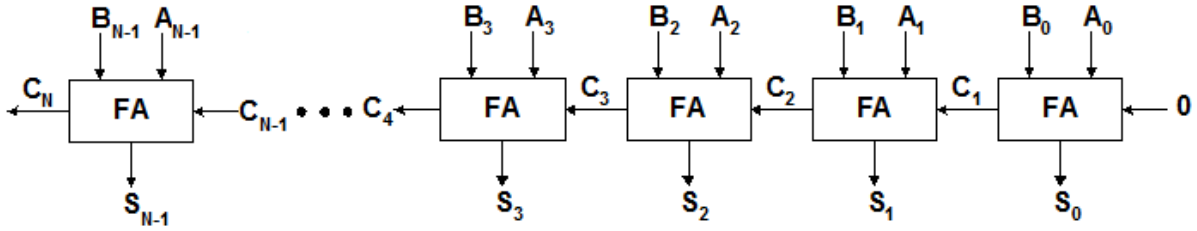
$$C_{out} = A \cdot B + C_{in} \cdot A + C_{in} \cdot B = A \cdot B + C_{in} \cdot (A \oplus B)$$



Somadores



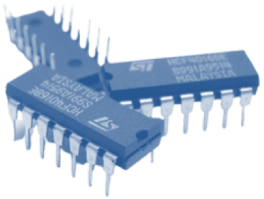
- Somador em cascata (ripple carry adder)



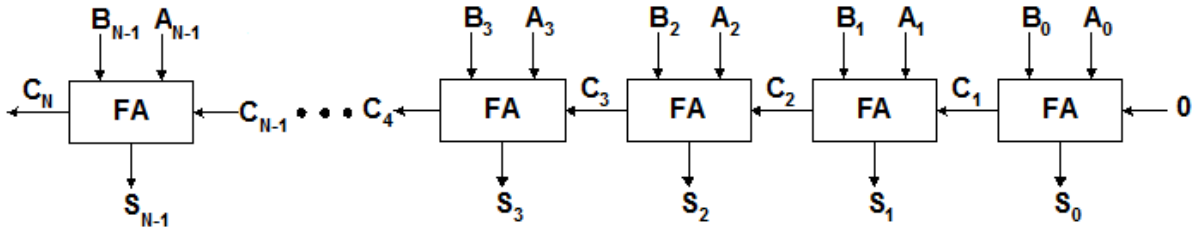
- O “Ripple Carry Adder” é o somador mais simples possível (que requer menos portas lógicas).

- Existem inúmeros circuitos alternativos para diversos compromissos velocidade/área.

Somadores



- Somador em cascata (ripple carry adder)



- A velocidade máxima de execução é limitada pela necessidade de propagar o "Carry" desde a soma do primeiro bit até à soma do bit mais significativo.
- No pior caso, o tempo de propagação do "Carry" será N x t_{PFA}.

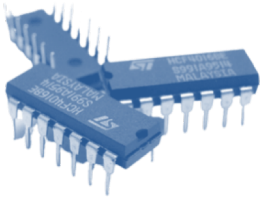
Exemplo:

	0	0	0	0
+	1	1	1	1
	1	1	1	1

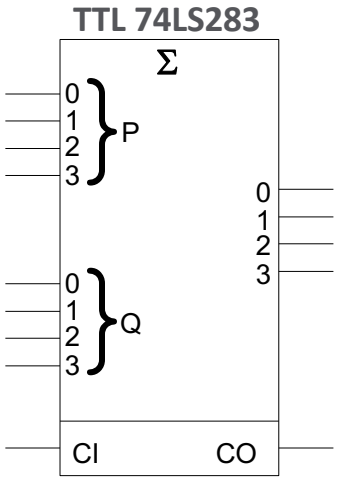
A_0 comuta de 0 para 1
$A_i = 0, \forall_{i \neq 0}$
$B_i = 1, \forall_i$

	1	1	1	1
+	0	0	0	1
	0	0	0	0

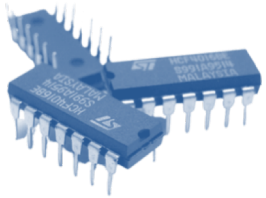
Somadores



- Exemplo: somador de 4 bits
  - 74LS283 - somador de 4 bits completo:
    - Soma:
      - 2 números de 4 bits cada
      - 1 bit de carry-in.
    - Gera:
      - Resultado da soma, com 4 bits
      - 1 bit de carry-out.

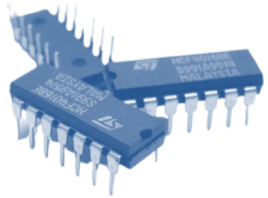


TÉCNICO LISBOA



Representação de números com sinal

Representação de números com sinal



- Representação de números negativos

  - **Módulo + Sinal**
    - O bit mais significativo representa o sinal, e os restantes bits representam o seu valor absoluto.

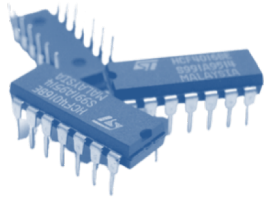
      Ex.: -9 = 10001001

    - **O valor zero tem duas representações…**

Decimal	Módulo + Sinal
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111
-8	-

?

Representação de números com sinal



- Representação de números negativos

  - **Complemento para 1**
    - O complemento para 1 de N, em n bits, é definido como $(2^n - 1) - N$.
    - $2^n - 1$ é um número constituído por n 1's.
    - Subtrair de 1 equivale a inverter o bit:

      $1 - 0 = 1$ e $1 - 1 = 0$.
    - Portanto, complementar para 1 corresponde a inverter todos os bits ($0 \rightarrow 1$ e $1 \rightarrow 0$).

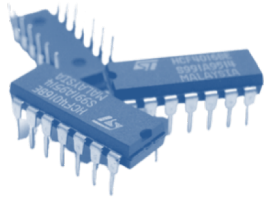
      Ex.: -9 = 11110110

      (= 11111111 – 00001001 = $255_{10} - 9_{10}$).

    - **O valor zero tem duas representações…**

Decimal	Complemento para 1	
+7	0111	
+6	0110	
+5	0101	
+4	0100	
+3	0011	
+2	0010	
+1	0001	
+0	0000	?
-0	1111	
-1	1110	
-2	1101	
-3	1100	
-4	1011	
-5	1010	
-6	1001	
-7	1000	
-8	-	

Representação de números com sinal



- Representação de números negativos

  - **Complemento para 2**
    - O complemento para 2 de N, em n bits, é definido como $2^n - N$ para $N \neq 0$, e 0 para $N = 0$.
    - Portanto, complementar para 2 corresponde a complementar para 1 e somar 1.

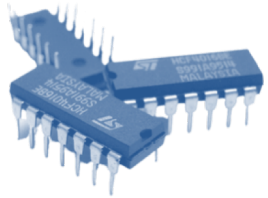
      Ex.: -9 = 11110111

      (= 100000000 – 00001001 = $256_{10} - 9_{10}$).

    - Na prática, o complemento para 2 pode ser formado do seguinte modo: mantém-se todos os 0's menos significativos e o primeiro 1, e invertem-se todos os outros bits mais significativos.
    - **Uma única representação para o valor zero.**

Decimal	Complemento para 2	
+7	0111	
+6	0110	
+5	0101	
+4	0100	
+3	0011	
+2	0010	
+1	0001	
+0	0000	?
-0	-	
-1	1111	
-2	1110	
-3	1101	
-4	1100	
-5	1011	
-6	1010	
-7	1001	
-8	1000	

Representação de números com sinal

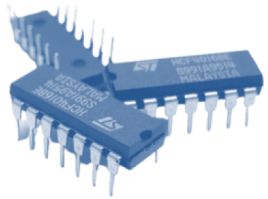


- Números binários com sinal
  - As operações usando o sistema de sinal e valor são mais complicadas, devido à necessidade de gerir separadamente o sinal e o valor.
  - Por isso, são normalmente utilizadas representações em complemento.
  - A representação em complemento para 2 é habitualmente preferida em sistemas digitais por ter uma única representação para o valor zero, e por as operações envolvidas serem mais simples.

Decimal	Complemento para 2	Complemento para 1	Módulo + Sinal
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	-	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	-	-

?

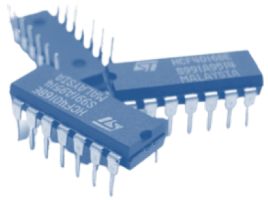
Representação de números com sinal



- Extensão do sinal (complemento para 2)
  - Representação de um número utilizando um determinado número de bits, através da adição/remoção de bits à esquerda iguais ao bit de sinal

Exemplos:

Representação de números com sinal



- Extensão do sinal (complemento para 2)
  - Representação de um número utilizando um determinado número de bits, através da adição/remoção de bits à esquerda iguais ao bit de sinal

Exemplos:

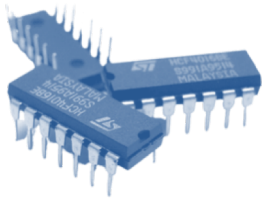
`0100` = +4 (4 bits) → `00000100` = +4 (8 bits)

`1011` = - 5 (4 bits) → `11111011` = - 5 (8 bits)

`0010` = +2 (4 bits) → `010` = +2 (3 bits)

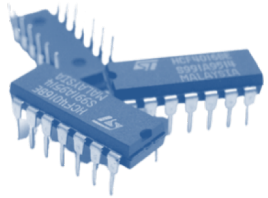
`1010` = - 6 (4 bits) → `???` = - 6 (3 bits)

TÉCNICO LISBOA



Operações com números com sinal

Representação de números com sinal



- Soma aritmética de números com sinal usando complemento para 2
  - A soma aritmética de dois números binários com sinal, representados em complemento para 2, é obtida pela simples adição dos dois números incluindo os bits de sinal.
    O último "carry out" não é considerado.

Exemplos:

	0 0 0 0	
4	0 1 0 0	
+ 3	+ 0 0 1 1	
7	0 1 1 1	

	1 1 0 0	
4	0 1 0 0	
+ (– 3)	+ 1 1 0 1	
1	0 0 0 1	

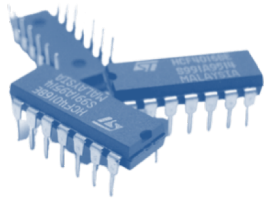
	0 0 0 0	
– 4	1 1 0 0	
+ 3	+ 0 0 1 1	
– 1	1 1 1 1	

	1 1 0 0	
– 4	1 1 0 0	
+ (– 3)	+ 1 1 0 1	
– 7	1 0 0 1	

Decimal	Complemento para 2
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

TÉCNICO LISBOA

Subtractores



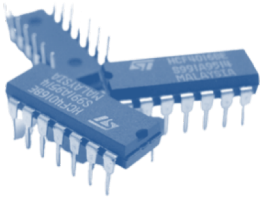
- Subtracção de números com sinal
  - A subtracção de 2 números binários com sinal, representados em complemento para 2, é realizada de forma idêntica ao que acontece na representação decimal:
  - Exemplo:

5	0 1 0 1	
– 2	– 0 0 1 0	
3	0 0 1 1	
	0 0 0 1 0	← *Borrow*

  - O bit de empréstimo (borrow) é um valor que vai ser retirado ao bit de peso seguinte.

Decimal	Complemento para 2
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Subtractores



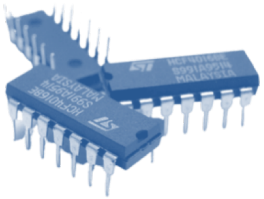
- Subtracção de números com sinal usando complemento para 2
  - A subtracção de dois números binários com sinal, representados em complemento para 2, é obtida do seguinte modo:
    - forma-se o complemento para 2 do subtractor
    - soma-se ao subtraendo.

Exemplo:

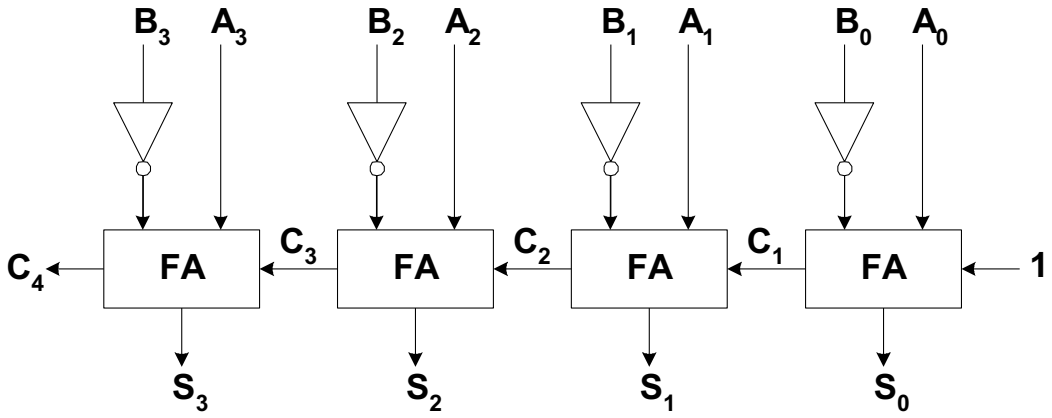
$$\begin{array}{rr} 4 & 0\,1\,0\,0 \\ -\,3 & -\,0\,0\,1\,1 \\ \hline 1 & \end{array} \quad \Rightarrow \quad \begin{array}{rr} 4 & 0\,1\,0\,0 \\ +\,(-3) & +\,1\,1\,0\,1 \\ \hline 1 & 0\,0\,0\,1 \end{array} \quad \Rightarrow \quad \begin{array}{r} 1 \\ 0\,1\,0\,0 \\ +\,1\,1\,0\,0 \\ \hline 0\,0\,0\,1 \end{array}$$

(através de complemento para 2) (através de complemento para 1)

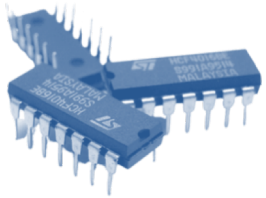
Subtractores



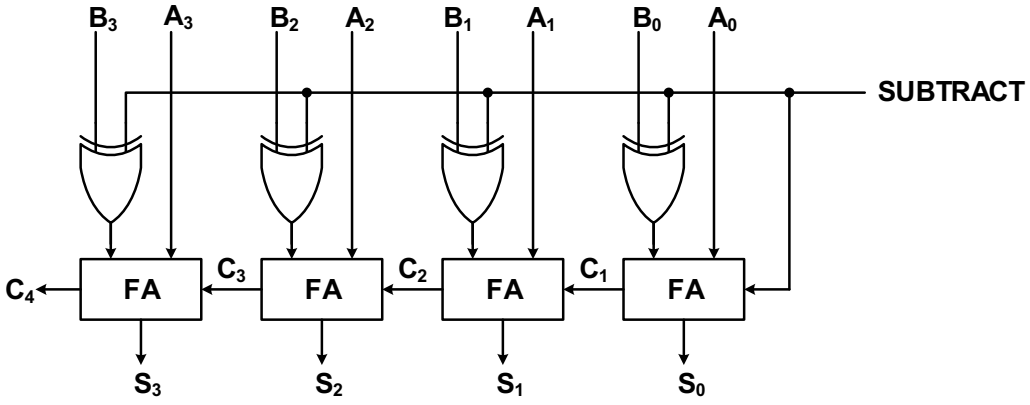
- Subtracção de números com sinal usando complemento para 2
  - Complemento para 2 = (Complemento para 1) + 1
    - A complementação para 1 é realizada invertendo todos os bits do subtractor.
    - A adição de 1 é efectuada pondo o Carry inicial a 1.



Circuito somador/subtractor

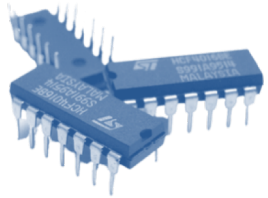


- Circuito somador/subtractor
  - As operações de adição e subtracção são habitualmente combinadas num único somador genérico, através da inclusão de 1 porta ou-exclusivo em cada Full-Adder.
    - Quando o sinal de controlo SUBTRACT = 0, é realizada a adição A + B (os operandos B_i não são invertidos e $C_0 = 0$).
    - Quando o sinal de controlo SUBTRACT = 1, é realizada a subtracção A – B (os operandos B_i são invertidos e $C_0 = 1$).



SUBTRACT	Bi	Yi = SUBTRACT ⊕ Bi	
0	0	0	Yi = Bi
0	1	1	
1	0	1	Yi = Bi
1	1	0	

Excesso

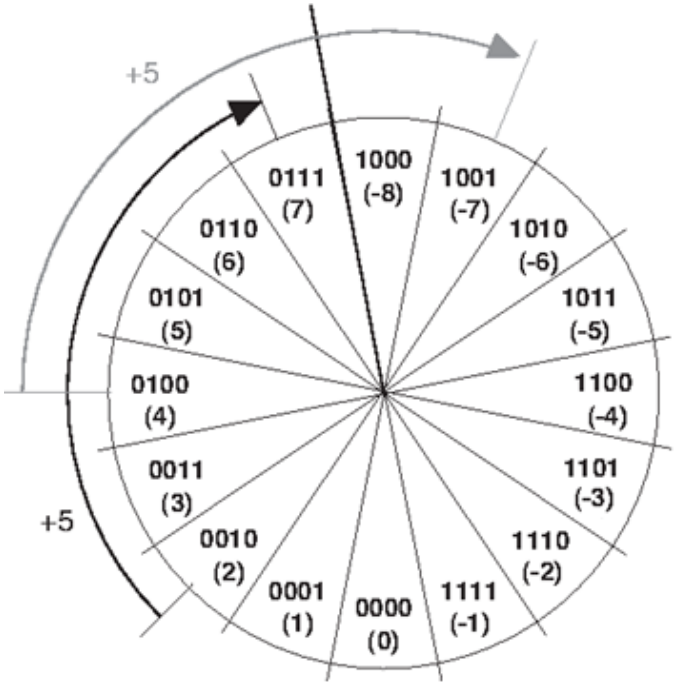


- Excesso (overflow)

		0	0	0	0
2		0	0	1	0
+ 5	+	0	1	0	1
7		0	1	1	1

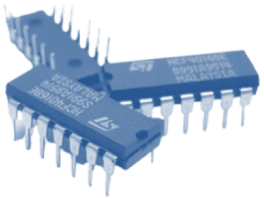
		0	1	0	0
4		0	1	0	0
+ 5	+	0	1	0	1
???		1	0	0	1

Decimal	Complemento para 2
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000



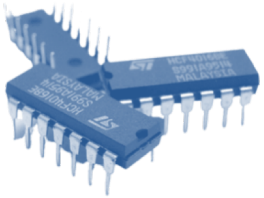
TÉCNICO LISBOA

Excesso



- Excesso (overflow)
  - Para se obter um resultado correcto, na adição e na subtracção, é necessário assegurar que o resultado tem um número de bits suficiente. Se somarmos dois números de N bits e o resultado ocupar N+1 bits diz-se que ocorreu um **overflow**.
  - As unidades aritméticas digitais usam um número fixo de bits para armazenar os operandos e os resultados, sendo necessário detectar e sinalizar a ocorrência de um **overflow**.
    - Exemplo: um **overflow** pode ocorrer na adição se os dois operandos são ambos positivos ou se são ambos negativos.

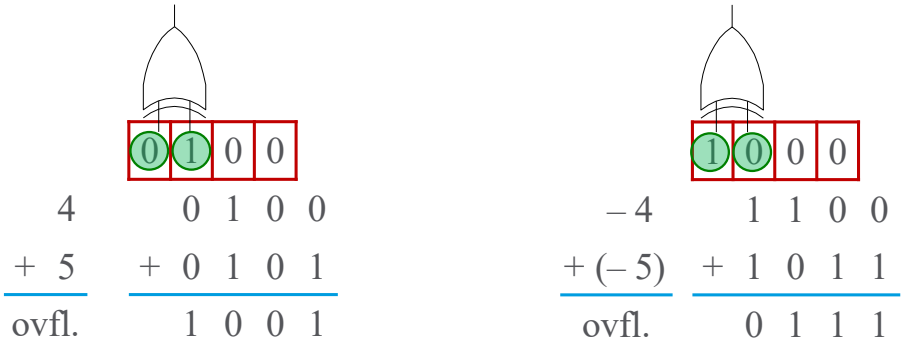
Excesso



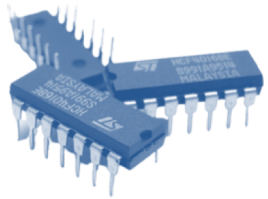
- Excesso (overflow)
  - A condição de overflow pode ser detectada por inspecção dos dois bits de carry mais significativos.

Exemplo:

$$Overflow = CarryOut_{N-1} \oplus CarryOut_{N-2}$$



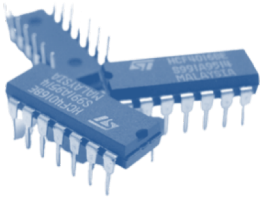
Excesso



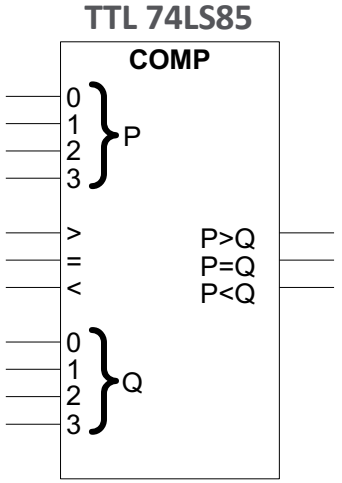
- Qual a diferença entre os sinais de *carry* e *overflow*?

Representação	$C = C_{N-1} = 1$	$O = 1$
SEM sinal	Excedeu a capacidade de representação	Sem significado
COM sinal	Sem significado	Excedeu a capacidade de representação

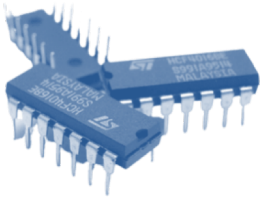
Circuito Comparador



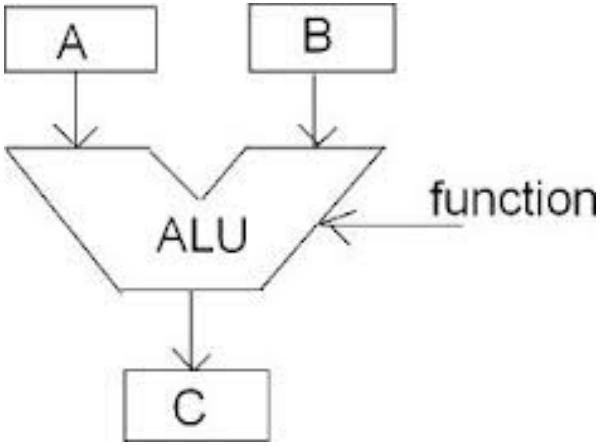
- Comparador de números de 4 bits
  - Este circuito faz a comparação de 2 números binários de 4 bits.
  - A comparação é realizada através de uma operação de subtracção e análise do resultado.
  - O circuito pode ser ligado em cascata, para realizar comparações entre números de N > 4 bits, utilizando os 3 bits de entrada suplementares.



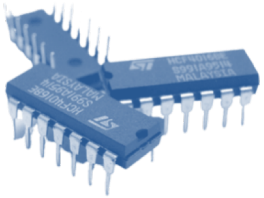
TÉCNICO LISBOA



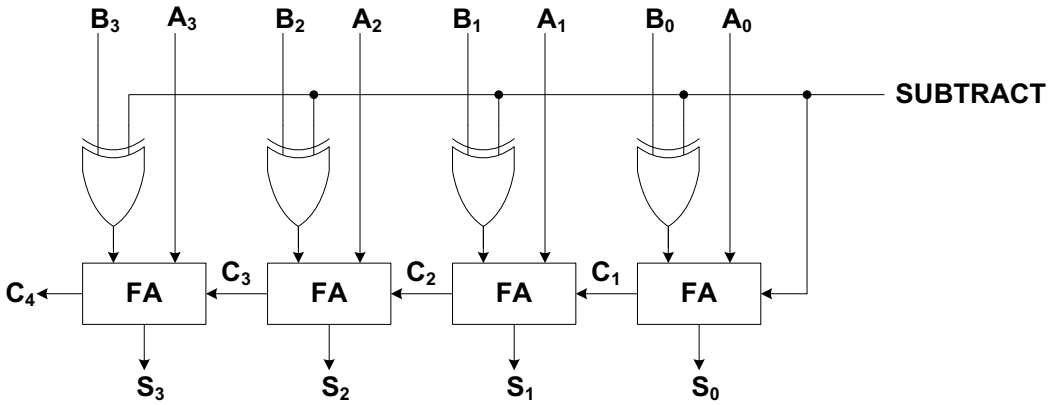
Unidade Lógica e Aritmética



Circuito somador/subtractor



- Circuito somador/subtractor (revisão)

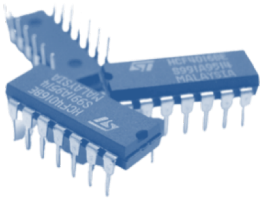


SUBTRACT = 0 ➞ S = A + B

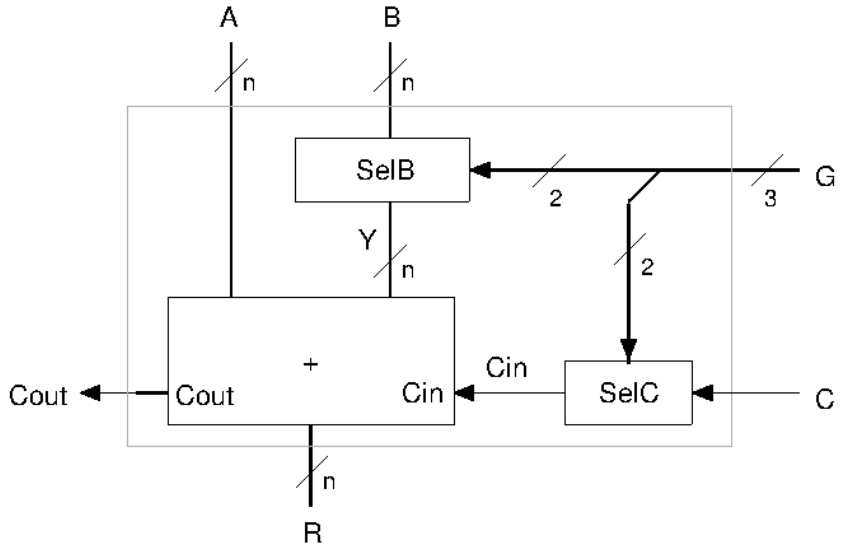
SUBTRACT = 1 ➞ S = A – B

- Será possível realizar unidades aritméticas mais completas com um único circuito combinatório?

Unidade Aritmética

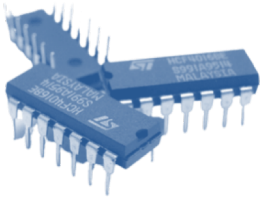


- Unidade aritmética baseada num único circuito:

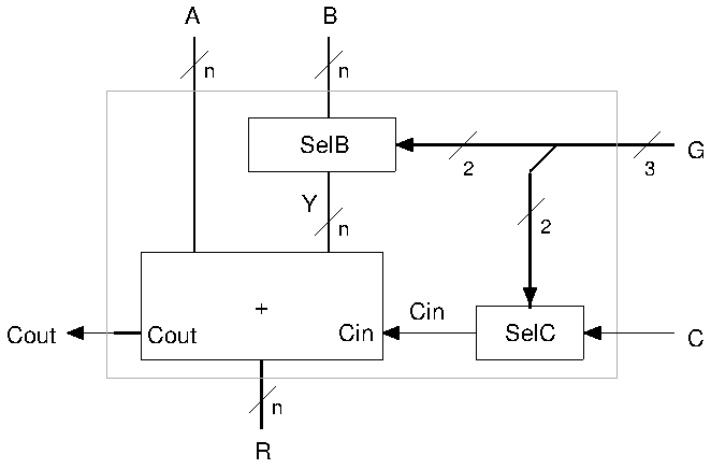


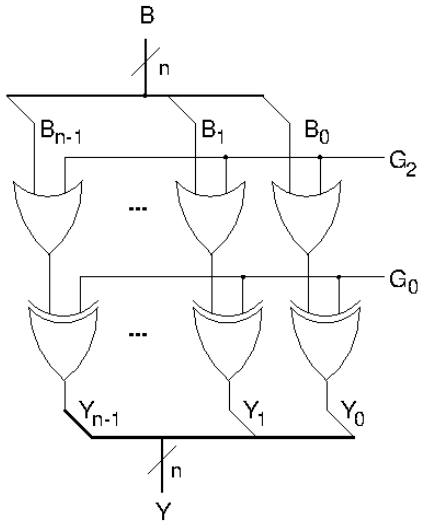
- A entrada **G** define o **tipo de operação**.

Unidade Aritmética



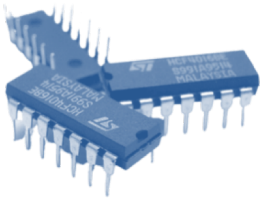
- Unidade aritmética baseada num único circuito:



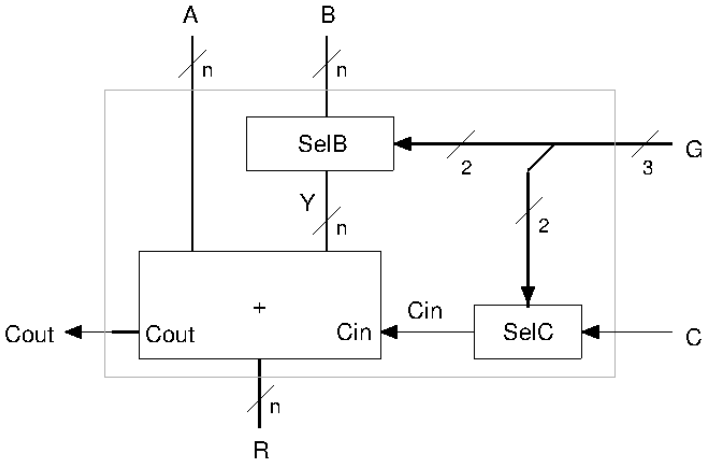


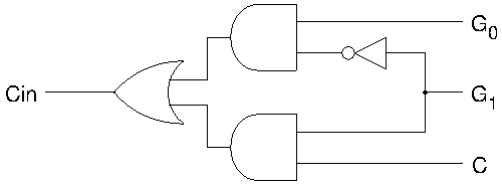
G_2G_0	Y_i
00	B_i
01	$\overline{B_i}$
10	1
11	0

Unidade Aritmética



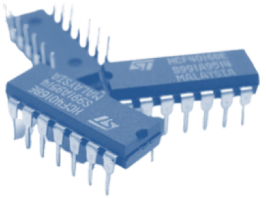
- Unidade aritmética baseada num único circuito:



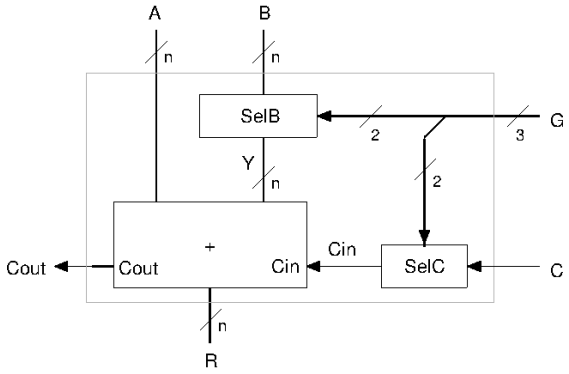


G_1G_0	C_{in}
00	0
01	1
10	C
11	C

Unidade Aritmética



- Unidade aritmética baseada num único circuito:

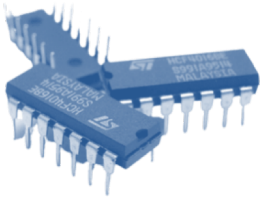


$G_2G_1G_0$	Y_i	C_{in}	Operação
000	B_i	0	

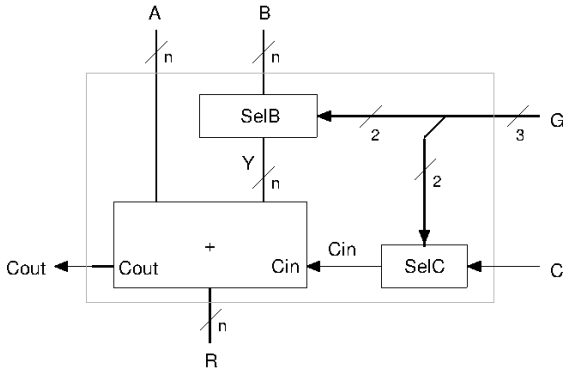
G_2G_0	Y_i
00	B_i
01	$\overline{B_i}$
10	1
11	0

G_1G_0	C_{in}
00	0
01	1
10	C
11	C

Unidade Aritmética



- Unidade aritmética baseada num único circuito:



G_2G_0	Y_i
00	B_i
01	$\overline{B_i}$
10	1
11	0

G_1G_0	C_{in}
00	0
01	1
10	C
11	C

$G_2G_1G_0$	Y_i	C_{in}		Operação
000	B_i	0	R ← A + B	Soma
001	$\overline{B_i}$	1	R ← A – B	Subtracção
010	B_i	C	R ← A + B + C	Soma com bit de transporte
011	$\overline{B_i}$	C	R ← A – B – $\overline{C}$	Subtracção com transporte negado
100	1	0	R ← A – 1	Decremento
101	0	1	R ← A + 1	Incremento
110	1	C	R ← A – $\overline{C}$	Decremento, se C=0
111	0	C	R ← A + C	Incremento, se C=1

$A+\overline{B}+C_{in}$

$= A+\overline{B}+C_{in}+1-1$

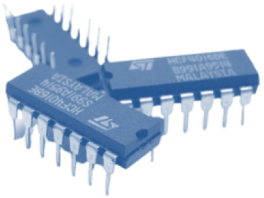
$= \underbrace{(A+\overline{B}+1)}_{A-B}+C_{in}-1$

$= (A-B) - (1-C_{in})$

$= (A-B) - \overline{C_{in}}$

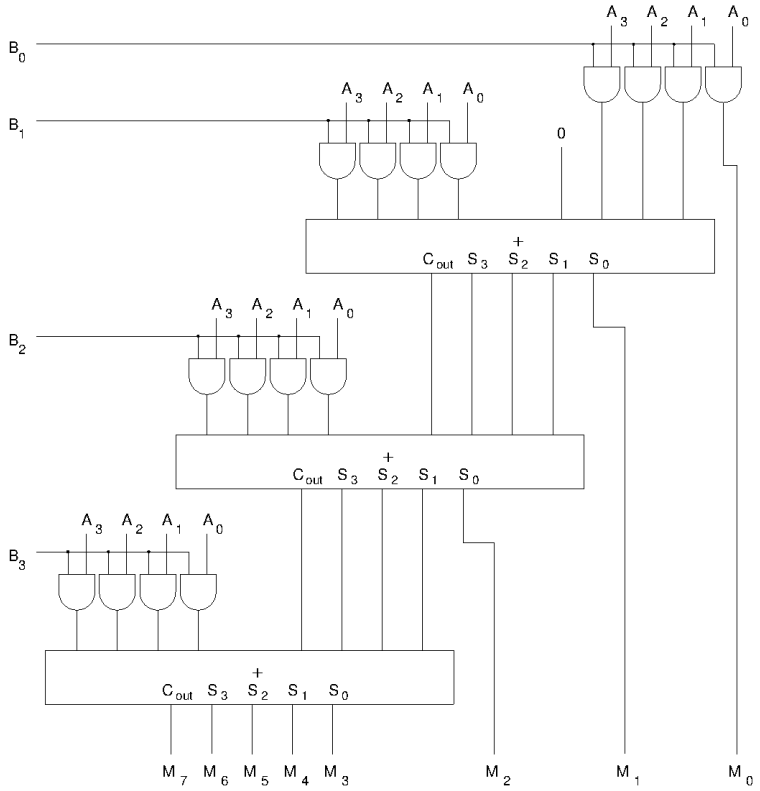
Borrow

Outras Operações



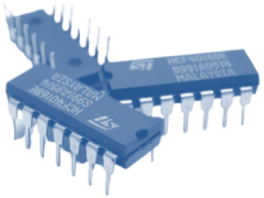
- Multiplicação (representação sem sinal)

A					1	1	0	1	Multiplicando
B			x		1	0	1	0	Multiplicador
					0	0	0	0	Produto parcial
				1	1	0	1		Produto parcial
			0	0	0	0			Produto parcial
		1	1	0	1				Produto parcial
M	1	0	0	0	0	0	1	0	Resultado



TÉCNICO LISBOA

Outras Operações



- Caso particular: multiplicação por uma potência inteira de 2

Exemplo:

$6 \times 4 = 24$

$\Leftrightarrow (\underline{0} \times 2^3 + \underline{1} \times 2^2 + \underline{1} \times 2^1 + \underline{0} \times 2^0) \times 4 = 24$

$\Leftrightarrow (\underline{0} \times 2^3 + \underline{1} \times 2^2 + \underline{1} \times 2^1 + \underline{0} \times 2^0) \times 2^2 = 24$

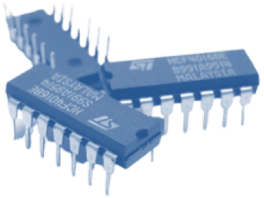
$\Leftrightarrow \underline{0} \times 2^5 + \underline{1} \times 2^4 + \underline{1} \times 2^3 + \underline{0} \times 2^2 + \underline{0} \times 2^1 + \underline{0} \times 2^0 = 24$

Ou seja: $000110 \times 4 = 011000$

Deslocamento à esquerda de 2 posições

A multiplicação pela k potência de 2 (i.e. 2^k) corresponde a deslocar os bits do operando em k posições para a esquerda.

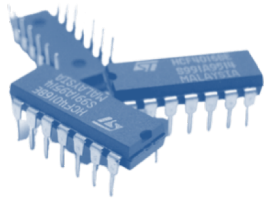
Outras Operações



- Divisão

	Dividendo								Divisor				
	1	0	0	1	0	0	1	1	0	1	0	1	
-	0	1	0	1					0	0	0	1	1	1	0	1
	0	1	0	0	0				Quociente				
	-	0	1	0	1								
		0	0	1	1	0							
		-	0	1	0	1							
			0	0	0	1	1	1					
				-	0	1	0	1					
					0	0	1	0	Resto				

Outras Operações



- Divisão
  - Não tem uma sequência fixa de operações elementares
  - O número seleccionado de bits do dividendo em cada passo é variável

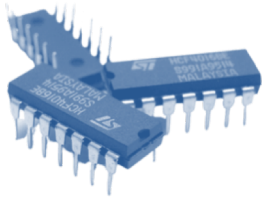
    +
  - Operação pouco frequente, na maioria das aplicações
  - Operação complexa

    ↓
  - Implementada tipicamente através de uma sequência de operações mais simples

    ↓

    Programa

Outras Operações



- Caso particular: divisão por uma potência inteira de 2

Exemplo:

$36 \div 4 = 9$

$\Leftrightarrow (\underline{1} \times 2^5 + \underline{0} \times 2^4 + \underline{0} \times 2^3 + \underline{1} \times 2^2 + \underline{0} \times 2^1 + \underline{0} \times 2^0) \div 4 = 9$

$\Leftrightarrow (\underline{1} \times 2^5 + \underline{0} \times 2^4 + \underline{0} \times 2^3 + \underline{1} \times 2^2 + \underline{0} \times 2^1 + \underline{0} \times 2^0) \div 2^2 = 9$

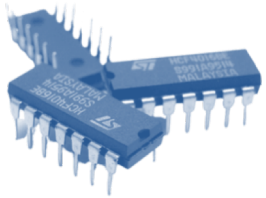
$\Leftrightarrow (\underline{1} \times 2^3 + \underline{0} \times 2^2 + \underline{0} \times 2^1 + \underline{1} \times 2^0) = 9$

Ou seja: $100100 \div 4 = 1001$ ~~00~~

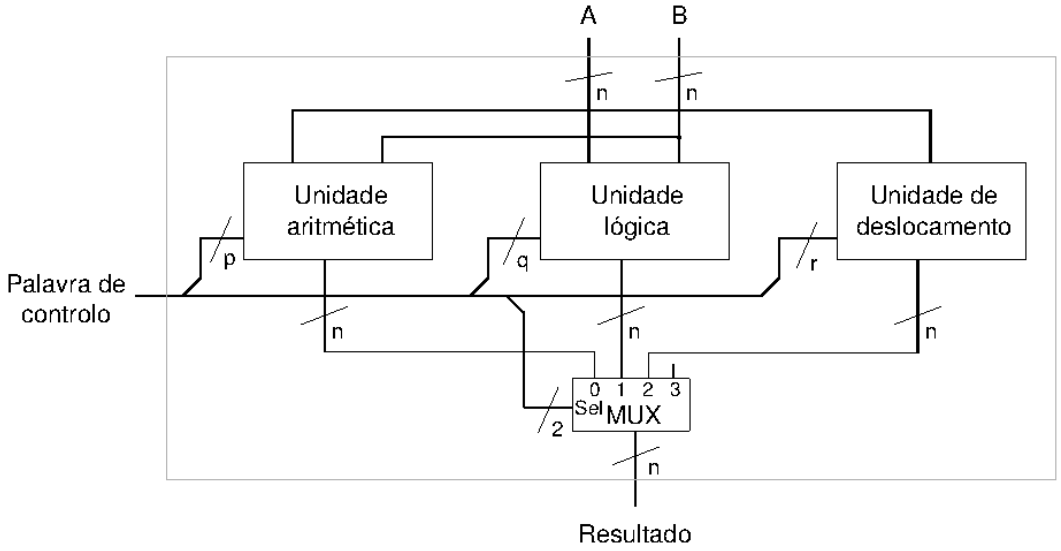
Deslocamento à direita de 2 posições

A divisão pela k potência de 2 (i.e. 2^k) corresponde a deslocar os bits do operando em k posições para a direita.

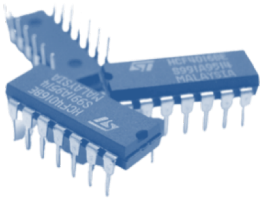
Unidade Lógica e Aritmética



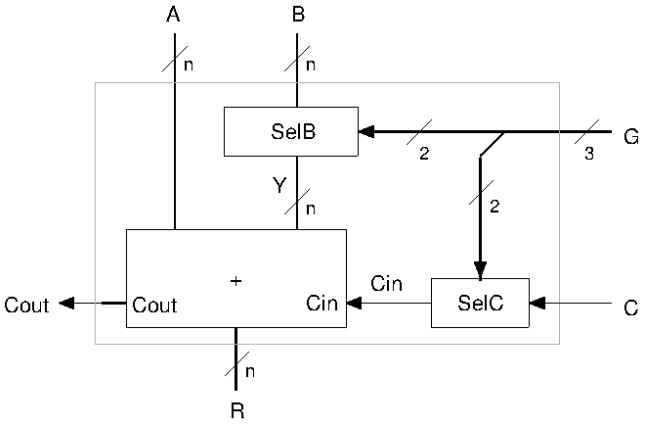
- Unidade Lógica e Aritmética
  - Circuito combinatório que implementa as operações:
    - **Aritméticas**
    - **Lógicas**
    - **Deslocamento**



Unidade Lógica e Aritmética

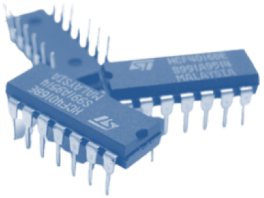


- Unidade Aritmética

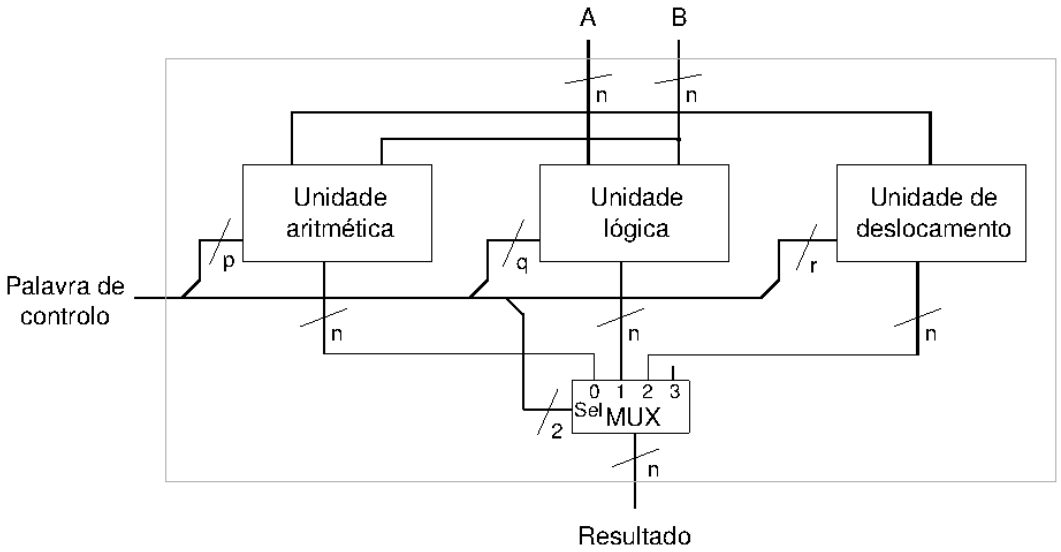


$G_2G_1G_0$	Y_i	C_{in}		Operação
000	B_i	0	R ← A + B	Soma
001	$\overline{B_i}$	1	R ← A – B	Subtracção
010	B_i	C	R ← A + B + C	Soma com bit de transporte
011	$\overline{B_i}$	C	R ← A – B – $\overline{C}$	Subtracção com transporte negado
100	1	0	R ← A – 1	Decremento
101	0	1	R ← A + 1	Incremento
110	1	C	R ← A – $\overline{C}$	Decremento, se C=0
111	0	C	R ← A + C	Incremento, se C=1

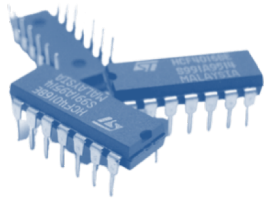
Unidade Lógica e Aritmética



- Unidade Lógica e Aritmética
  - Circuito combinatório que implementa as operações:
    - **Aritméticas**
    - **Lógicas**
    - **Deslocamento**

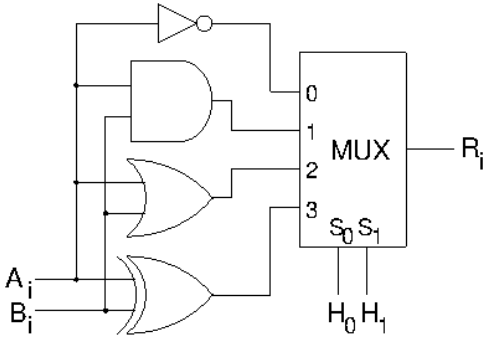


Unidade Lógica e Aritmética



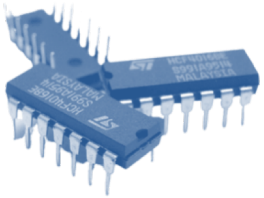
- Unidade Lógica
  - As operações lógicas aplicam-se individualmente a cada bit dos operandos de entrada:

$$R \leftarrow A \wedge B \quad \Leftrightarrow \quad R \leftarrow A_{n-1} \wedge B_{n-1} \mid A_{n-2} \wedge B_{n-2} \mid \ldots \mid A_0 \wedge B_0$$

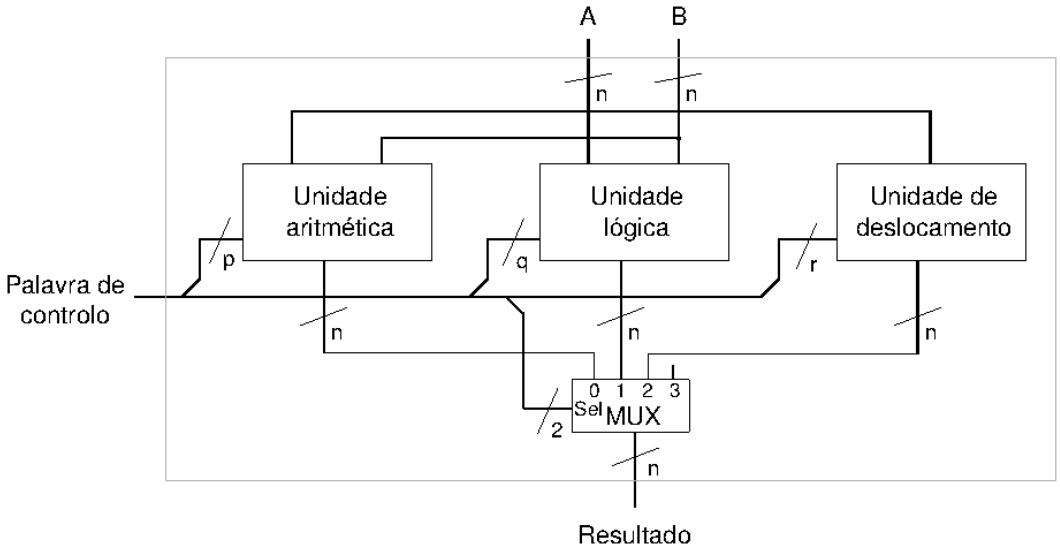


H_1H_0		Operação
00	$R \leftarrow \overline{A}$	Complemento
01	$R \leftarrow A \wedge B$	Conjunção
10	$R \leftarrow A \vee B$	Disjunção
11	$R \leftarrow A \oplus B$	Disjunção exclusiva

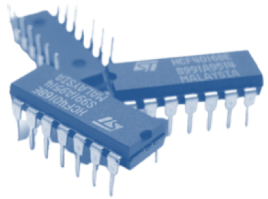
Unidade Lógica e Aritmética



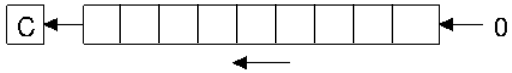
- Unidade Lógica e Aritmética
  - Circuito combinatório que implementa as operações:
    - **Aritméticas**
    - **Lógicas**
    - **Deslocamento**

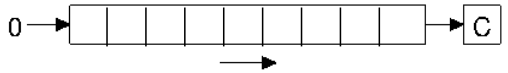


Unidade Lógica e Aritmética

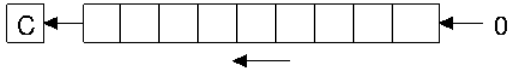


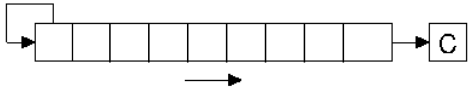
- Operações de deslocamento
  - Deslocamento simples



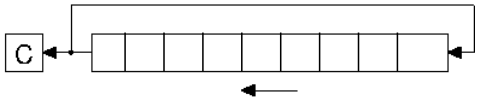


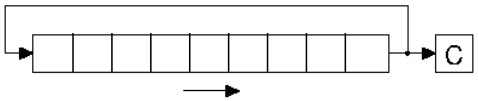
  - Deslocamento aritmético



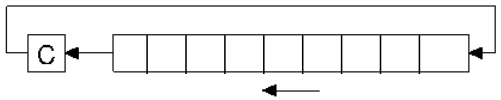


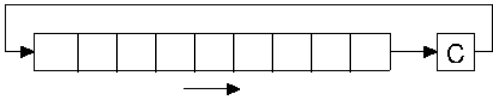
  - Rotação



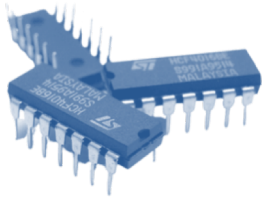


  - Rotação com transporte



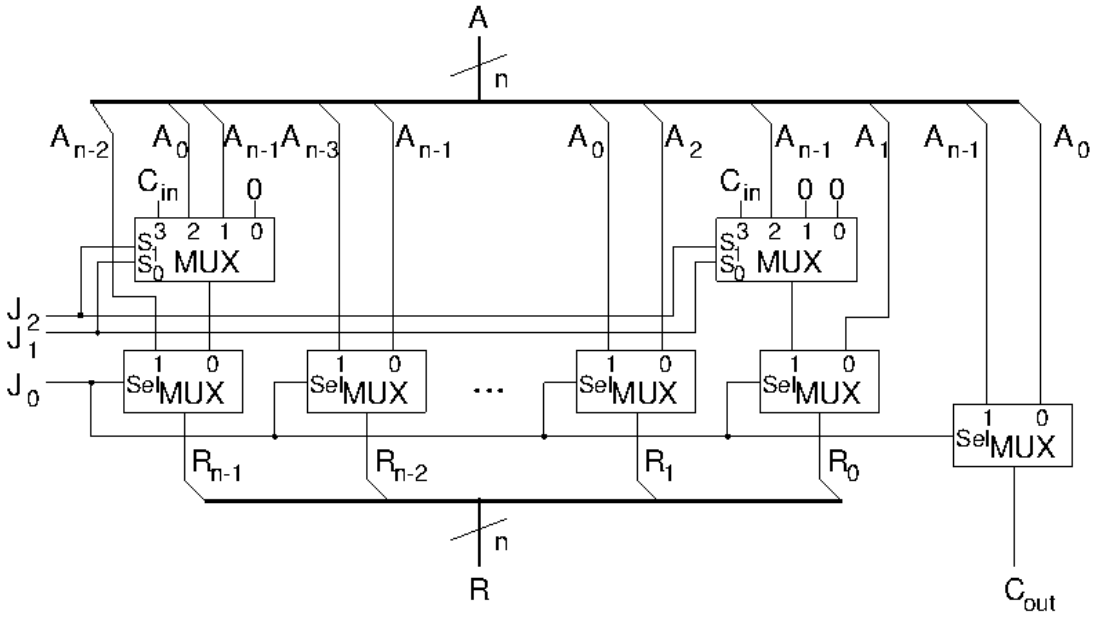


Unidade Lógica e Aritmética

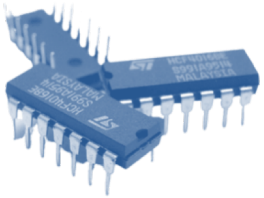


- Unidade de deslocamento

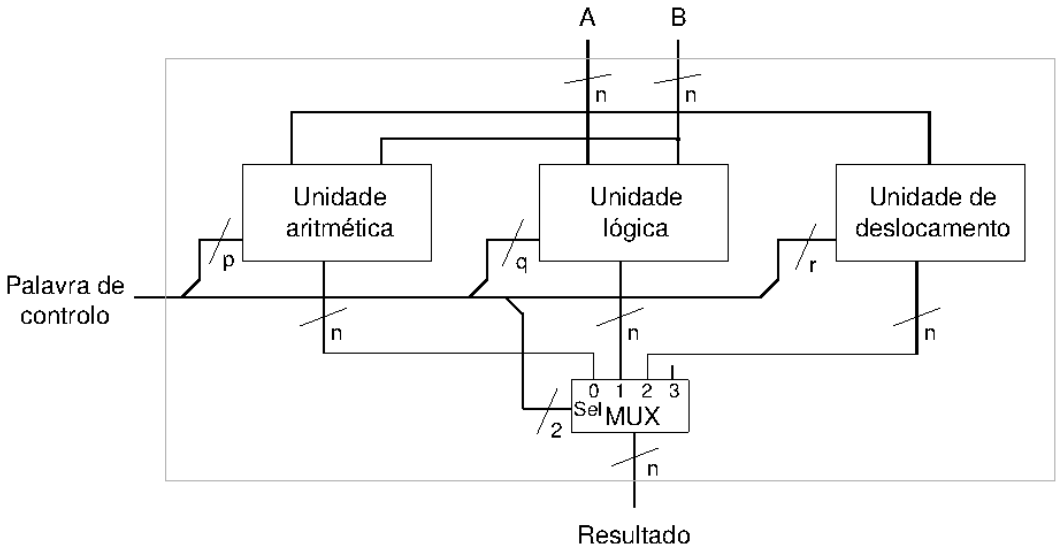
$J_2J_1J_0$		Operação
000	R ← SHR A	Deslocamento lógico à direita
001	R ← SHL A	Deslocamento lógico à esquerda
010	R ← SHRA A	Deslocamento aritmético à direita
011	R ← SHLA A	Deslocamento aritmético à esquerda
100	R ← ROR A	Rotação à direita
101	R ← ROL A	Rotação à esquerda
110	R ← RORC A	Rotação à direita com transporte
111	R ← ROLC A	Rotação à esquerda com transporte



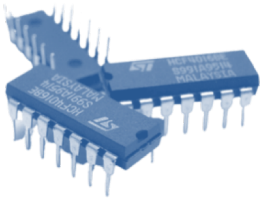
Unidade Lógica e Aritmética



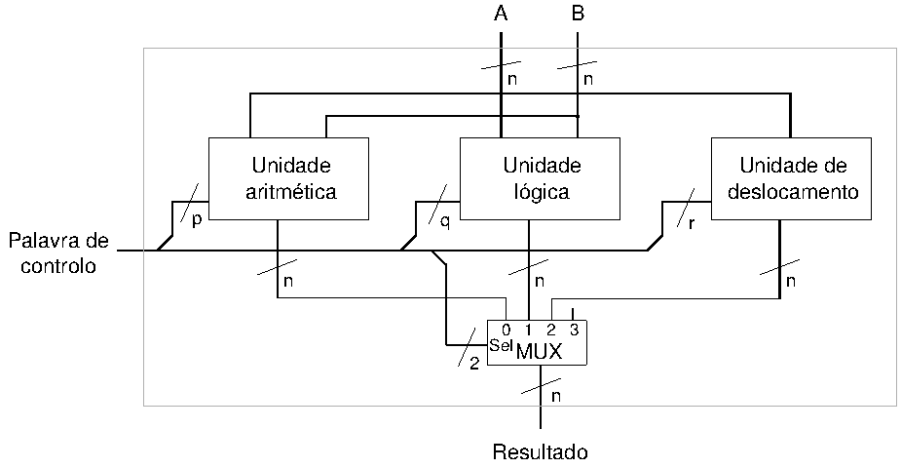
- Unidade Lógica e Aritmética
  - Circuito combinatório que implementa as operações:
    - **Aritméticas**
    - **Lógicas**
    - **Deslocamento**



Unidade Lógica e Aritmética



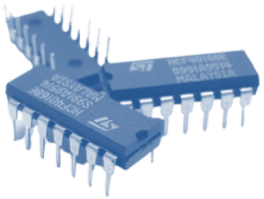
- Operações da unidade lógica e aritmética



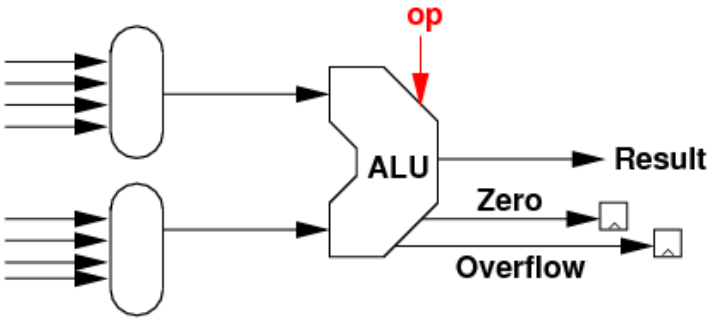
- A função realizada é definida por uma palavra de comando;
- Várias codificações possíveis.

$S_4S_3S_2S_1S_0$		Operação
00000	R ← A + B	Soma
00001	R ← A – B	Subtracção
00010	R ← A + B + C	Soma com bit de transporte
00011	R ← A – B – $\overline{C}$	Subtracção com transporte negado
00100	R ← A – 1	Decremento
00101	R ← A + 1	Incremento
00110	R ← A – $\overline{C}$	Decremento, se C=0
00111	R ← A + C	Incremento, se C=1
01-00	R ← $\overline{A}$	Complemento
01-01	R ← A ∧ B	Conjunção
01-10	R ← A ∨ B	Disjunção
01-11	R ← A ⊕ B	Disjunção exclusiva
10000	R ← SHR A	Deslocamento lógico à direita
10001	R ← SHL A	Deslocamento lógico à esquerda
10010	R ← SHRA A	Deslocamento aritmético à direita
10011	R ← SHLA A	Deslocamento aritmético à esquerda
10100	R ← ROR A	Rotação à direita
10101	R ← ROL A	Rotação à esquerda
10110	R ← RORC A	Rotação à direita com transporte
10111	R ← ROLC A	Rotação à esquerda com transporte
11---	R ← A	Transferência

Unidade Lógica e Aritmética



- Bits de Estado (*flags*)
  - Para além do resultado, as unidades lógicas e aritméticas disponibilizam também um conjunto de **bits de estado**, que reflectem o valor do resultado. Exemplos:
    - **Z** – activo quando o resultado corresponde ao valor zero;
    - **N** – activo quando o resultado corresponde a um valor negativo;
    - **P** – activo quando o resultado corresponde a um valor positivo;
    - **C** – activo quando o *carry* gerado aquando do cálculo do bit mais significativo do resultado é '1';
    - **O** – activo quando o resultado corresponde a uma situação de *overflow*.



TÉCNICO LISBOA