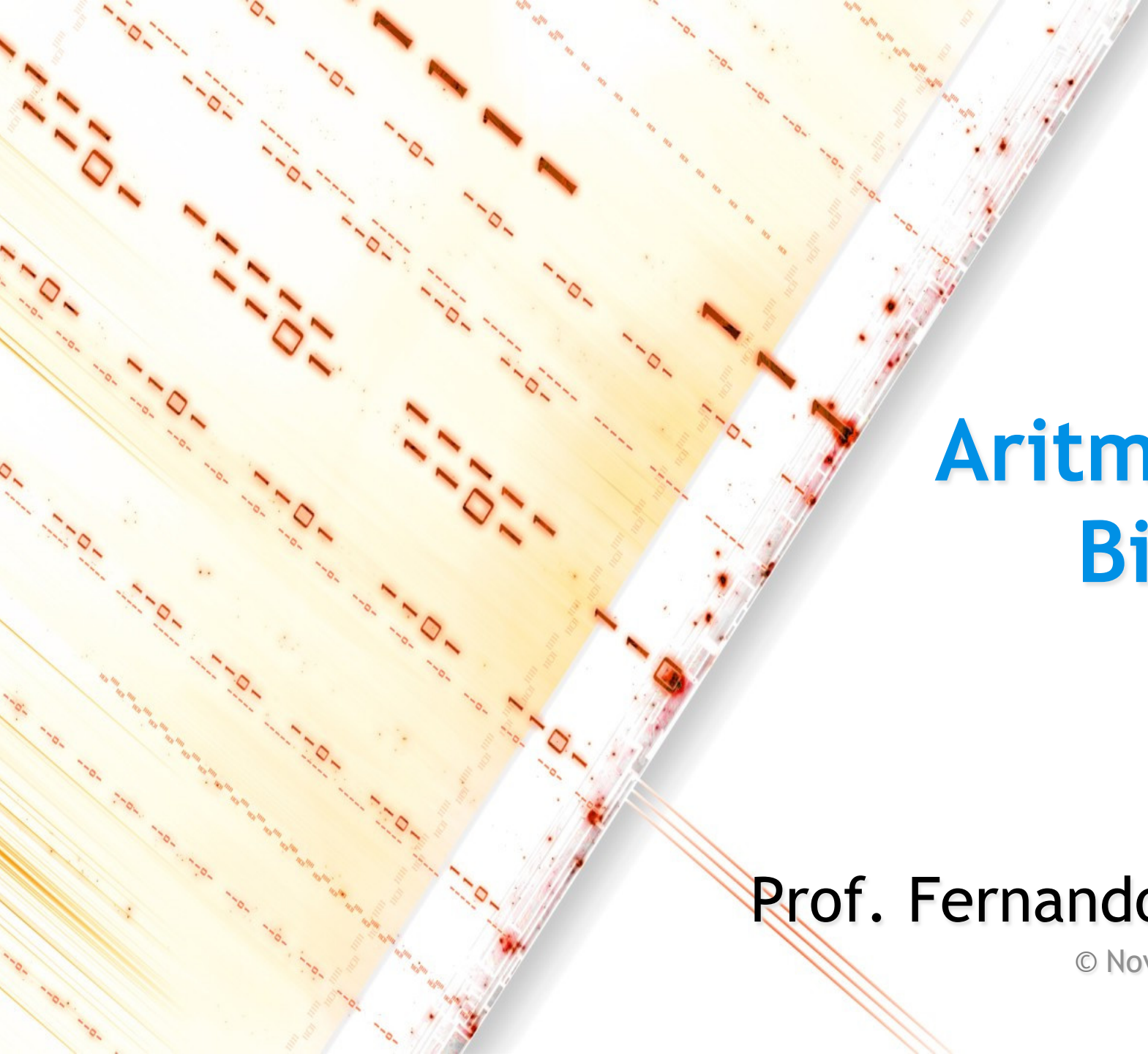


Aritmética Binária & ULAs

Prof. Fernando Passold

REVISÃO

Cascateamento de Somadores BCD:

CI Somador BCD:
- 74HCT583 (4 bits)

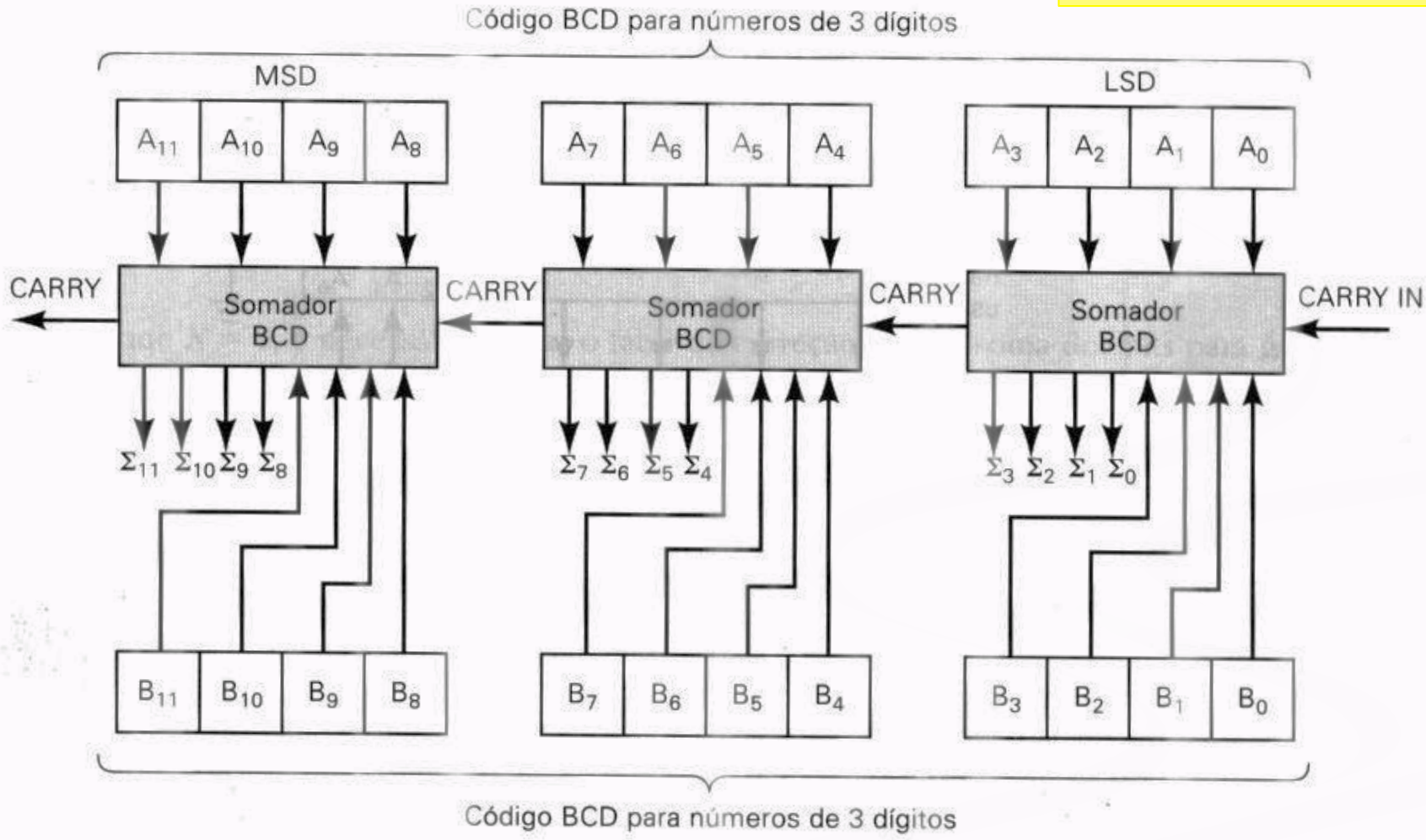
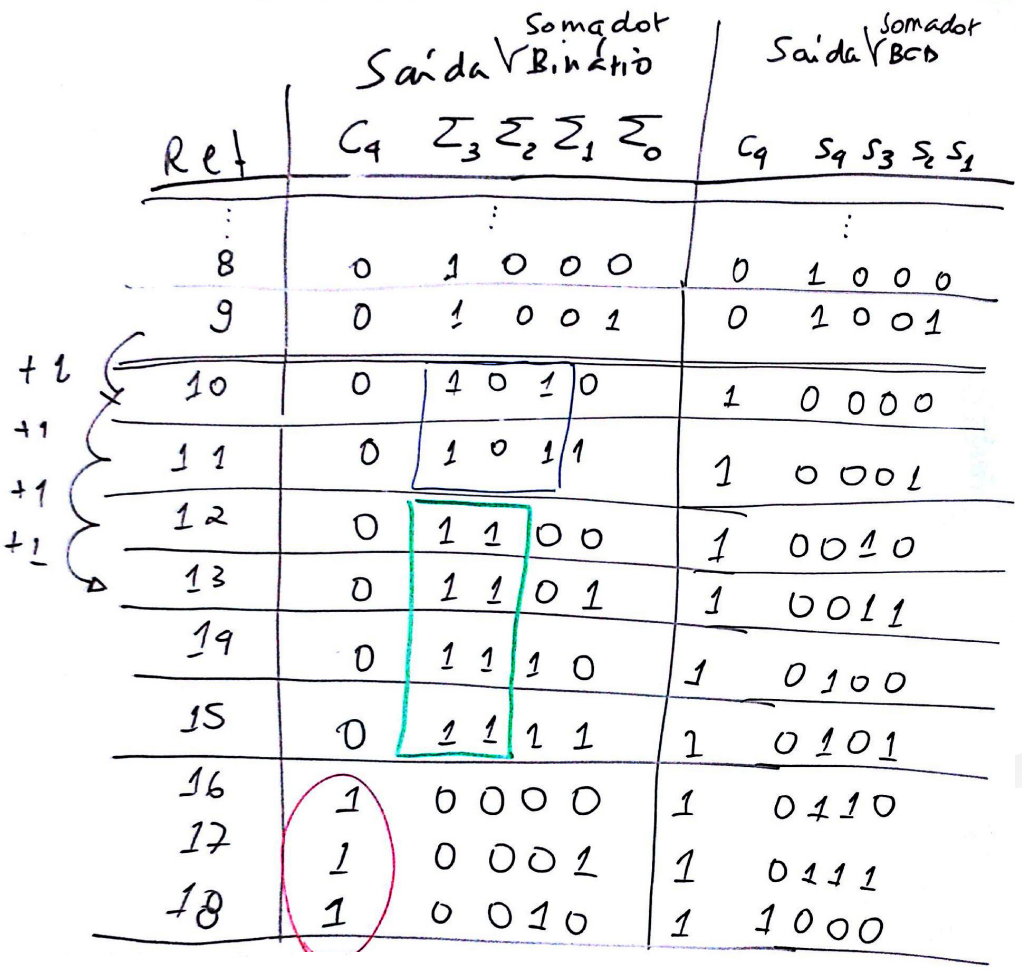


FIGURA 6.15 Conexão de somadores em cascata para somar dois números decimais de três dígitos.

Estrutura Interna Somador BCD:

Note: enquanto o resultado de uma soma não ultrapassa o número 9, o somador BCD atua exatamente igual a um somador binário.

O que acontece então quando o resultado é >= 10 ?



Detecta casos de resultados > 9

$$Y = C_4 + \Sigma_3 \cdot \Sigma_2 + \Sigma_3 \overline{\Sigma_2} \Sigma_1$$

$$Y = C_4 + \Sigma_3 \left(\Sigma_2 + \overline{\Sigma_2} \cdot \Sigma_1\right)$$

$x + \bar{x}y = x + y$

$$Y = C_4 + \Sigma_3 \left(\Sigma_2 + \Sigma_1\right)$$

Estrutura Interna Somador BCD:

Exemplo:

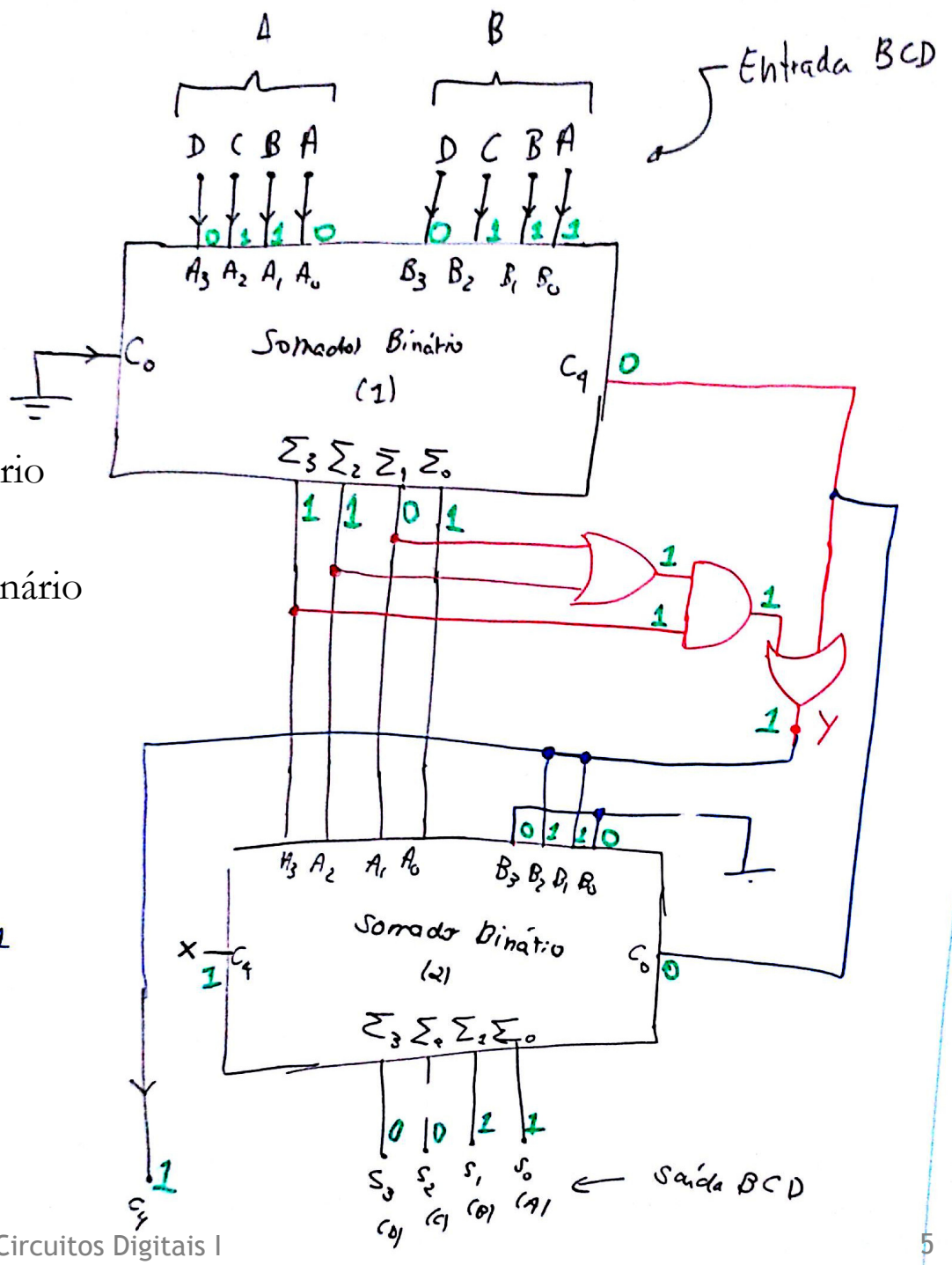
```
  6        0100
+ 7      + 0111      } 1o-Somador Binário
 13        1011
          + 0110      } 2o-Somador Binário
          1 0011
```

Para DEC/Display 7-Segmentos

$$Y = C_4 + \Sigma_3 \cdot \Sigma_2 + \overline{\Sigma_3}\,\overline{\Sigma_2}\,\Sigma_1$$

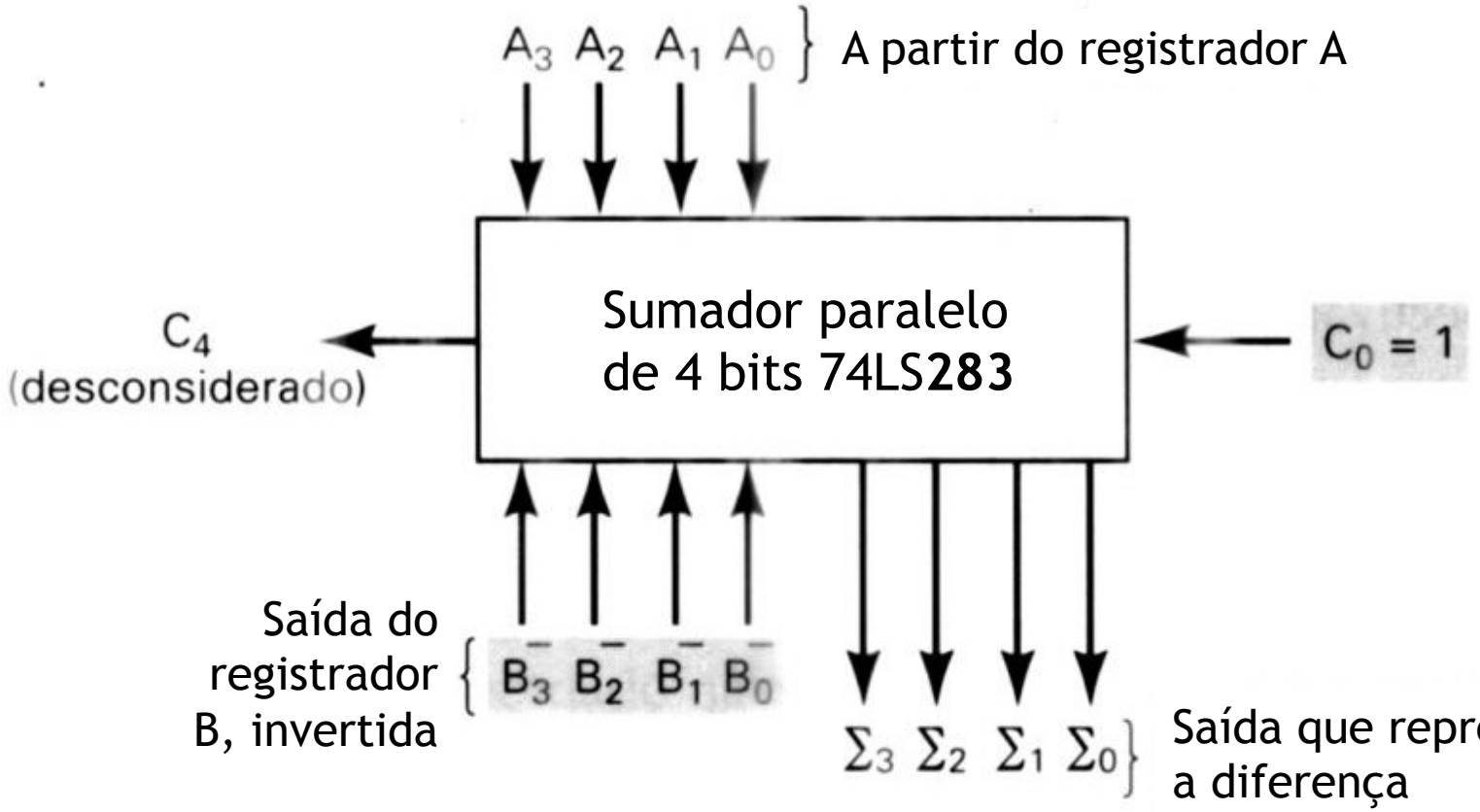
$$Y = C_4 + \overline{\Sigma_3}\,(\Sigma_2 + \overline{\Sigma_2} \cdot \Sigma_1)$$

$$Y = C_4 + \overline{\Sigma_3}\,(\Sigma_2 + \Sigma_1)$$

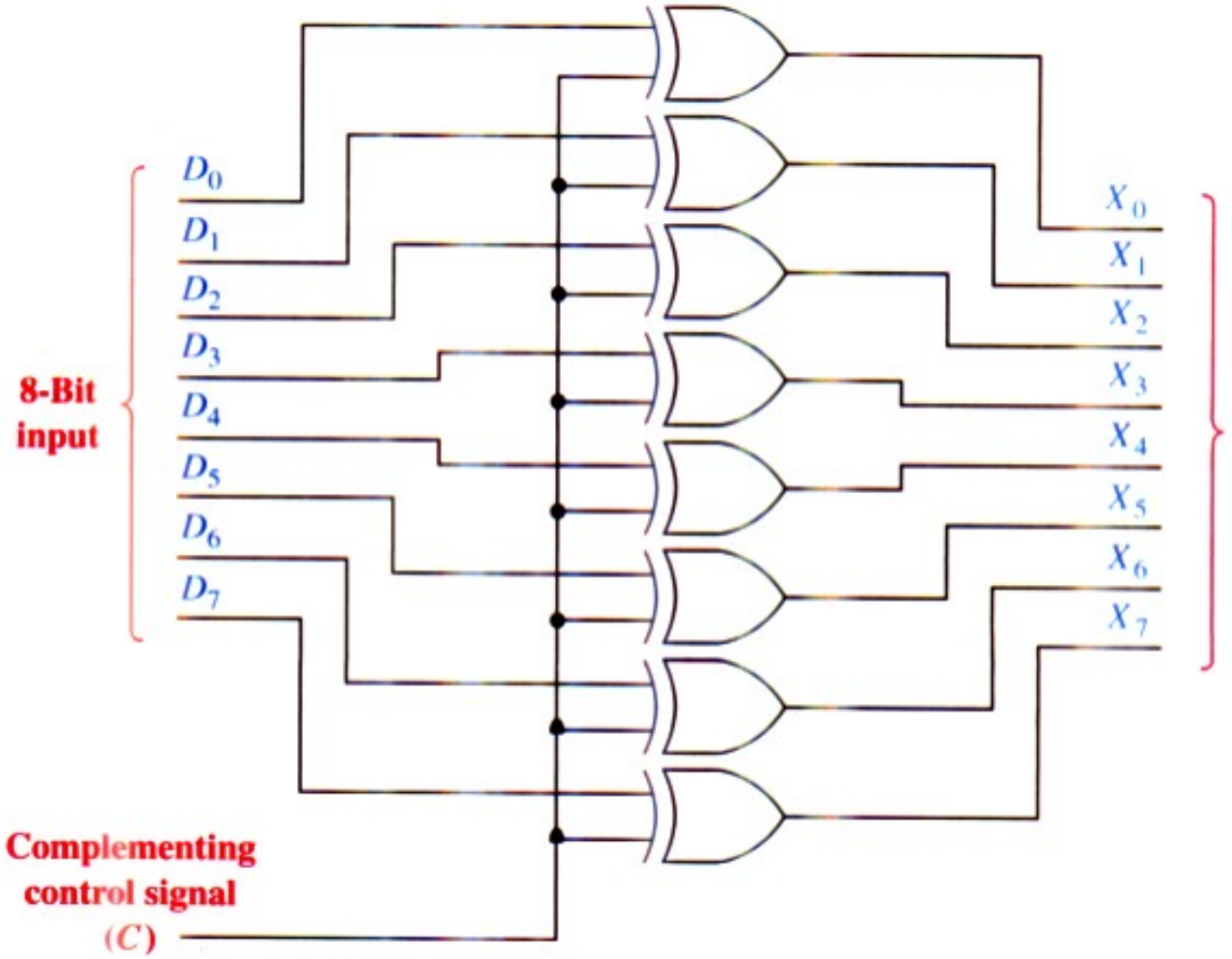


Recordando:
Subtração (C_2) usando CIs Somadores binários:

Neste caso: $\Sigma = A - B$



Uso do “Inversor Controlado”:



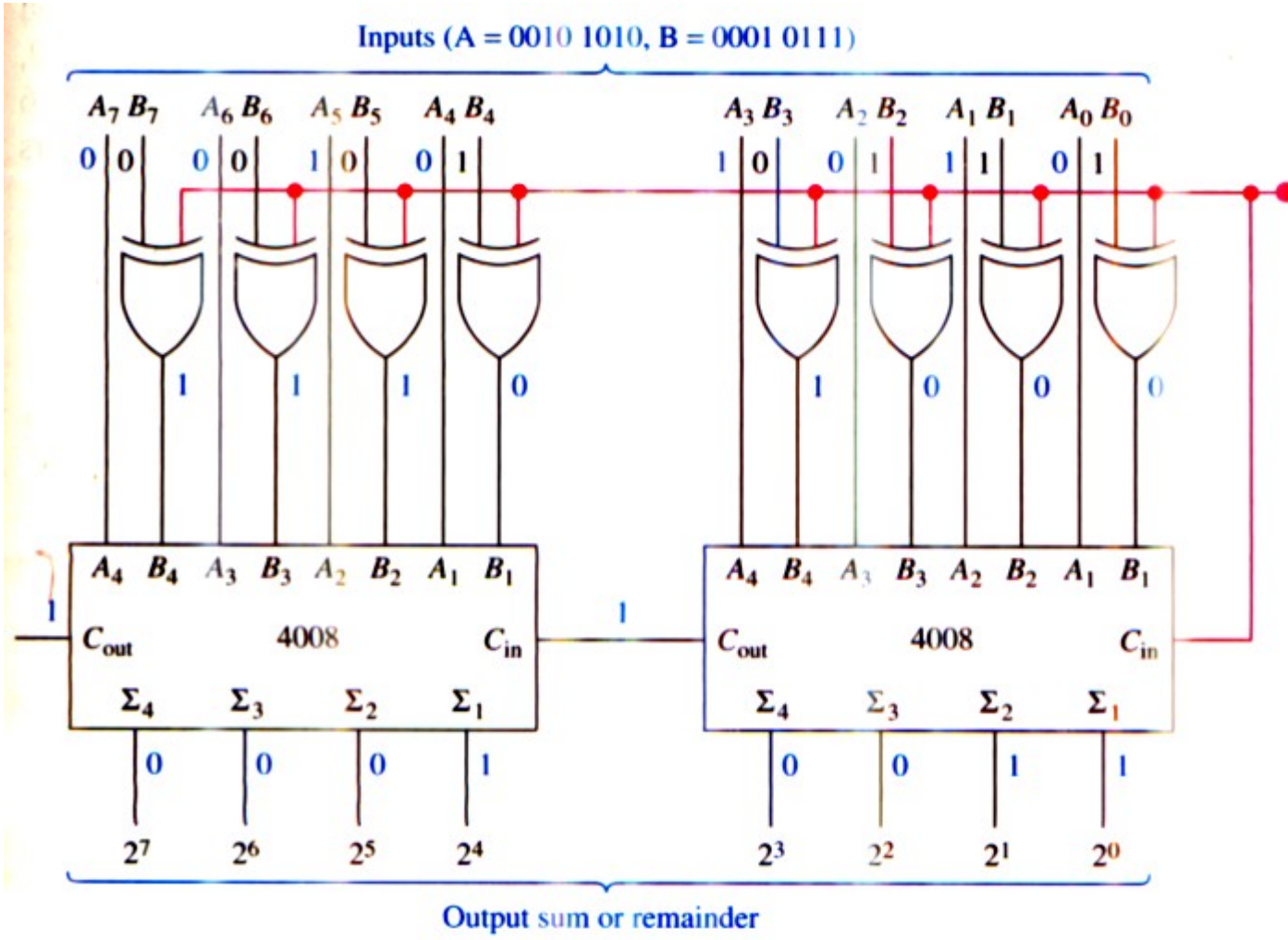
Saída Controlada:

$$Se: C = 1, X_{0-7} = \overline{D_{0-7}}$$
$$Se: C = 0, X_{0-7} = D_{0-7}$$

Fig.: Exemplo de solução típica empregada em circuitos aritméticos binários.

Exemplo usando Inversor Controlado:

- Circuito de substração: $\Sigma = A - B$



Carry-in

Neste exemplo:

Carry-out

```
          1 11  1     1
  42    0010 1010
- 23  + 1110 1000
 ----  -----------
  19    0001 0011
```

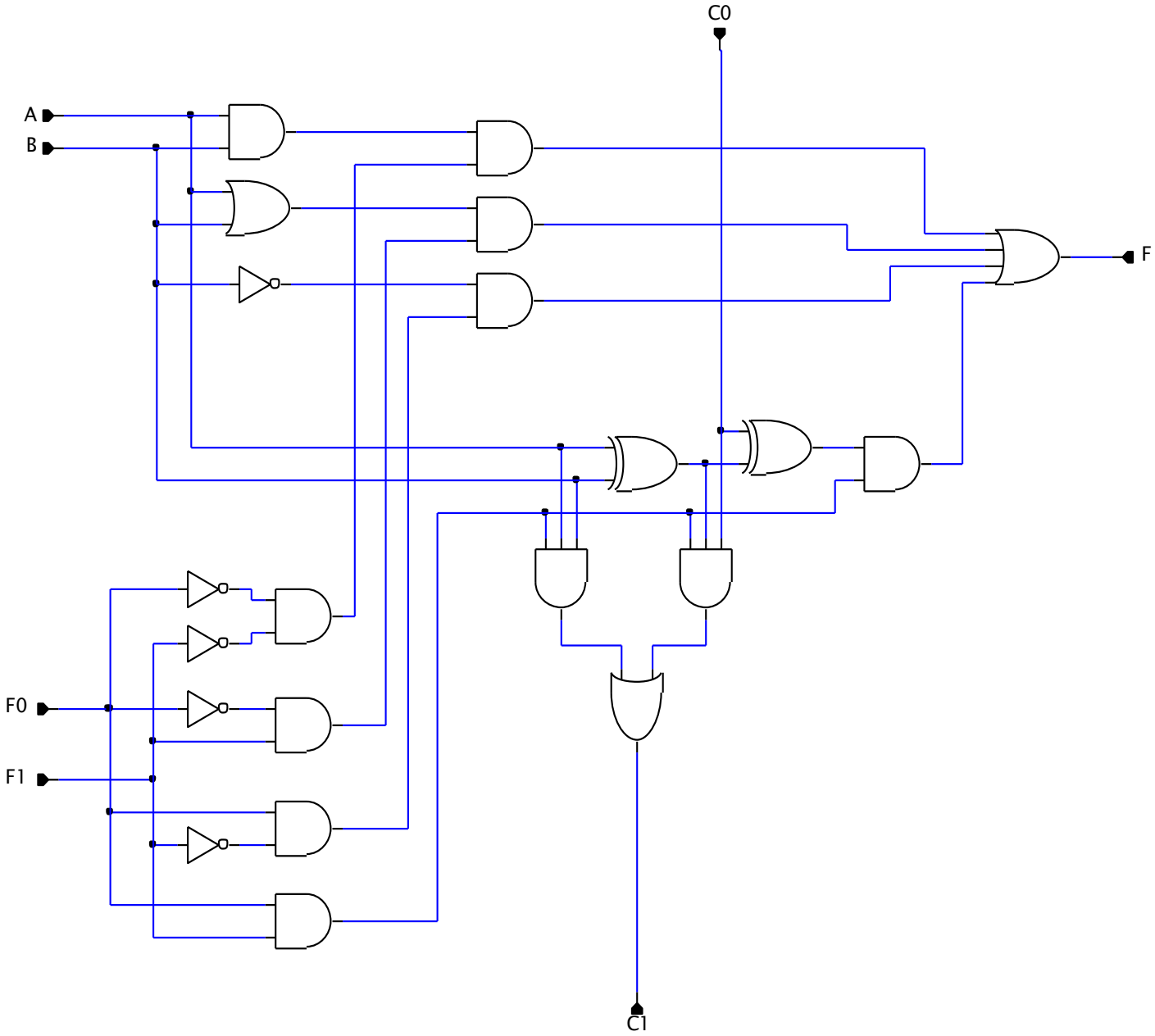
+23 = 0001 0111

Circuitos de ULAs

Circuitos Digitais I

Prof. Fernando Passold

ULA simples de 1 bit:



ULA:
-**U**nidade **L**ógica e **A**ritmética;
-Resume em si mesma a posibilidade de execução de operações básicas (AND, OR, NOT) e soma de duas palavras binárias.

↖ Figura ao lado: ULA simples de 1 bit.

(Cont) ULA simples de 1 bit:

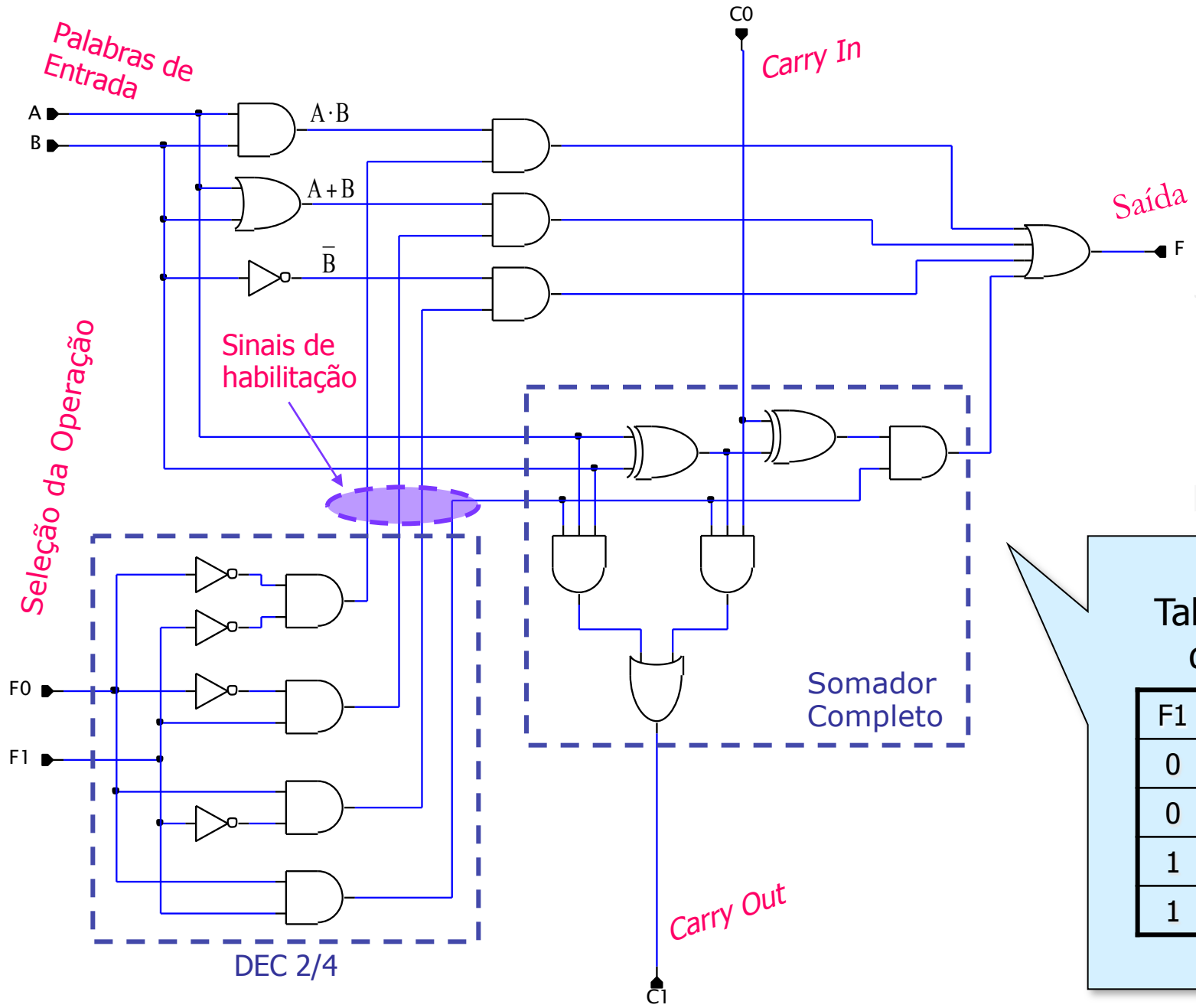
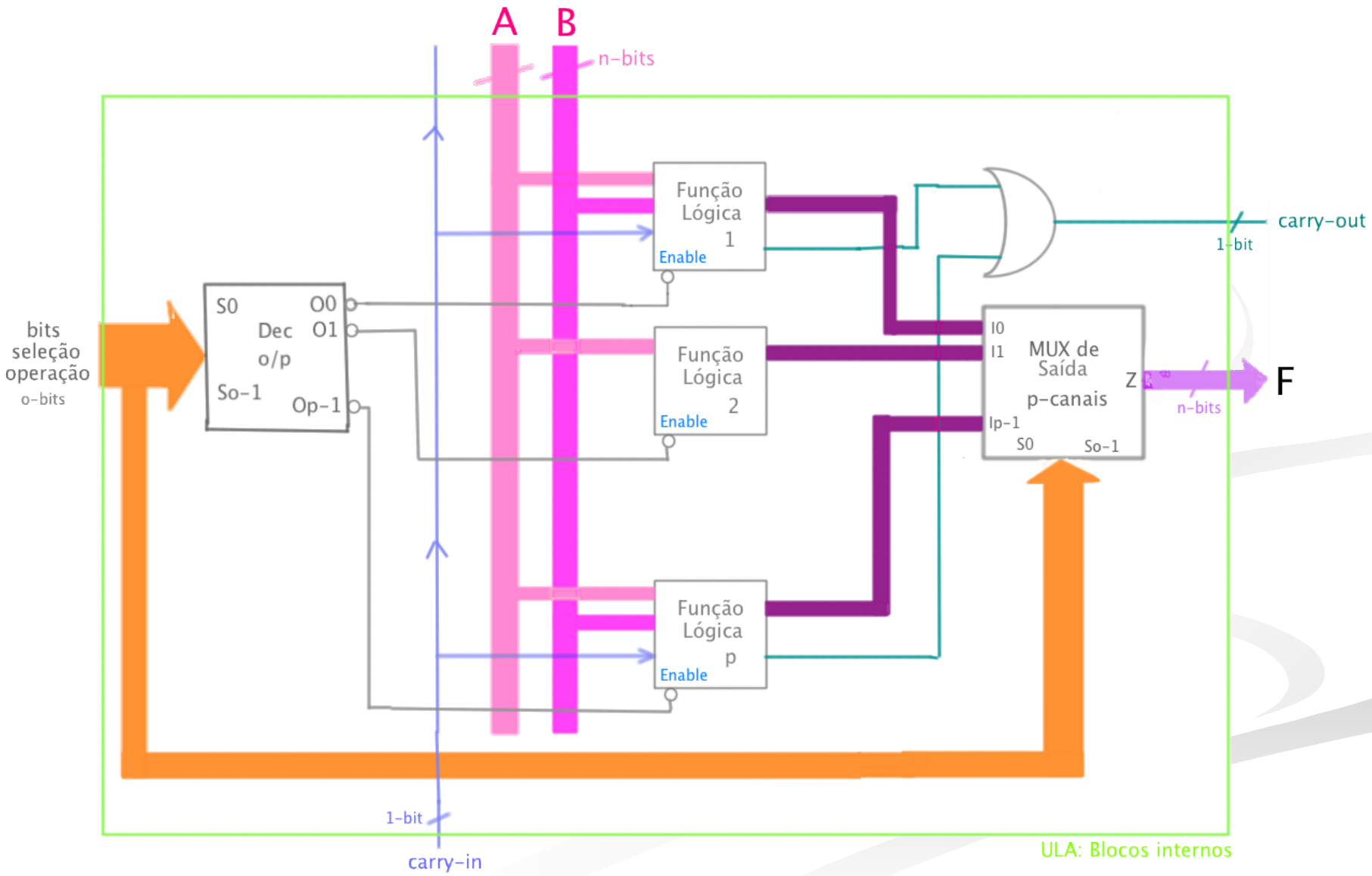


Tabela Verdade desta ULA:

F1	F0	Saída
0	0	A AND B
0	1	A OR B
1	0	NOT(B)
1	1	A + B

Arquitetura genérica de uma ULA



ULA 74LS382/HC382:

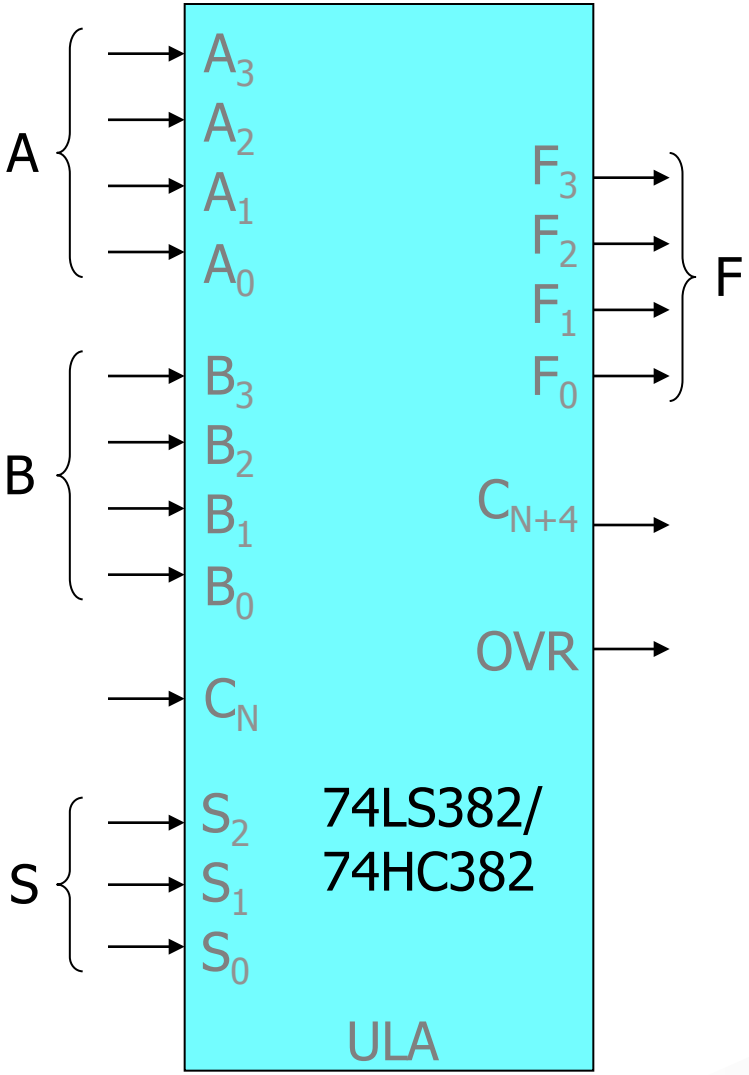


Tabela de função do 74HC382:

S_2 S_1 S_0	Operação	Comentário
0 0 0	CLEAR	$F_3F_2F_1F_0 = 0000$
0 0 1	B – A	Requer $C_N = 1$
0 1 0	A – B	Requer $C_N = 1$
0 1 1	A + B	Requer $C_N = 0$
1 0 0	$A \oplus B$	XOR
1 0 1	A + B	OR
1 1 0	AB	AND
1 1 1	PRESET	$F_3F_2F_1F_0 = 1111$

Coment/arios:

A = número de entrada de 4 bits.

B = número de entrada de 4 bits.

C_N = "*carry-in*" en posición LSB.

S = 3 bits para seleccionar operación.

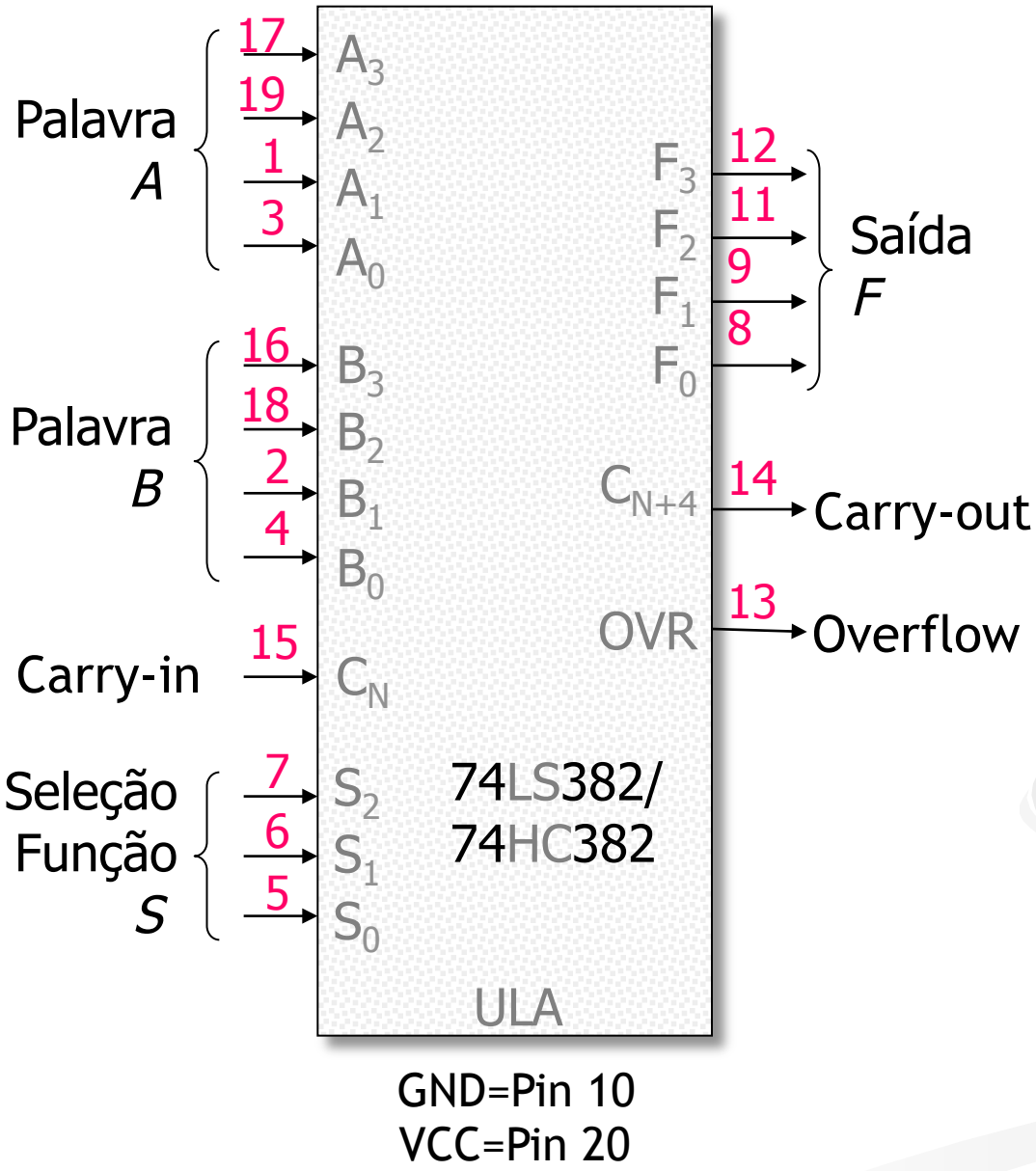
F = número de salida de 4 bits.

C_{N+4} = "*carry-out*" de posición MSB.

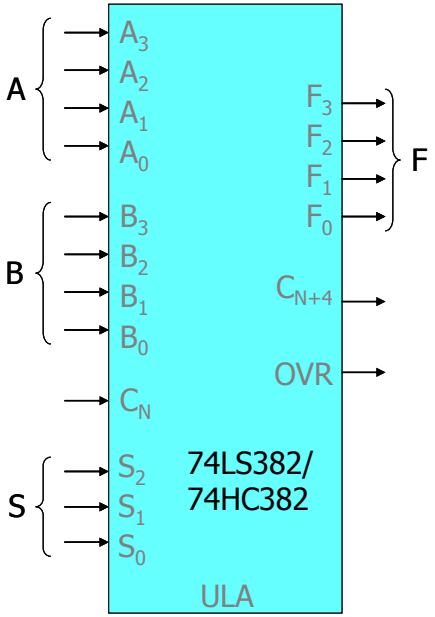
OVR = 1 para o*verflow* de números con signo.

NULL =1 cuando salida igual a cero

SGN = 1 para indicar resultado negativo


Palavra
A
17
19
1
3
A3
A2
A1
A0
Palavra
B
16
18
2
4
B3
B2
B1
B0
Carry-in
15
CN
Seleção
Função
S
7
6
5
S2
S1
S0
74LS382/
74HC382
ULA
F3
F2
F1
F0
12
11
9
8
Saída
F
CN+4
14
Carry-out
OVR
13
Overflow
GND=Pin 10
VCC=Pin 20

ULA 74LS382/HC382 (cont)



Exemplos de Operações:

- Subtração (em \C2):
  - Com $S_2S_1S_0 = 001$, a ULA subtrai o número de entrada A do número de entrada B. O resultado surge em $F_3F_2F_1F_0$.
    Note que a operação de substração exige C_N=1!
  - Compare com:

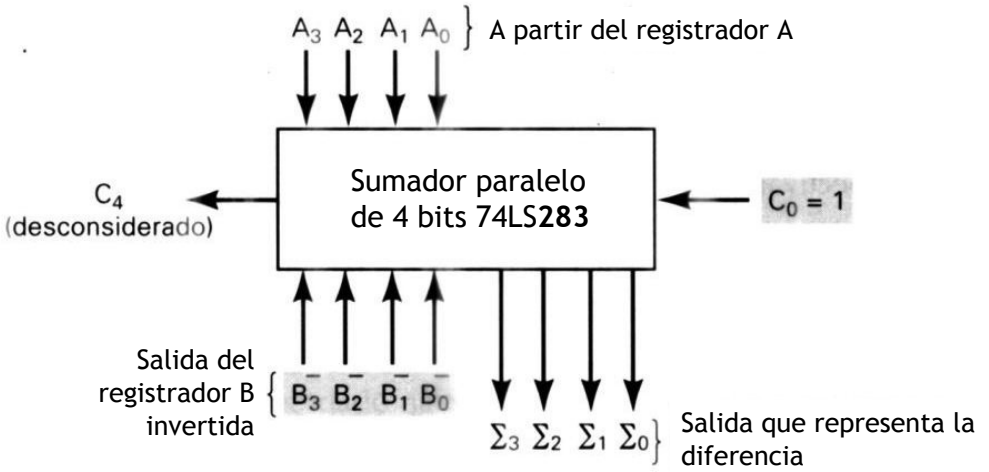


Tabela de funções do 74HC382:

S_2 S_1 S_0	Operação	Comentário
0 0 0	CLEAR	$F_3F_2F_1F_0 = 0000$
0 0 1	B – A	Requer $C_N = 1$
0 1 0	A – B	Requer $C_N = 1$
0 1 1	A + B	Requer $C_N = 0$
1 0 0	$A \oplus B$	XOR
1 0 1	A + B	OR
1 1 0	AB	AND
1 1 1	PRESET	$F_3F_2F_1F_0 = 1111$

ULA 74LS382/HC382 (cont)

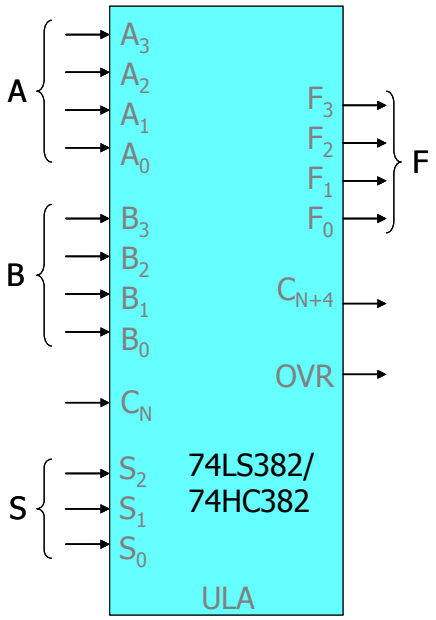


Tabela de funções do 74HC382:

S_2 S_1 S_0	Operação	Comentário
0 0 0	CLEAR	$F_3F_2F_1F_0$ = 0000
0 0 1	B – A	Requer C_N = 1
0 1 0	A – B	Requer C_N =1
0 1 1	A + B	Requer C_N =0
1 0 0	$A \oplus B$	XOR
1 0 1	A + B	OR
1 1 0	AB	AND
1 1 1	PRESET	$F_3F_2F_1F_0$ = 1111

Exemplos de Operações:

Subtração (em \C2):

Com $S_2S_1S_0$ = 001, a ULA subtrai o número de entrada A do número de entrada B. O resultado surge em $F_3F_2F_1F_0$.

Note que a operação de substração exige C_N=1!

- Operação XOR:
  - Com $S_2S_1S_0$ = 100, a ULA realiza XOR bit à bit sobre as entradas A e B.
    - Se $A_3A_2A_1A_0$=0110 e $B_3B_2B_1B_0$=1100, será obtido:

$$F_3 = A_3 \oplus B_3 = 0 \oplus 1 = 1$$
$$F_2 = A_2 \oplus B_2 = 1 \oplus 1 = 0$$
$$F_1 = A_1 \oplus B_1 = 1 \oplus 0 = 1$$
$$F_0 = A_0 \oplus B_0 = 0 \oplus 0 = 0$$

ULA 74LS382/HC382 (cont)

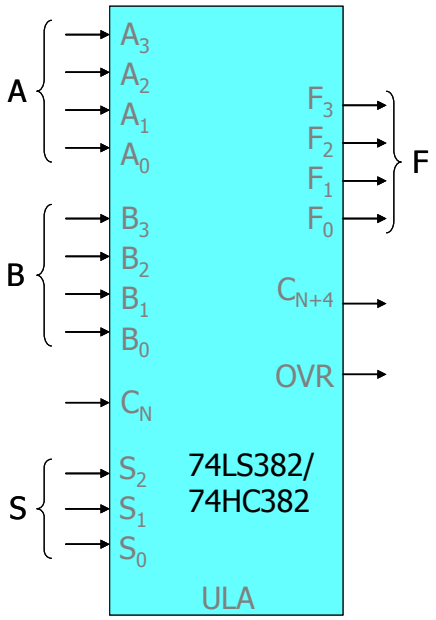


Tabela de funções do 74HC382:

S_2 S_1 S_0	Operação	Comentário
0 0 0	CLEAR	$F_3F_2F_1F_0 = 0000$
0 0 1	B – A	Requer $C_N = 1$
0 1 0	A – B	Requer $C_N = 1$
0 1 1	A + B	Requer $C_N = 0$
1 0 0	$A \oplus B$	XOR
1 0 1	A + B	OR
1 1 0	AB	AND
1 1 1	PRESET	$F_3F_2F_1F_0 = 1111$

Problemas:

1) Determine as saídas do 74LS382 para as seguintes entradas:
$S_2S_1S_0 = 010$;
$A_3A_2A_1A_0 = 0100$;
$B_3B_2B_1B_0 = 0001$, e $C_N = 1$.

Resposta:

$F_3F_2F_1F_0 = 0011$
$C_{N+4} = 1$
OVR = 0

2) Altere o código de selección anterior para 011 e repita o exercício mantendo as mesmas entradas.

Resposta:

$F_3F_2F_1F_0 = 0110$
$C_{N+4} = 0$
OVR = 0

ULA 74LS382/HC382 (cont)

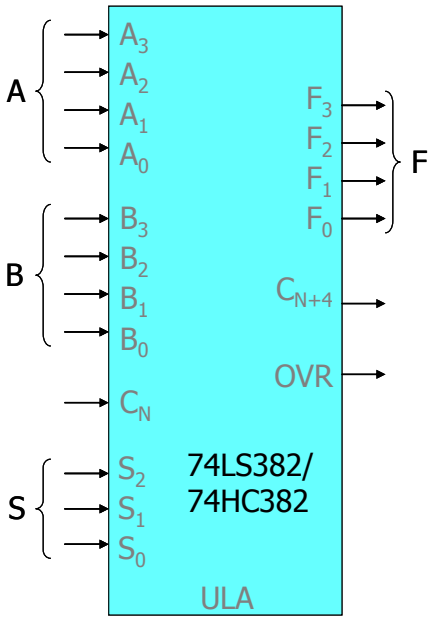


Tabela de funções do 74HC382:

S_2 S_1 S_0	Operação	Comentário
0 0 0	CLEAR	$F_3F_2F_1F_0 = 0000$
0 0 1	B – A	Requer $C_N = 1$
0 1 0	A – B	Requer $C_N = 1$
0 1 1	A + B	Requer $C_N = 0$
1 0 0	$A \oplus B$	XOR
1 0 1	A + B	OR
1 1 0	AB	AND
1 1 1	PRESET	$F_3F_2F_1F_0 = 1111$

Problemas:

1) Determine as saídas do 74LS382 para as seguintes entradas:
$S_2S_1S_0 = 010$;
$A_3A_2A_1A_0 = 0100$;
$B_3B_2B_1B_0 = 0001$, e $C_N = 1$.

   Resposta:

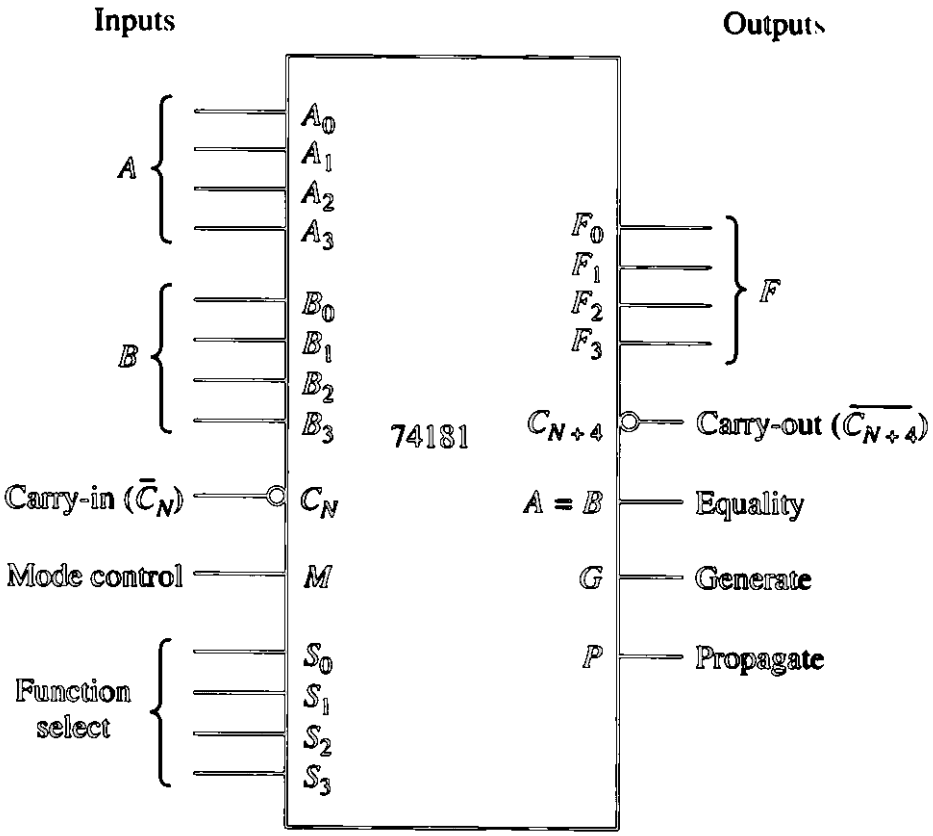
   $F_3F_2F_1F_0 = 0011$
   $C_{N+4} = 1$
   OVR = 0

2) Altere o código de selección anterior para 011 e repita o exercício mantendo as mesmas entradas.

   Resposta:

   $F_3F_2F_1F_0 = 0110$
   $C_{N+4} = 0$
   OVR = 0

Outra ULA: 74181



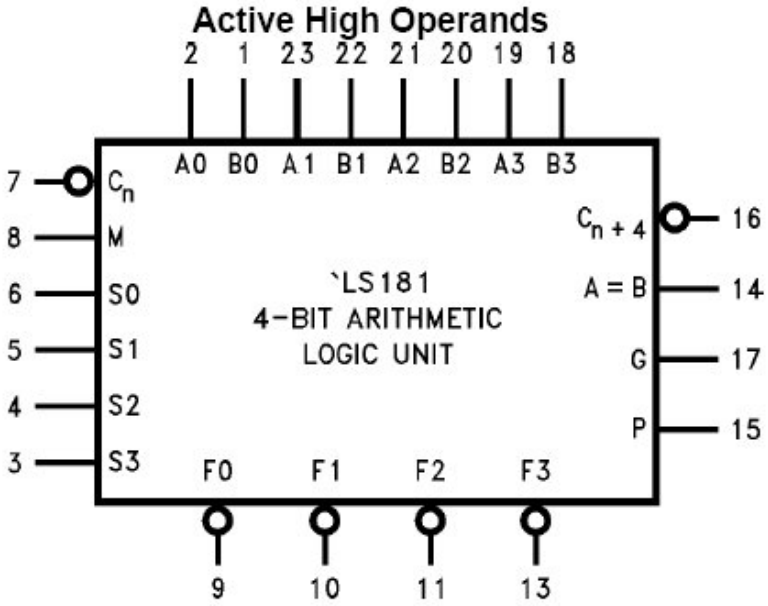
Mode select				Logic functions	Arithmetic operations
S_3	S_2	S_1	S_0	$(M = H)$	$(M = L)(\overline{C}_n = H)$
L	L	L	L	$F = \overline{A}$	$F = A$
L	L	L	H	$F = \overline{A + B}$	$F = A + B$
L	L	H	L	$F = \overline{A}B$	$F = A + \overline{B}$
L	L	H	H	$F = 0$	$F =$ minus 1 (2's comp.)
L	H	L	L	$F = \overline{AB}$	$F = A$ plus $A\overline{B}$
L	H	L	H	$F = \overline{B}$	$F = (A + B)$ plus $A\overline{B}$
L	H	H	L	$F = A \oplus B$	$F = A$ minus B minus 1
L	H	H	H	$F = A\overline{B}$	$F = A\overline{B}$ minus 1
H	L	L	L	$F = \overline{A} + B$	$F = A$ plus AB
H	L	L	H	$F = \overline{A \oplus B}$	$F = A$ plus B
H	L	H	L	$F = B$	$F = (A + \overline{B})$ plus AB
H	L	H	H	$F = AB$	$F = AB$ minus 1
H	H	L	L	$F = 1$	$F = A$ plus A*
H	H	L	H	$F = A + \overline{B}$	$F = (A + B)$ plus A
H	H	H	L	$F = A + B$	$F = (A + \overline{B})$ plus A
H	H	H	H	$F = A$	$F = A$ minus 1

*Each bit is shifted to the next-more-significant position.

Obs.: pinos G e P previstos para gerar o propagar de forma acelerada do *carry-out* (*fast-look-ahead carry*)

Simulador (em java) disponible em (25/11/2008):
http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/20-arithmetic/50-74181/demo-74182-ALU-CLA.html

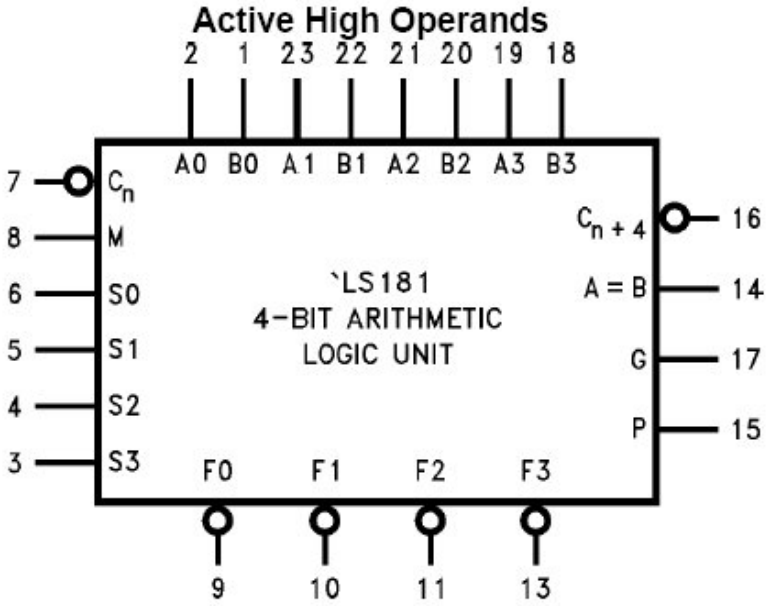
Outra ULA: 74181



Obs: pines G y P previstos para generar o propagar de forma acelerada el *carry-out* (*fast-look-ahead carry*).

Seletor de operación S_3 S_2 S_1 S_0	**Lógica** (M=1)	**Aritmética** (M=0) (~C_{in}=0)	**Aritmética** (M=0) (~C_{in}=1)
0 0 0 0	~A	A + 1	A
0 0 0 1	~(A | B)	(A | B) + 1	A | B
0 0 1 0	~A & B	(A | ~B) + 1	A | ~B
0 0 1 1	0 (zero)	0 (zero)	-1
0 1 0 0	~(A & B)	A + (A & ~B) + 1	A + (A & ~B)
0 1 0 1	~B	(A | B)+(A & ~B) +1	(A | B) + (A & ~B)
0 1 1 0	A ^ B	A - B	A - B - 1
0 1 1 1	A & ~B	A & B	(A & B) - 1
1 0 0 0	~A | B	A + (A & B) + 1	A + (A & B)
1 0 0 1	~(A ^ B)	A + B + 1	A + B
1 0 1 0	B	(A | ~B)+(A & B) +1	(A | ~B) + (A & B)
1 0 1 1	A & B	A & B	(A & B) - 1
1 1 0 0	-1	A + A + 1	A + A
1 1 0 1	A | ~B	(A | B) + A + 1	(A | B) + A
1 1 1 0	A | B	(A | ~B) + A + 1	(A | ~B) + A
1 1 1 1	A	A	A - 1
Obs: ~	NOT Negación (complementa los bits).	^	OR

Outra ULA: 74181

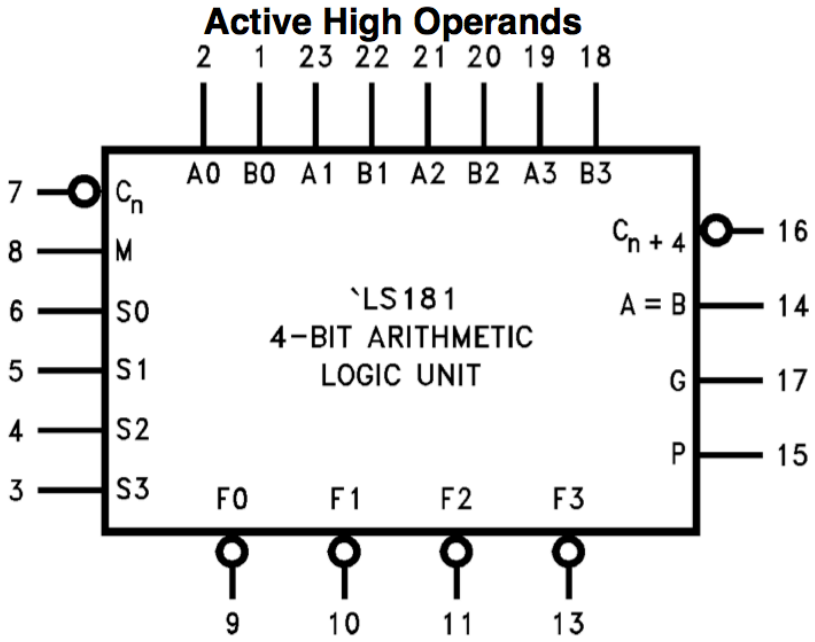


Códigos de selección				Funciones lógicas	Funciones aritméticas *	
					M = 0	
S3	S2	S1	S0	M = 1	Cn = 1 (sin acarreo)	Cn = 0 (con acarreo)
0	0	0	0	$\overline{A}$	A	A + 1
0	0	0	1	$\overline{A+B}$	A + B	(A + B) + 1
0	0	1	0	$\overline{A}B$	A + $\overline{B}$	(A + $\overline{B}$) + 1
0	0	1	1	0	-1	0
0	1	0	0	$\overline{AB}$	A + A$\overline{B}$	A + A$\overline{B}$ + 1
0	1	0	1	$\overline{B}$	(A + B) + A$\overline{B}$	(A + B) + A$\overline{B}$ + 1
0	1	1	0	A $\oplus$ B	A - B - 1	A - B
0	1	1	1	A$\overline{B}$	A$\overline{B}$ - 1	A$\overline{B}$
1	0	0	0	$\overline{A}$ + B	A + AB	A + AB + 1
1	0	0	1	$\overline{A \oplus B}$	A + B	A + B + 1
1	0	1	0	B	(A + $\overline{B}$) + AB	(A + $\overline{B}$) + AB + 1
1	0	1	1	AB	AB - 1	AB
1	1	0	0	1	A + A	A + A + 1
1	1	0	1	A + $\overline{B}$	(A + B) + A	(A + B) + A + 1
1	1	1	0	A + B	(A + $\overline{B}$) + A	(A + $\overline{B}$) + A + 1
1	1	1	1	A	A - 1	A

*: Expresadas en complemento a 2
+: Operador OR en funciones lógicas y signo más en operaciones aritméticas
-: Signo aritmético menos $\overline{\ }$: Barra de inversión lógica

Obs: pines G y P previstos para generar o propagar de forma acelerada el *carry-out* (*fast-look-ahead carry*).

Outra ULA: 74181 (cont.)

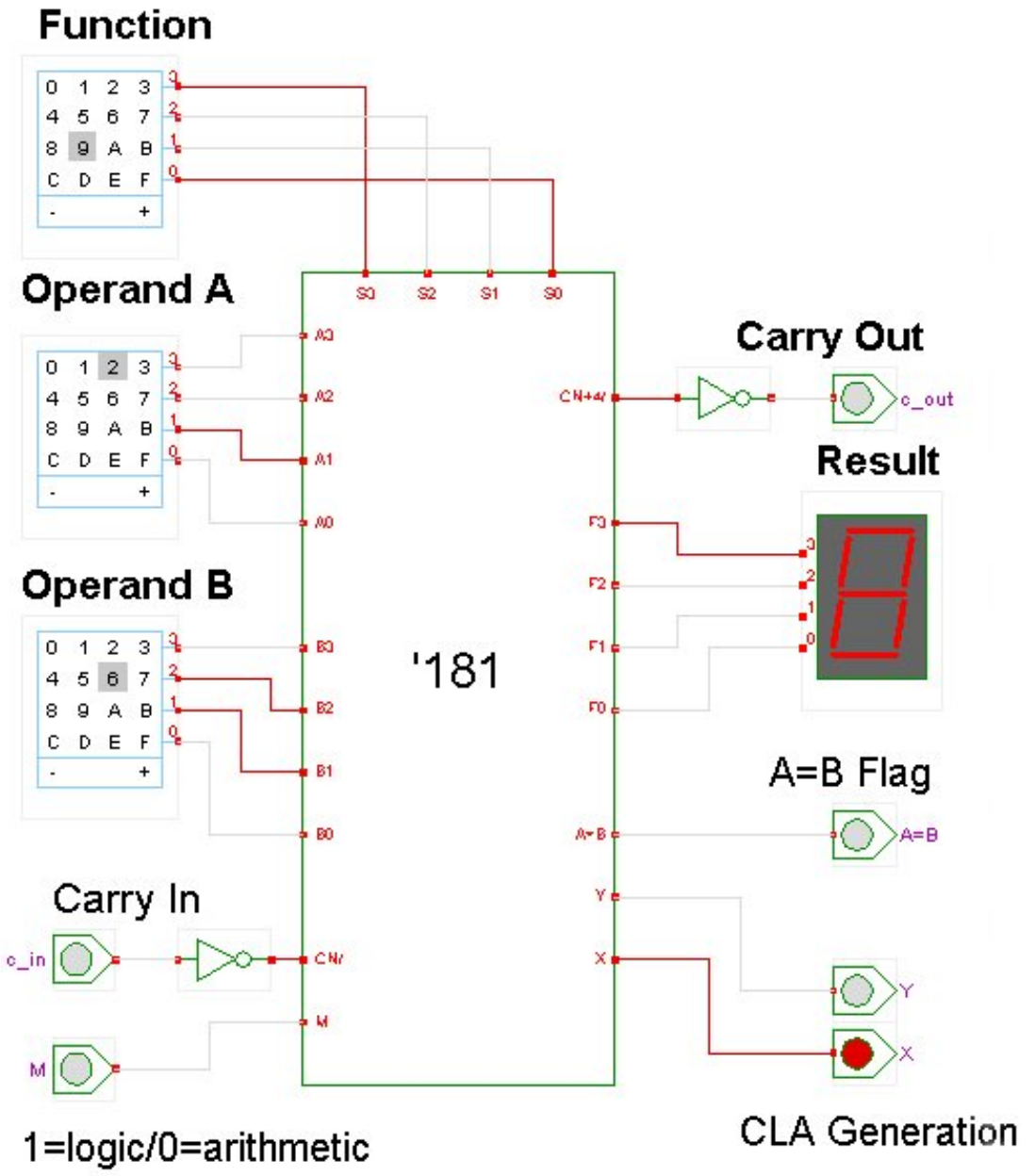


Distribuição e função dos pinos:

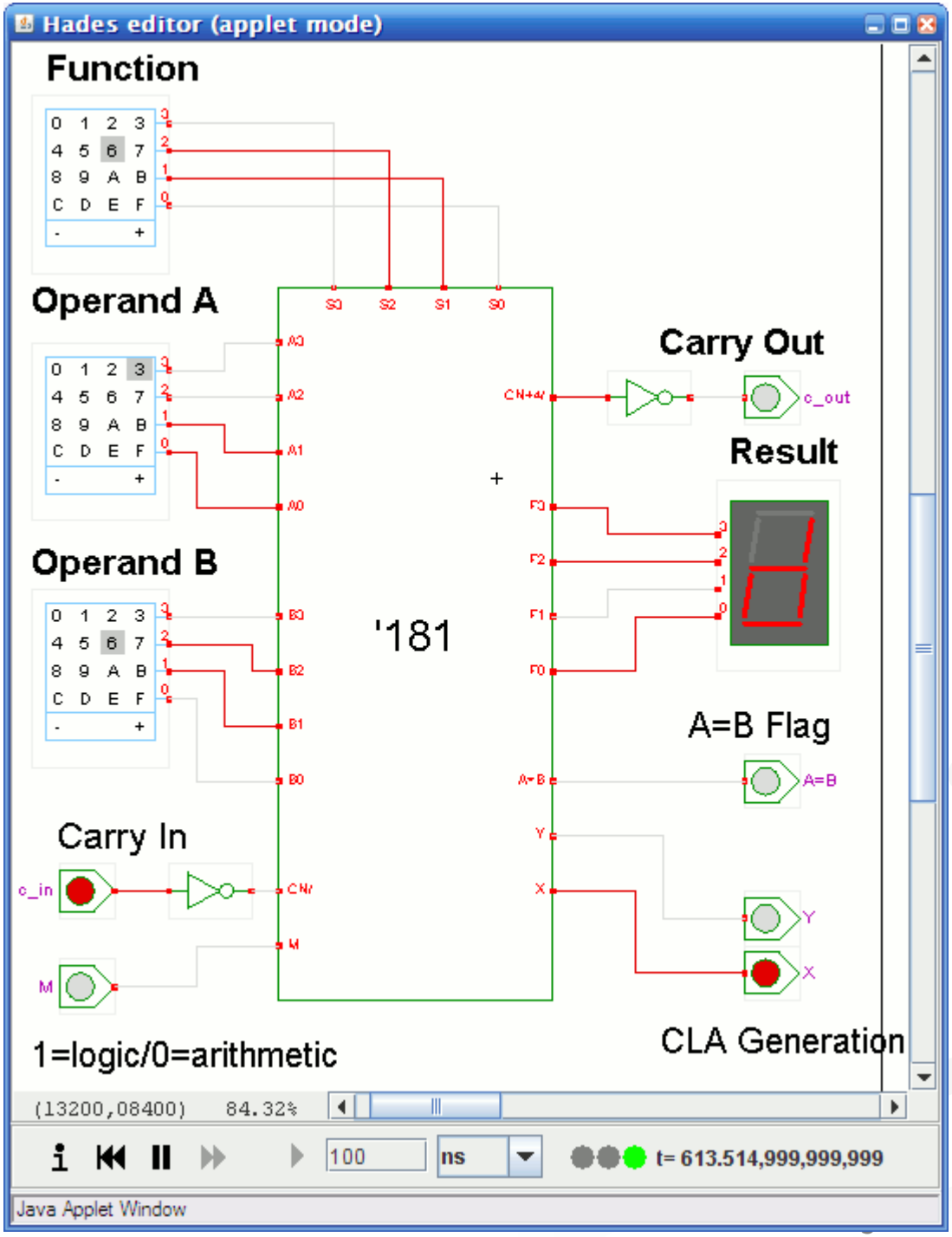
- A0 ... A3, entradas de operandos.
- B0 ... B3, entradas de operandos.
- /F0 ... /F3, saídas (ATIVO BAIXO) da ALU: onde aparecem os resultados das operações.
- Entradas S0 ... S3, correspondem as linhas de seleção de operação do circuito; mediante estas se seleciona a função que o circuito deve realizar.
- Entrada /M, entrada de controle (ATIVO BAIXO); por meio desta entrada se indica ao circuito a operação a realizar:
  - Si /M=1, realiza operações lógicas, e;
  - Si /M=0, realiza operações aritméticas.
- Entrada /Cn, entrada de "carry-in" (ATIVO BAIXO). Esta entrada deverá estar ativada ou não de acordo com a operação aritmética à ser realizada.
- Saída /Cn+4 é a saída de "carry-out" (ATIVO BAIXO).
- Saída A=B, é una saída em coletor aberto e indica quando os operadores de entrada são iguais. Por exemplo, quando é realizada a operação aritmética de subtração, esta saída se ativa quando ambos os operandos são iguais.
- Saída /G, de geração acelerada de carry (ATIVO BAIXO). Em operações aritméticas de soma, esta saída (se ativa) indica que o resultado F é maior ou igual a 16, e no caso de subtração, indica quando F é menor que zero.
- Saída /P, saída para propagação acelerada de carry (ATIVO BAIXO).
- As saídas /G e /P se usam para cascatear vários circuitos integrados do tipo 74181 empregando o método de propagação de carry em paralelo.

ULA 74181

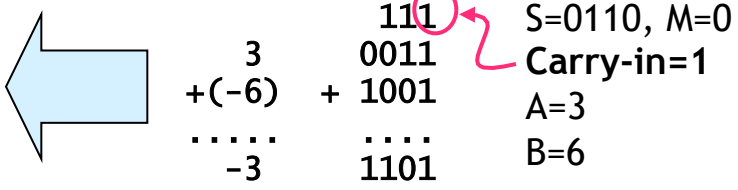
Selector de operación $S_3\ S_2\ S_1\ S_0$	Aritmética (M=0) ($\sim C_{in}$=1)
0 0 0 0	A
0 0 0 1	A | B
0 0 1 0	A | ~B
0 0 1 1	-1
0 1 0 0	A + (A & ~B)
0 1 0 1	(A | B) + (A & ~B)
0 1 1 0	A - B - 1
0 1 1 1	(A & B) - 1
1 0 0 0	A + (A & B)
1 0 0 1	**A + B**
1 0 1 0	(A | ~B) + (A & B)
1 0 1 1	(A & B) - 1
1 1 0 0	A + A
1 1 0 1	(A | B) + A
1 1 1 0	(A | ~B) + A
1 1 1 1	A - 1



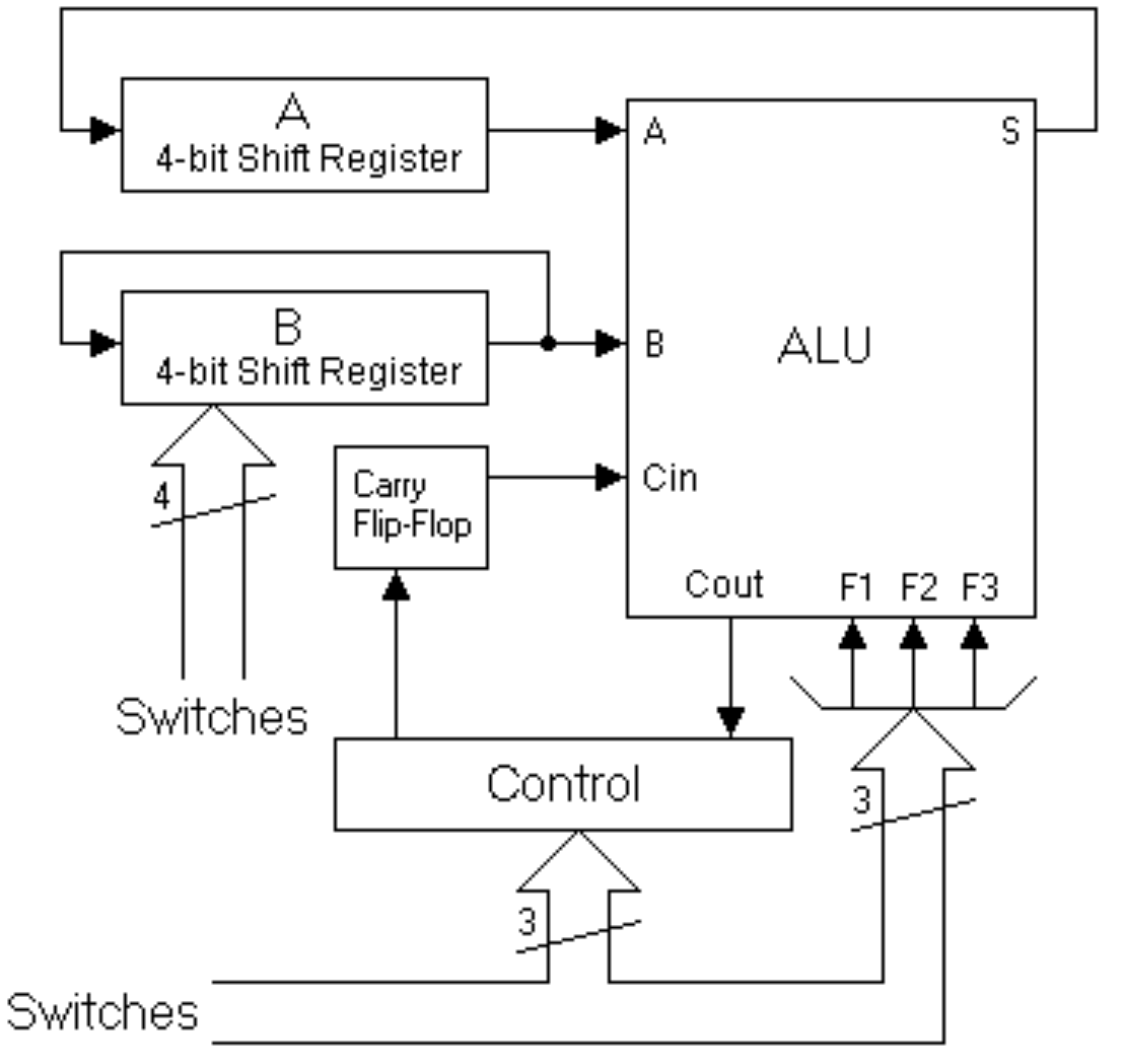
Simulador (em java) disponível em (25/11/2008):
http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/20-arithmetic/50-74181/demo-74181-ALU.html

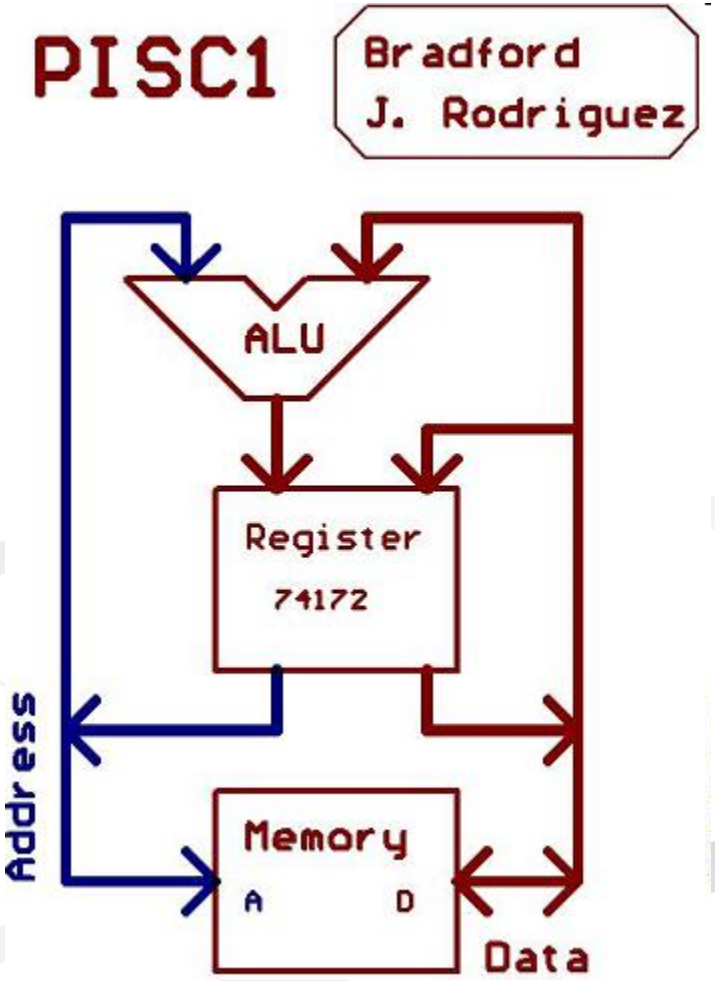


Selector de operación $S_3\ S_2\ S_1\ S_0$	Lógica (M=1)	Aritmética (M=0) (~C_{in}=0)
0 0 0 0	~A	A + 1
0 0 0 1	~(A | B)	(A | B) + 1
0 0 1 0	~A & B	(A | ~B) + 1
0 0 1 1	0 (zero)	0 (zero)
0 1 0 0	~(A & B)	A + (A & ~B) + 1
0 1 0 1	~B	(A | B)+(A & ~B) +1
0 1 1 0	**A ^ B**	**A - B**
0 1 1 1	A & ~B	A & B
1 0 0 0	~A | B	A + (A & B) + 1
1 0 0 1	~(A ^ B)	A + B + 1
1 0 1 0	B	(A | ~B)+(A & B) +1
1 0 1 1	A & B	A & B
1 1 0 0	-1	A + A + 1
1 1 0 1	A | ~B	(A | B) + A + 1
1 1 1 0	A | B	(A | ~B) + A + 1
1 1 1 1	A	A

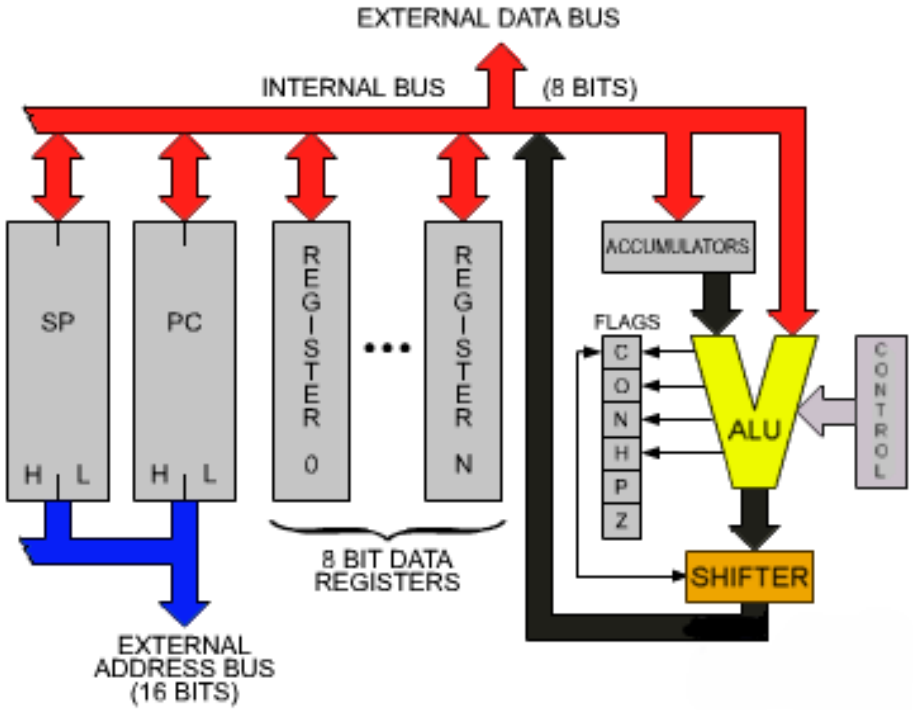


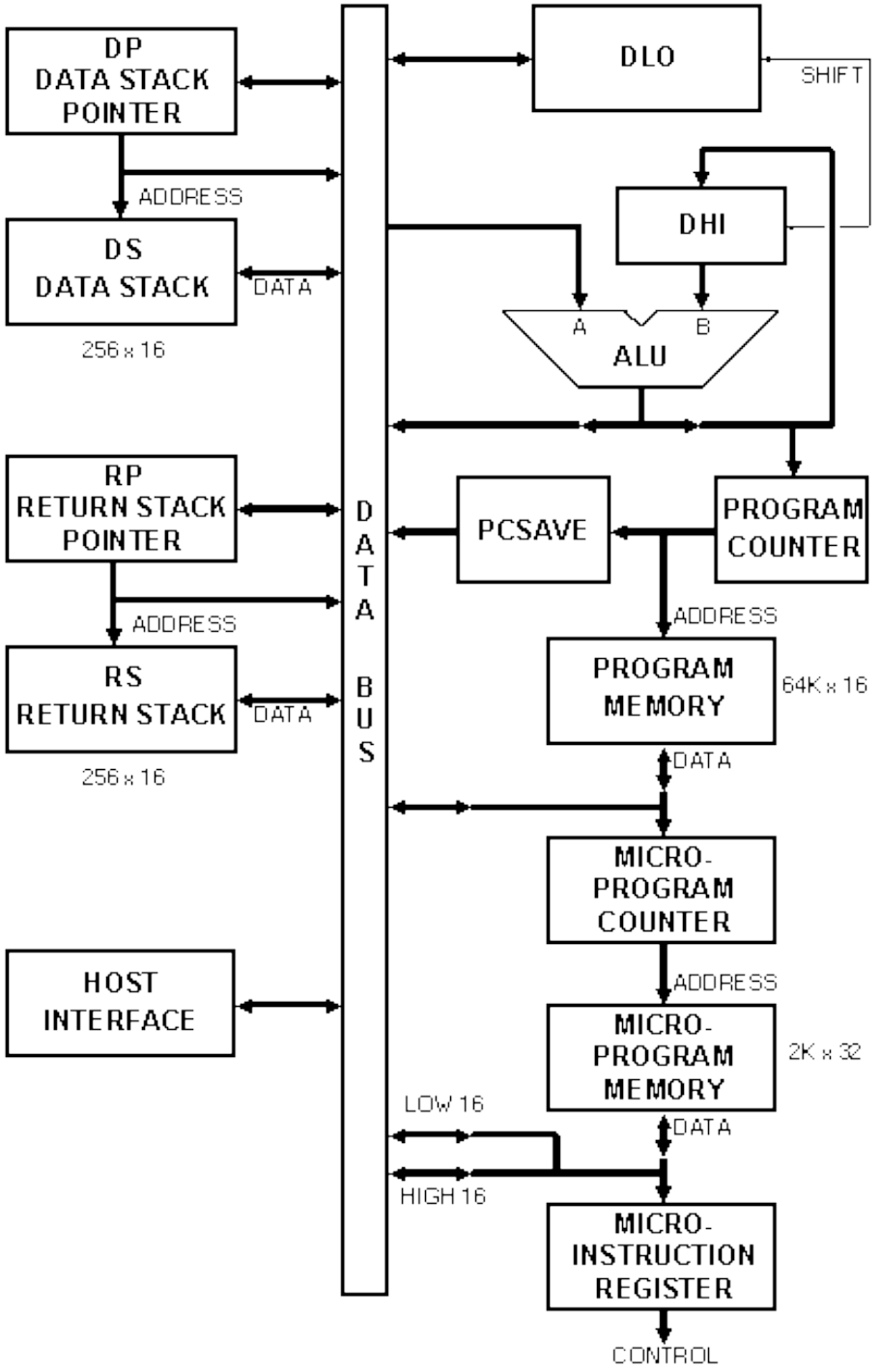
Aplicações de ULA → em CPUs



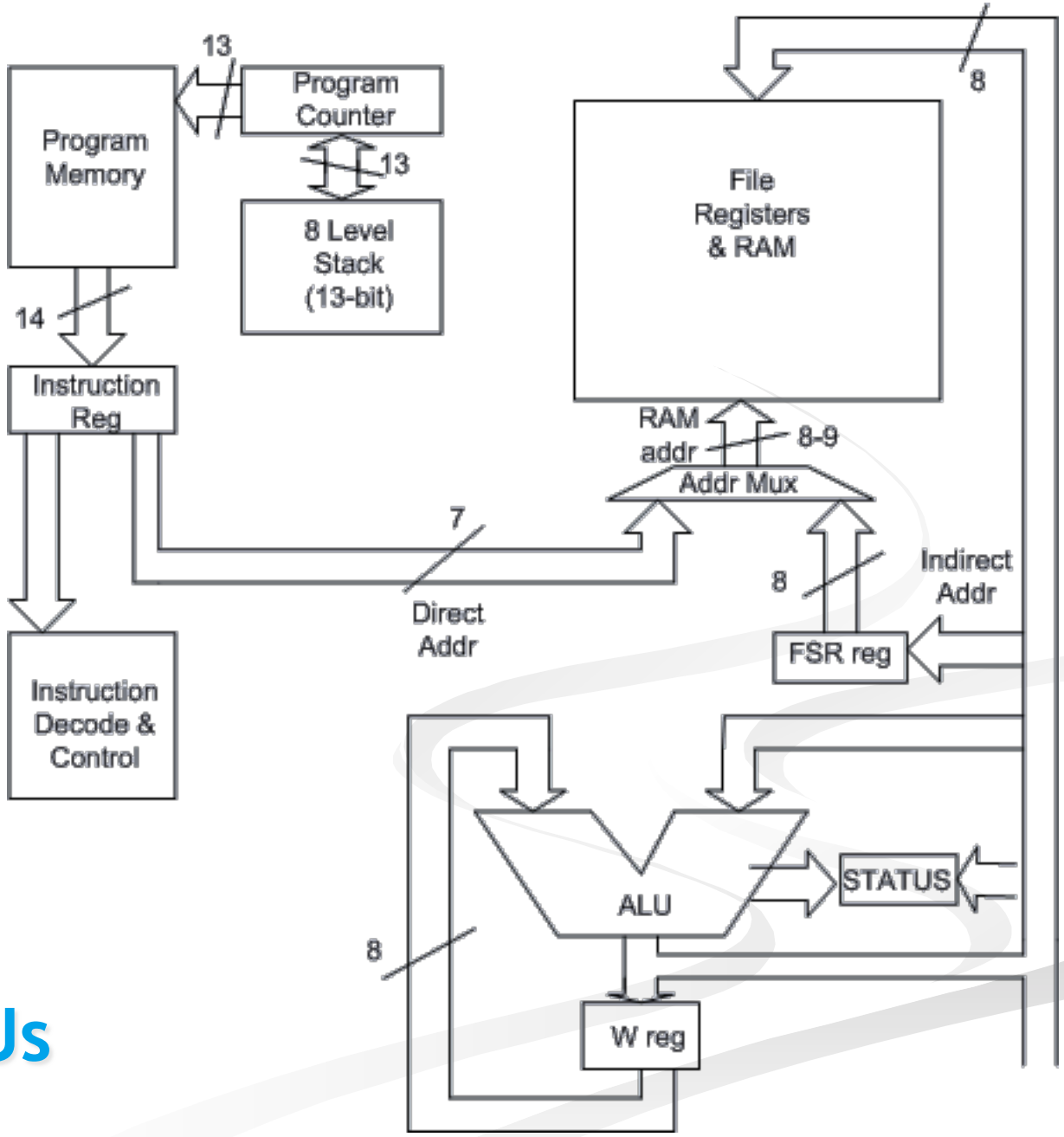


Aplicações de ULA → em CPUs

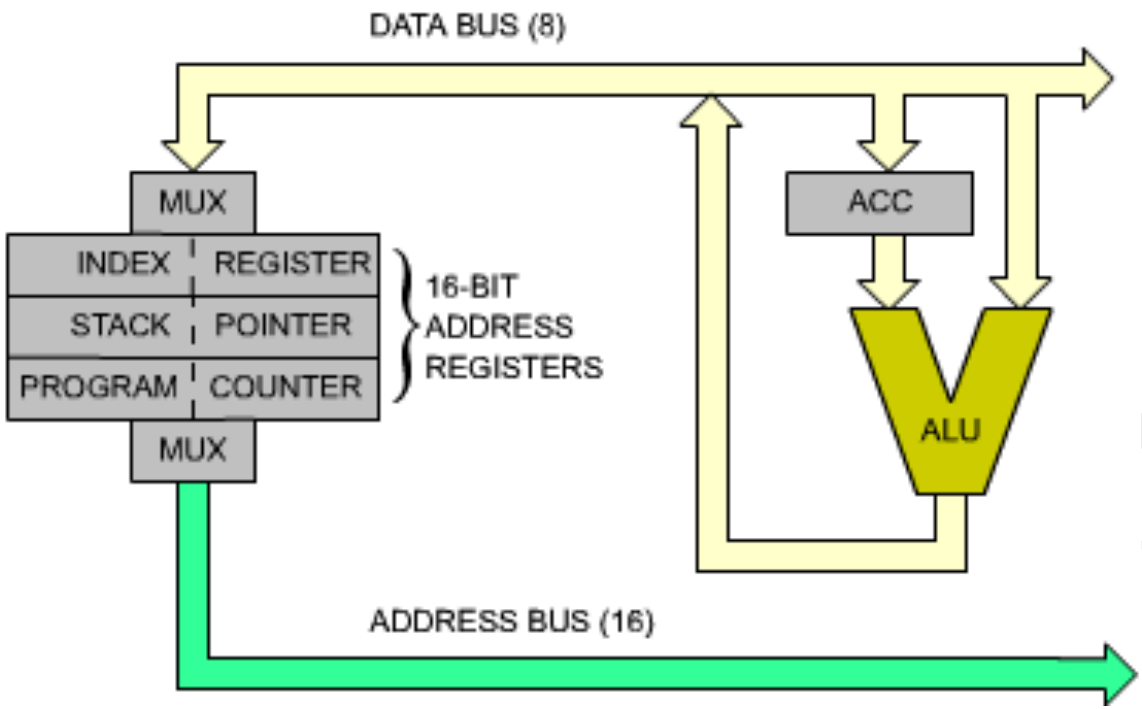




Aplicações de ULA → em CPUs



Aplicações de ULA → em CPUs



Exemplo de operação à nível de máquina (Linguagem Assembly):

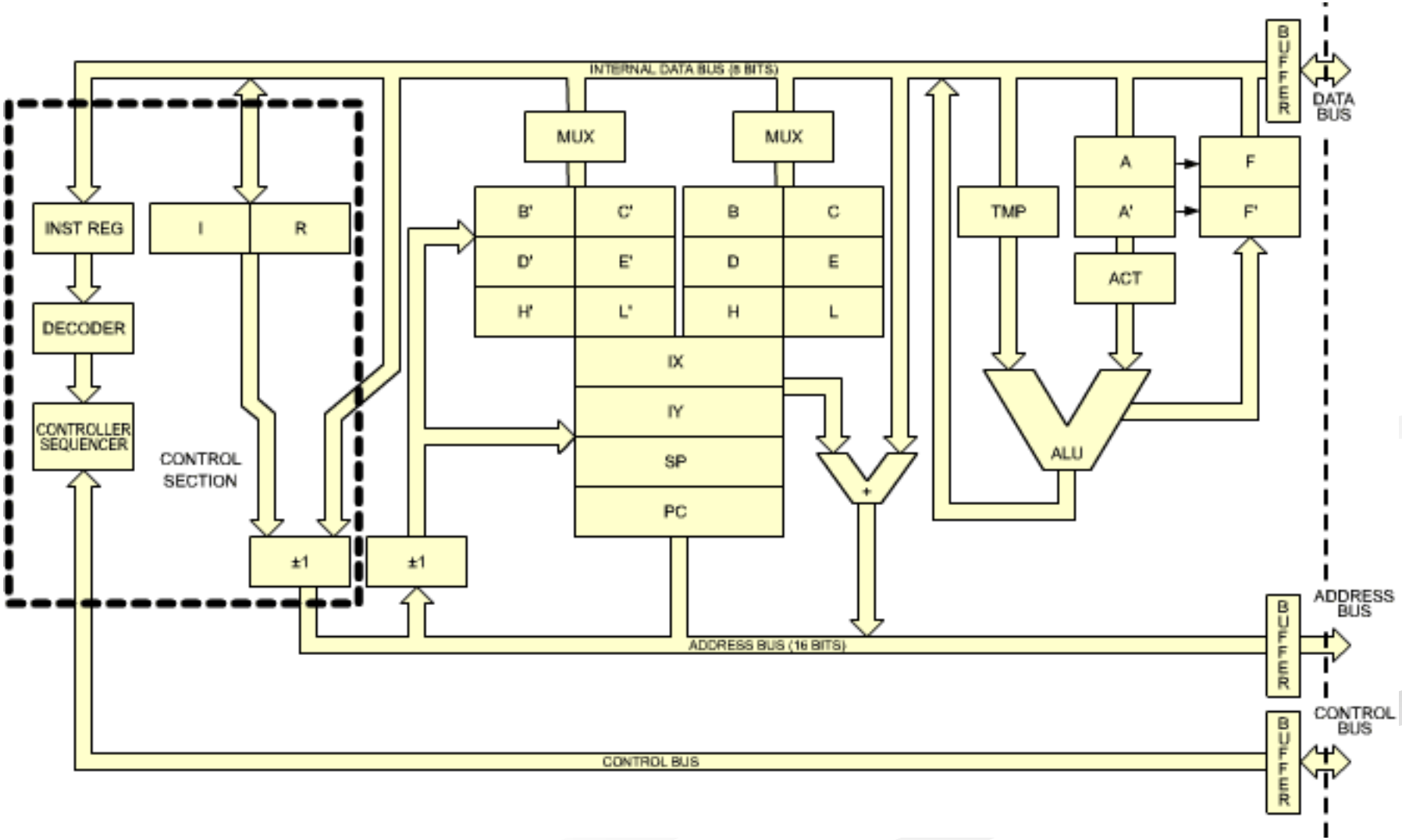
```
; Assembly program Z80

ORG 100H        ;Locate progra
LD HL,1234H     ;Address of fi
LD A,(HL)       ;Operand 1 int
INC HL          ;Address of 2r
ADD A,(HL)      ;Addition
INC HL          ;Address of re
LD (HL), A      ;store result
END             ;end of progra

ORG 1234H       ;Location of t
DB 100,200      ;put 64H and
```

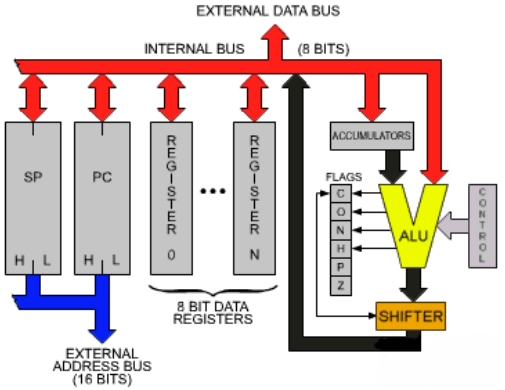
Fonte (da figura):
http://www.msxarchive.nl/pub/msx/mirrors/msx2.com/zaks/z80prg02.htm (Disponível em 10 Jun 2014)

Exemplo real: uP Z80 (8-bits)



Fonte: http://www.msxarchive.nl/pub/msx/mirrors/msx2.com/zaks/z80prg02.htm (Disponível em 10 Jun 2014)

Assembly - Exemplo:



Linha do programa

Posição na Memória (Endereço)

```
0001
0002
0003
0004   0100 21 34 12
0005   0103 7E
0006   0104 23
0007   0105 86
0008   0106 23
0009   0107 77
0108
0011
0012
0013   1234 64 C8

Number of errors = 0
```

Código Hexa (programa)

Outra Posição na Memória: Dados

```
; Assembly program Z80

ORG 100H          ;Locate program at 100H
LD HL,1234H       ;Address of first number
LD A,(HL)         ;Operand 1 into Accu
INC HL            ;Address of 2nd number
ADD A,(HL)        ;Addition
INC HL            ;Address of result (sum)
LD (HL), A        ;store result
END               ;end of program

ORG 1234H         ;Location of the data
DB 100,200        ;put 64H and C8H
```

Fonte: http://www.fb9dv.uni-duisburg.de/vs/de/education/Dv1/vorlesung2006/Z80-add.pdf (Disponível em 10 Jun 2014)

Assembly Z80 – Outros Exemplos.

```
ld a,5              ; load 5 into A
ld b,7              ; load 7 into B
add a,b             ; add B to A, store result to A
ld (Result),a       ; store the value of A into the variable Result
```

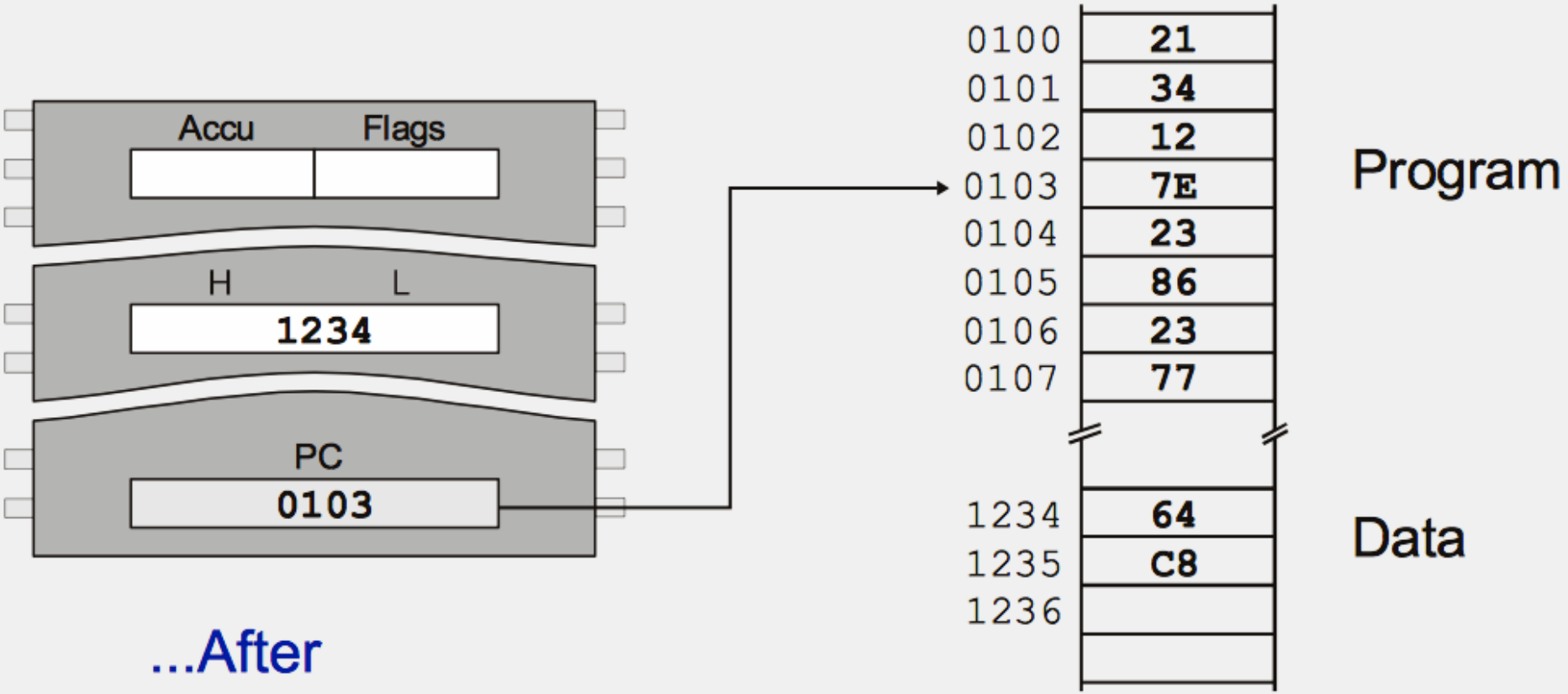
```
ld a,(Height)       ; loading Height into A
ld b,a              ; loading this value into B from A (it cannot be done
                    ; directly from the variable; see the possibilities below)
ld a,(Width)        ; loading Width into A
add a,a             ; adding A to itself, making it multiplied by two
add a,b             ; adding the height once
add a,b             ; adding the height again to complete the sum
ld (Perim),a        ; storing the result into Perim
```

Execução de programa em Assembly:

1st instruction:

```
LD    HL,1234H     ;Address of first number
```

Meaning: „HL:=1234H“ and (of course) „PC:=PC+3“

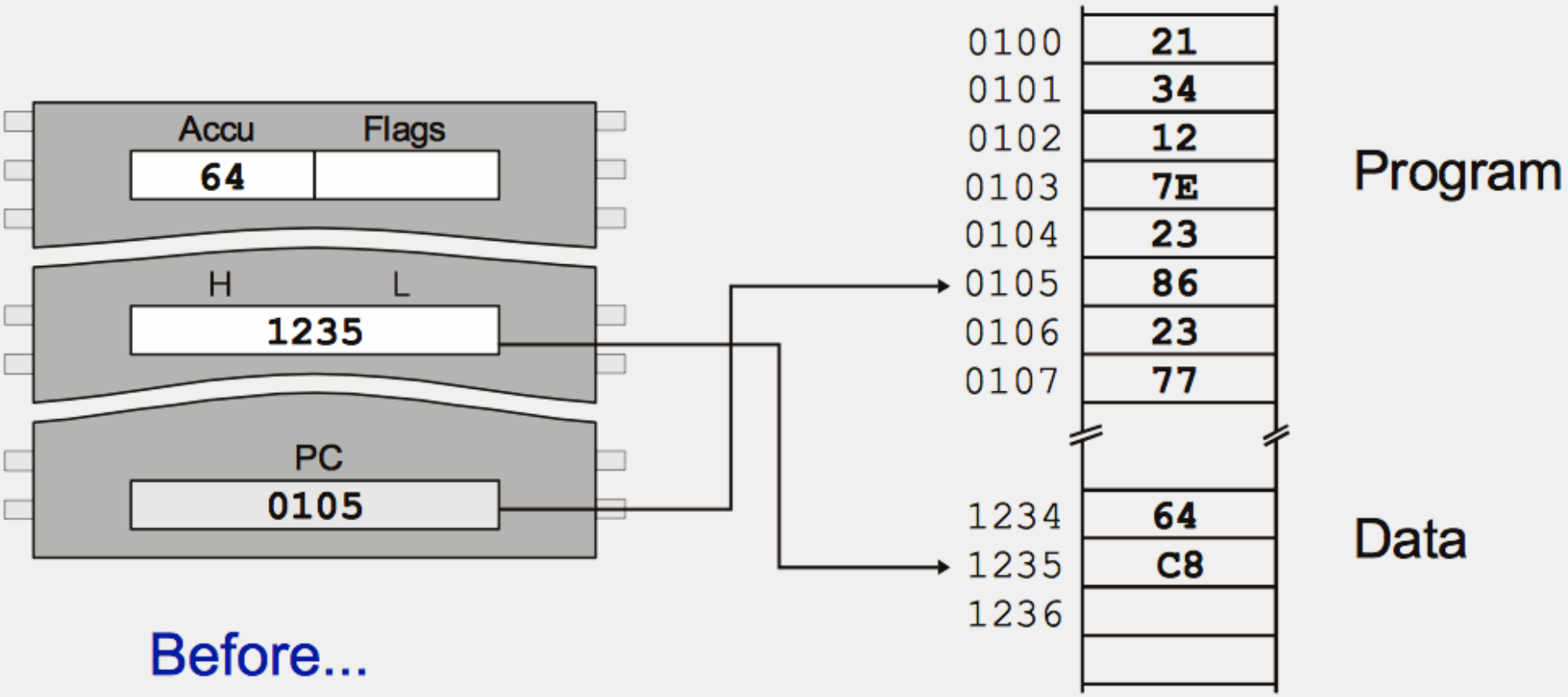


Execução de programa em Assembly:

4th instruction:

```
ADD   A,(HL)          ;Addition
```

Meaning: „A:=A+[HL] “ and „PC:=PC+1“



Registradores internos do Z80:

Z80 8-bit Data Registers

General Registers

A	Flags
B	C
D	E
H	L

Alternate Registers

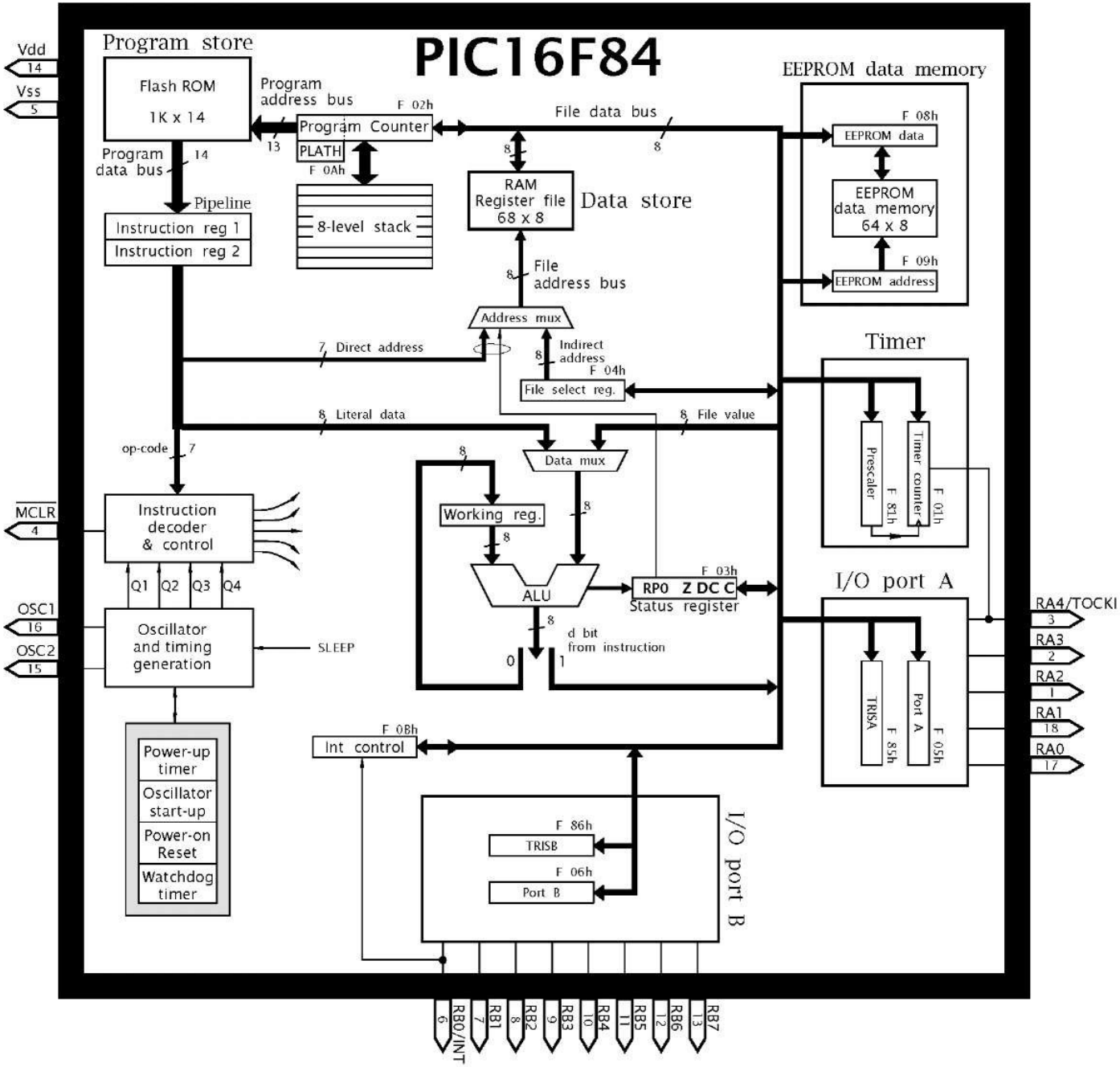
A'	F'
B'	C'
D'	E'
H'	L'

Z80 Flags

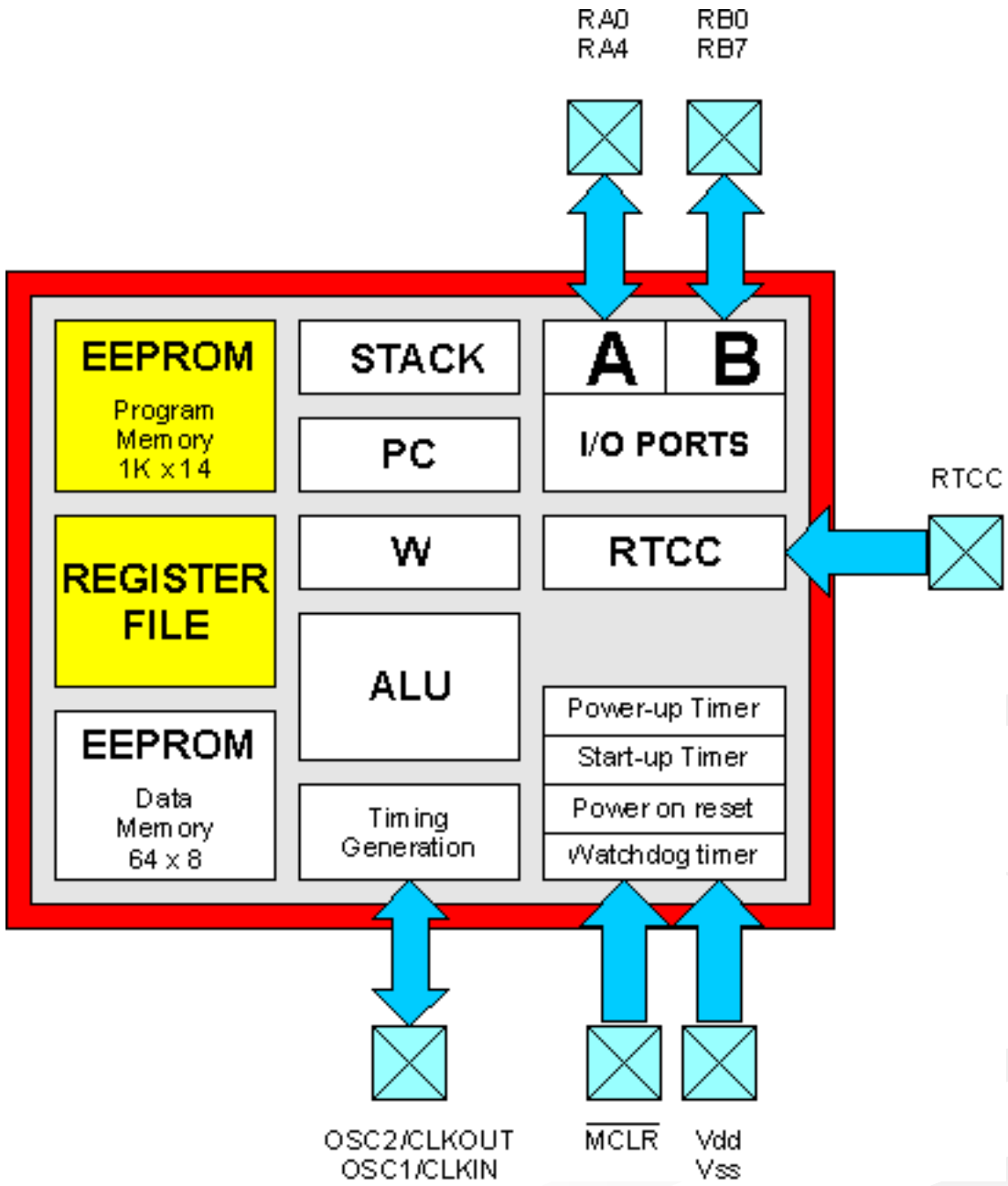
S	Z	X	H	X	P/V	N	C

- S Sign
- Z Zero
- X no flag
- H Half-Carry (Addition) for BCD Arithmetic
- X no flag
- P/V Parity/Overflow
- N Half-Carry (Subtraction) for BCD Arithmetic
- C Carry

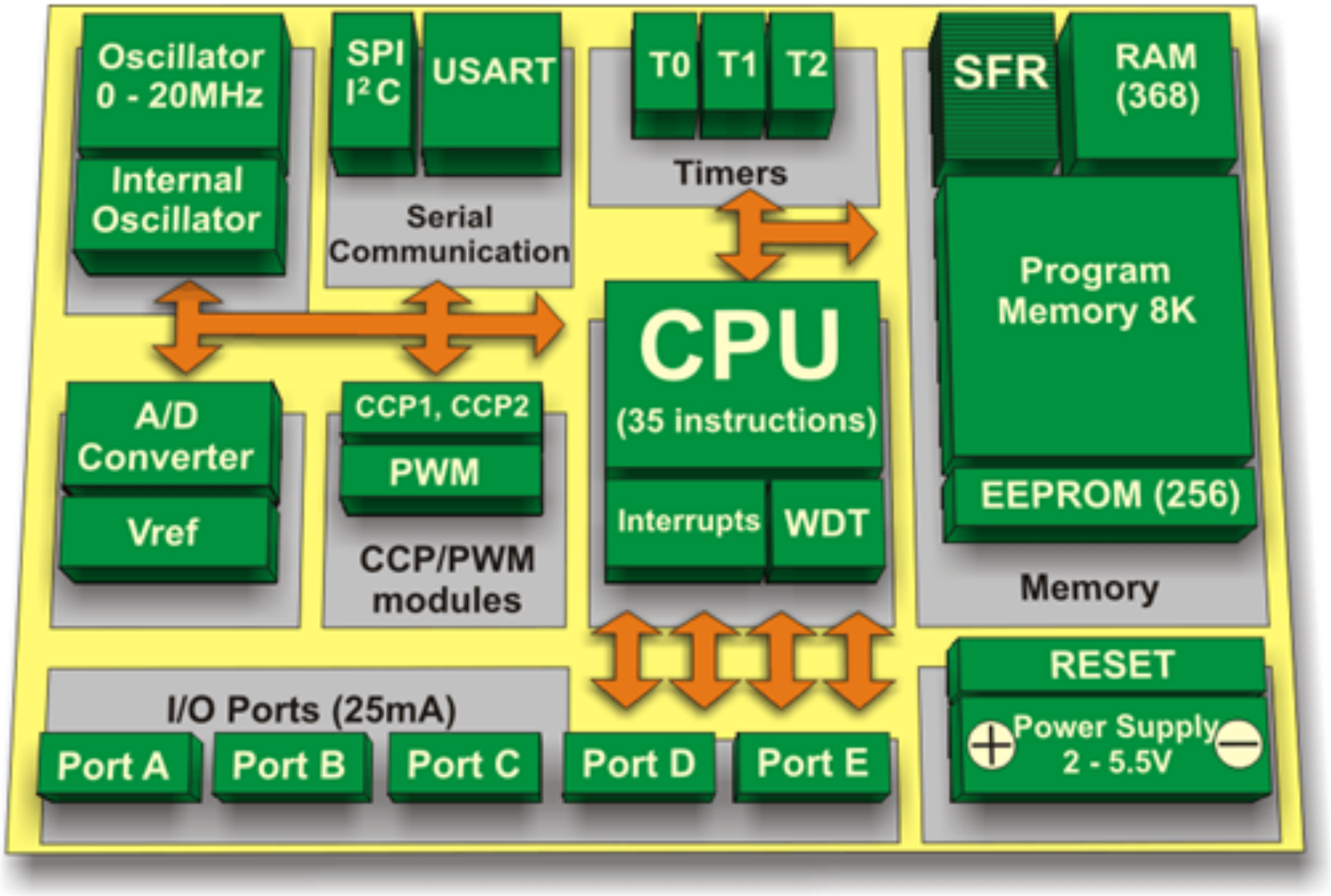
Arquitetura interna PIC 16F84 (8-bits)



Arquitetura interna PIC 16F84 (8-bits)

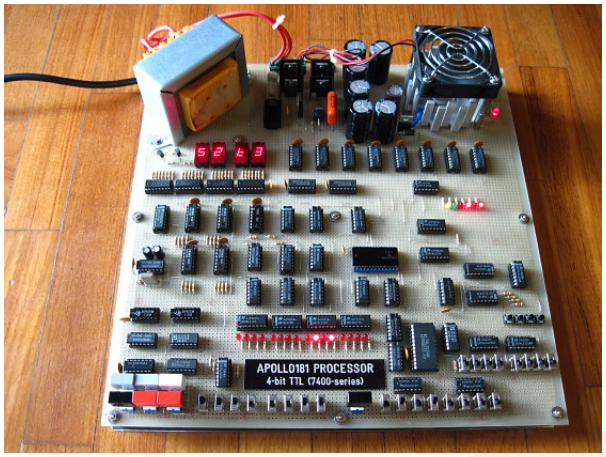


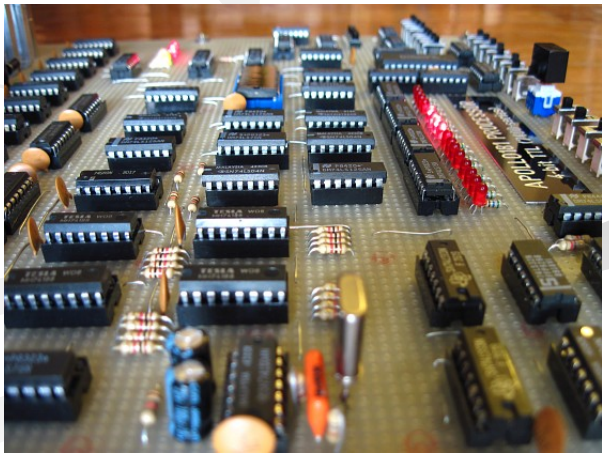
Arquitetura interna PIC 16F84 (8-bits)



Processador de 4-bits TTL:

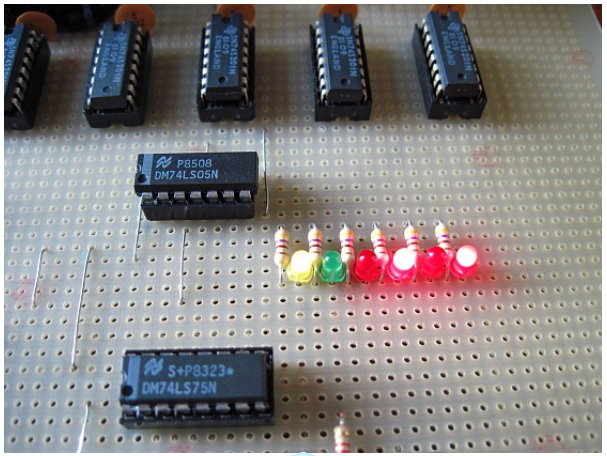
http://ygg-it.tripod.com/index.html
(Disponível em 20 Jun 2012)

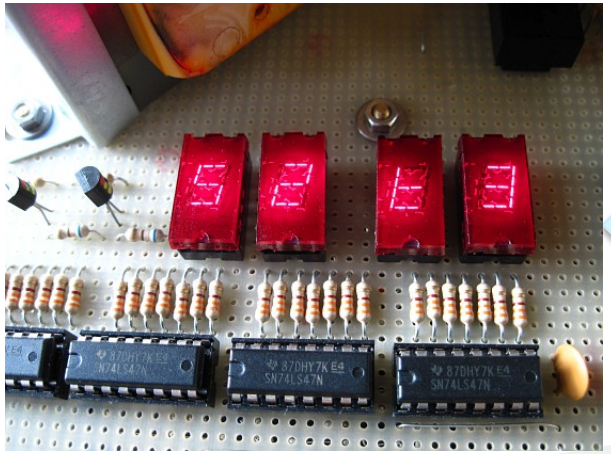




Processador de 4-bits TTL:

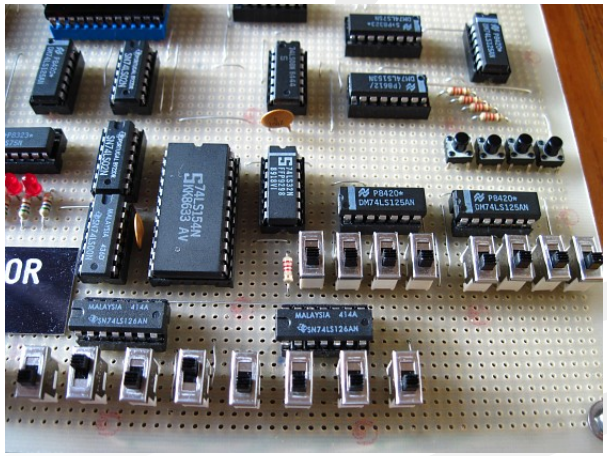
http://ygg-it.tripod.com/index.html
(Disponível em 20 Jun 2012)





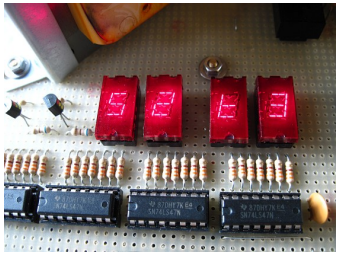
Resultados:
(Display de Saída)

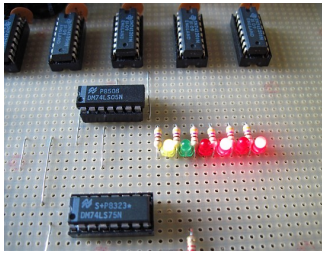
Debug Leds (Flags):
Mostra status da ULA:
Yellow=Carry,
Green=Zero,
Red=4-bit Accumulator

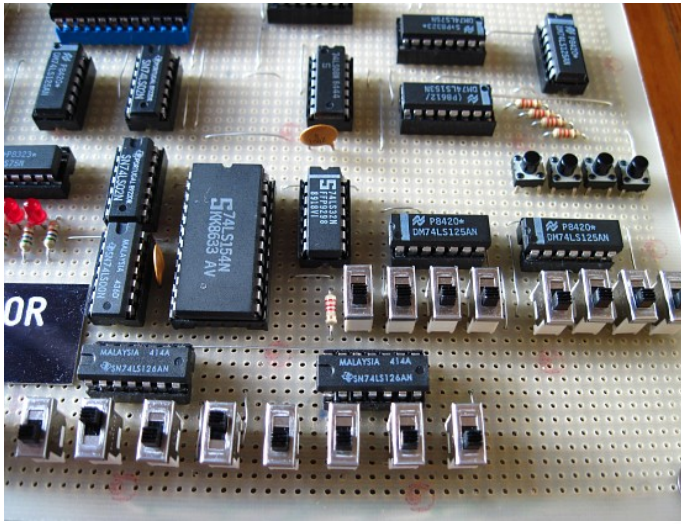


Entradas:
(Operadores +
Instruções)

Processador de 4





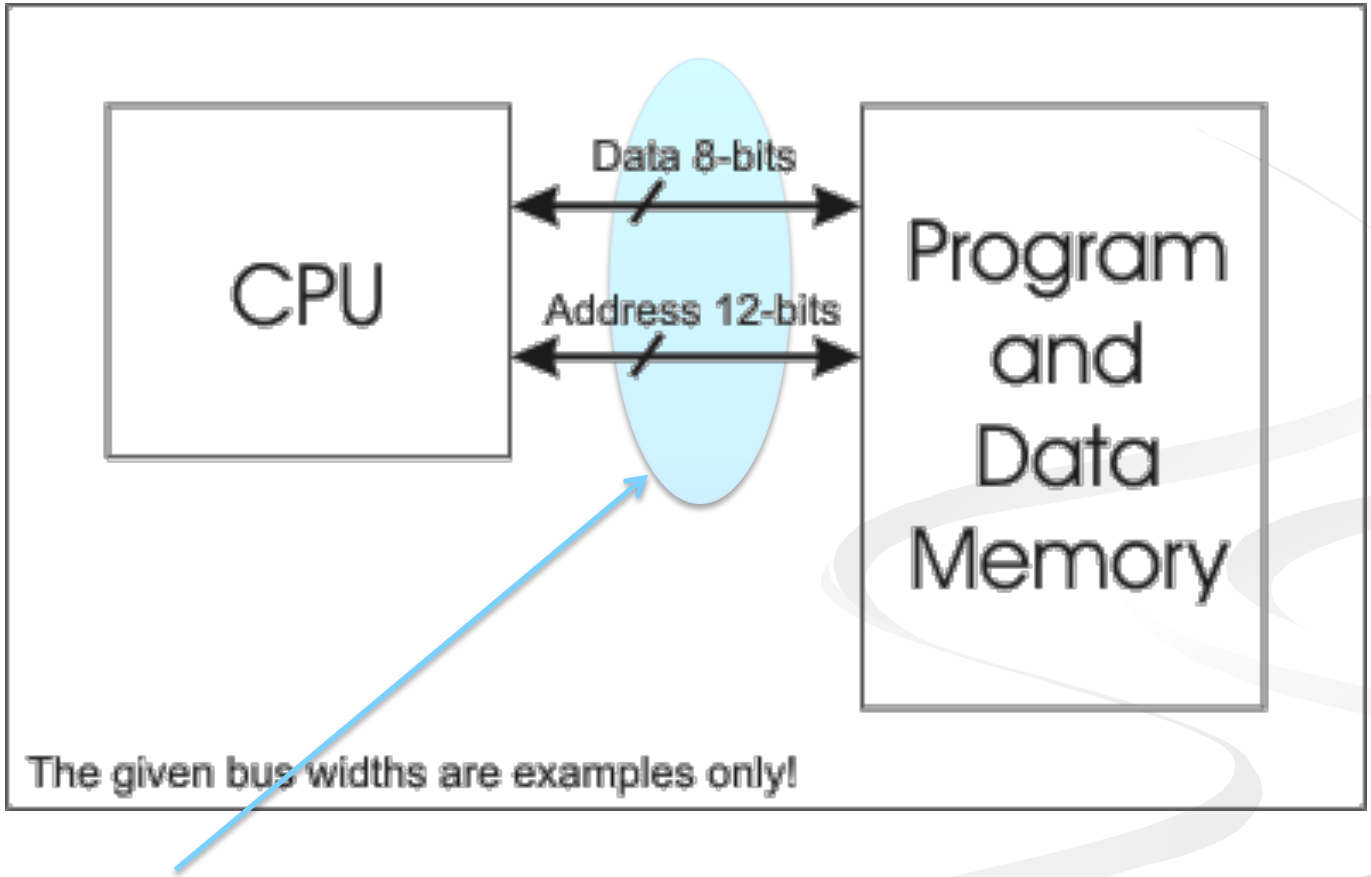


OP-CODE & OPERAND	APOLLO181 Basic Instructions	Program Command example
8r	Execute operation p in Logic Mode on operands A (ACC) and B (register r)	Example 49, **8E**: ALU s3,s2,s1,s0 = "1001" **ACC = ACC XNOR RegE**
9s	SET Carry flag to "no Carry" (s=0000) or SET Carry flag to "with Carry" (s=1111) (ACC unchanged)	Example **9F**: **SET "with carry"**
An	Compare n with ACC and SET "with Carry" if ACC>n or SET Zero flag=1 if ACC=n (ACC unchanged)	Example 19, **A8**: ACC = 9h **Compare and SET "with carry" (because 9h>8h)**
Br	Compare register r with ACC and SET "with Carry" if ACC>r or SET Zero flag=1 if ACC=r (ACC unchanged)	Example 17, 25, **B5**: ACC = 7h Reg5 = 7h **Compare and SET "Zero flag=1" (because 7h=7h)**
Cn	If Zero flag= 1 jump to location $(16 * n + ACC)_{10}$	Example 19, A9, **CF**: ACC = 9h Compare and SET "Zero flag=1" (because 9h=9h) **JUMP to F9**
Dn	If "with Carry" jump to location $(16 * n + ACC)_{10}$	Example 19, A8, 10, **DC**: ACC = 9h Compare and SET "with carry" (because 9h>8h) ACC = 0h **JUMP to C0**
En	Load ACC with Input Port(n)	Example **E2**: **ACC = PORT2**
Fn	Load Output Port(n) with ACC	Example **F8**: **PORT8 = ACC**

Arquitetura de Computadores:

Exemplos:
- ARM7

Von Neumann's → A mesma memória estoca o programa e os dados.

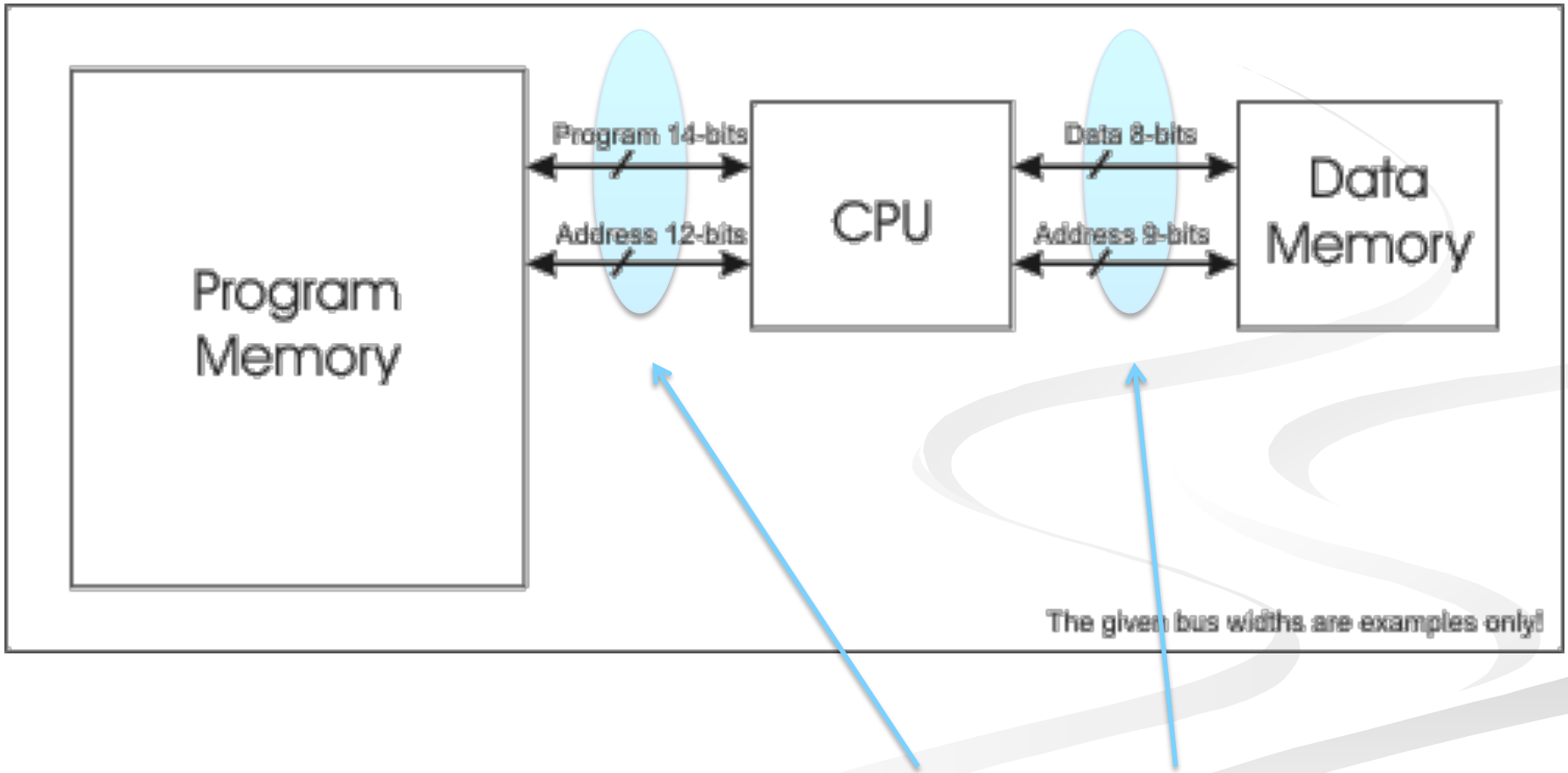


1 único barramento de dados + 1 único barramento de endereços
Eventual "gargalo" na busca e execução das instruções, limitando largura de banda!

Arquitetura de Computadores:

Exemplos:
- PIC;
- ARM9;
- ARM11;

Harward → Separa memória de dados da memória para estocar o programa.



Diferentes barramentos de dados e endereços: 1 para programa x outro para dados.

ARM Cortex M3 → RaspBerry Pi