Arquitetura de computadores e sistemas operacionais

Aritmética

Gabriel V C Candido
gabriel.candido@ifpr.edu.br

Instituto Federal do Paraná - Pinhais

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Sumário

Sumário

Adição na base 2

a	+	b	=	s	$\mathbb{N}$
0	+	0	=	0	0
0	+	1	=	1	1
1	+	0	=	1	1
1	+	1	=	?	2

Adição na base 2

vem	+	a	+	b	=	vai	s	$\mathbb{N}$
0	+	0	+	0	=	0	0	0
0	+	0	+	1	=	0	1	1
0	+	1	+	0	=	0	1	1
0	+	1	+	1	=	1	0	2
1	+	0	+	0	=	0	1	1
1	+	0	+	1	=	1	0	2
1	+	1	+	0	=	1	0	2
1	+	1	+	1	=	1	1	3

Sumário

Como representar números menores que zero?

Podemos usar um bit de sinal?

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2)

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2) $\rightarrow$ 1010 (-2)

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2) $\rightarrow$ 1010 (-2)

0000 (0)

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2) $\rightarrow$ 1010 (-2)

0000 (0) $\rightarrow$ 1000 (-0)

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2) $\rightarrow$ 1010 (-2)

0000 (0) $\rightarrow$ 1000 (-0)

Em decimal: $(+2) + (-2) = 0$

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2) $\rightarrow$ 1010 (-2)

0000 (0) $\rightarrow$ 1000 (-0)

Em decimal: $(+2) + (-2) = 0$

Mas em binário: $0010 + 1010 = 1100\ (-4)$

Como representar números menores que zero?

Podemos usar um bit de sinal?

0010 (2) $\rightarrow$ 1010 (-2)

0000 (0) $\rightarrow$ 1000 (-0)

Em decimal: $(+2) + (-2) = 0$

Mas em binário: $0010 + 1010 = 1100\ (-4)$

O circuito de soma e subtração deve ser diferente

Como representar números menores que zero?

E se tentarmos representar os números negativos de outra forma?

Como representar números menores que zero?

E se tentarmos representar os números negativos de outra forma?

$$-1 + 1 = 0 \rightarrow M + 0000.0001 = 0000.0000$$

$$\begin{array}{cccccccccc} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ + & m_7 & m_6 & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Como representar números menores que zero?

E se tentarmos representar os números negativos de outra forma?

$$-1 + 1 = 0 \rightarrow M + 0000.0001 = 0000.0000$$

$$\begin{array}{cccccccccc} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ + & m_7 & m_6 & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$-1 = 1111.1111$$

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Como representar números menores que zero?

$$
\begin{aligned}
(+A) + (-A) &= 0 \\
-A &= 0 - A \\
-A &= (1 - 1) - A \\
-A &= 1 + (-1 - A)
\end{aligned}
$$

Como representar números menores que zero?

$$
\begin{aligned}
(+A) + (-A) &= 0 \\
-A &= 0 - A \\
-A &= (1 - 1) - A \\
-A &= 1 + (-1 - A)
\end{aligned}
$$

Vamos olhar para $R = (-1 - A)$

Como representar números menores que zero?

-1		1	1	1	1	1	1	1	1
-A	-	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
R		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0

Como representar números menores que zero?

-1		1	1	1	1	1	1	1	1
-A	-	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
R		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0

Temos dois casos:

1. se $a_0 = 0$, então $r_0 = 1 - 0 = 1 = \overline{a_0}$
2. se $a_0 = 1$, então $r_0 = 1 - 1 = 0 = \overline{a_0}$

Como representar números menores que zero?

-1		1	1	1	1	1	1	1	1
-A	-	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
R		r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0

Temos dois casos:

1. se $a_0 = 0$, então $r_0 = 1 - 0 = 1 = \overline{a_0}$
2. se $a_0 = 1$, então $r_0 = 1 - 1 = 0 = \overline{a_0}$

Portanto, cada bit de R é o oposto (complemento) do bit correspondente de A, ou seja, $R = \overline{A}$

Como representar números menores que zero?

$$
\begin{aligned}
(+A) + (-A) &= 0 \\
-A &= 0 - A \\
-A &= (1 - 1) - A \\
-A &= 1 + (-1 - A) \\
-A &= 1 + R \\
-A &= 1 + \overline{A}
\end{aligned}
$$

Essa forma de representar números negativos é chamada *complemento de 2*

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Complemento de 2

$$0010(2) \rightarrow 1101 + 1 = 1110(-2)$$

Complemento de 2

$$0010(2) \rightarrow 1101 + 1 = 1110(-2)$$

O bit mais significativo k é multiplicado por -2^{k-1}

Complemento de 2

$0010(2) \rightarrow 1101 + 1 = 1110(-2)$

O bit mais significativo k é multiplicado por -2^{k-1}

O intervalo representado em k bits é
$[-2^{k-1}, 2^{k-1} - 1]$

Complemento de 2

$0010(2) \rightarrow 1101 + 1 = 1110(-2)$

O bit mais significativo k é multiplicado por -2^{k-1}

O intervalo representado em k bits é
$[-2^{k-1}, 2^{k-1} - 1]$

A soma é igual a subtração! Inclusive para o circuito!

Sumário

O que acontece ao somar números grandes?

$3 + 5$ usando 3 bits, sem considerar números negativos

O que acontece ao somar números grandes?

$3 + 5$ usando 3 bits, sem considerar números negativos

$$011 + 101 = 000$$

O que acontece ao somar números grandes?

$3 + 5$ usando 3 bits, sem considerar números negativos

$011 + 101 = 000$ Errado!

O que acontece ao somar números grandes?

$3 + 5$ usando 3 bits, sem considerar números negativos

$011 + 101 = 000$ Errado!

Não é possível representar o resultado em apenas 3 bits!

O que acontece ao somar números grandes?

$3 + 5$ usando 3 bits, sem considerar números negativos

$011 + 101 = 000$ Errado!

Não é possível representar o resultado em apenas 3 bits!

E agora considerando números negativos, em complemento de 2?

O que acontece ao somar números grandes?

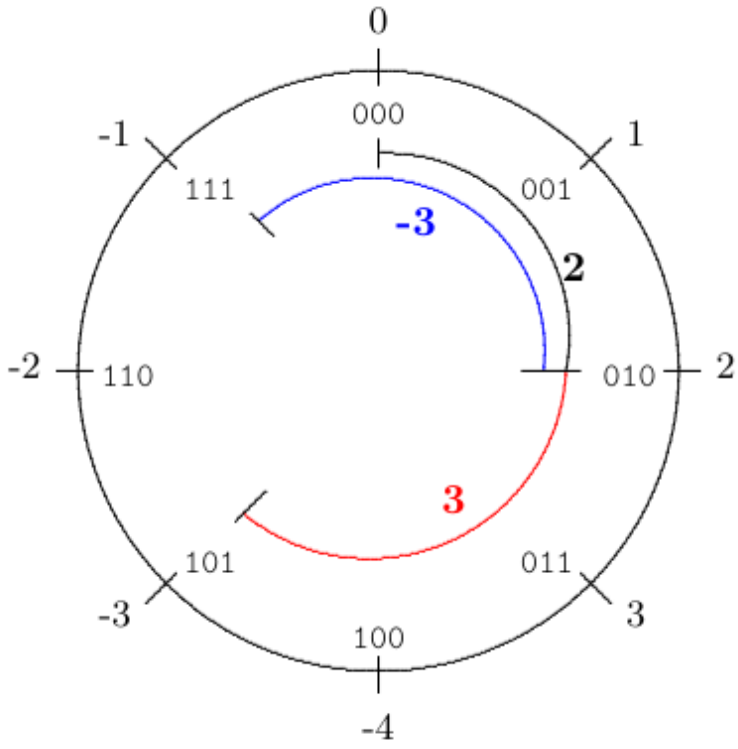

Figura: Representação em complemento de 2 e *overflow*.
Fonte: RH

Sumário

Circuito somador

vem	a	b	vai	s	$\mathbb{N}$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Circuito somador

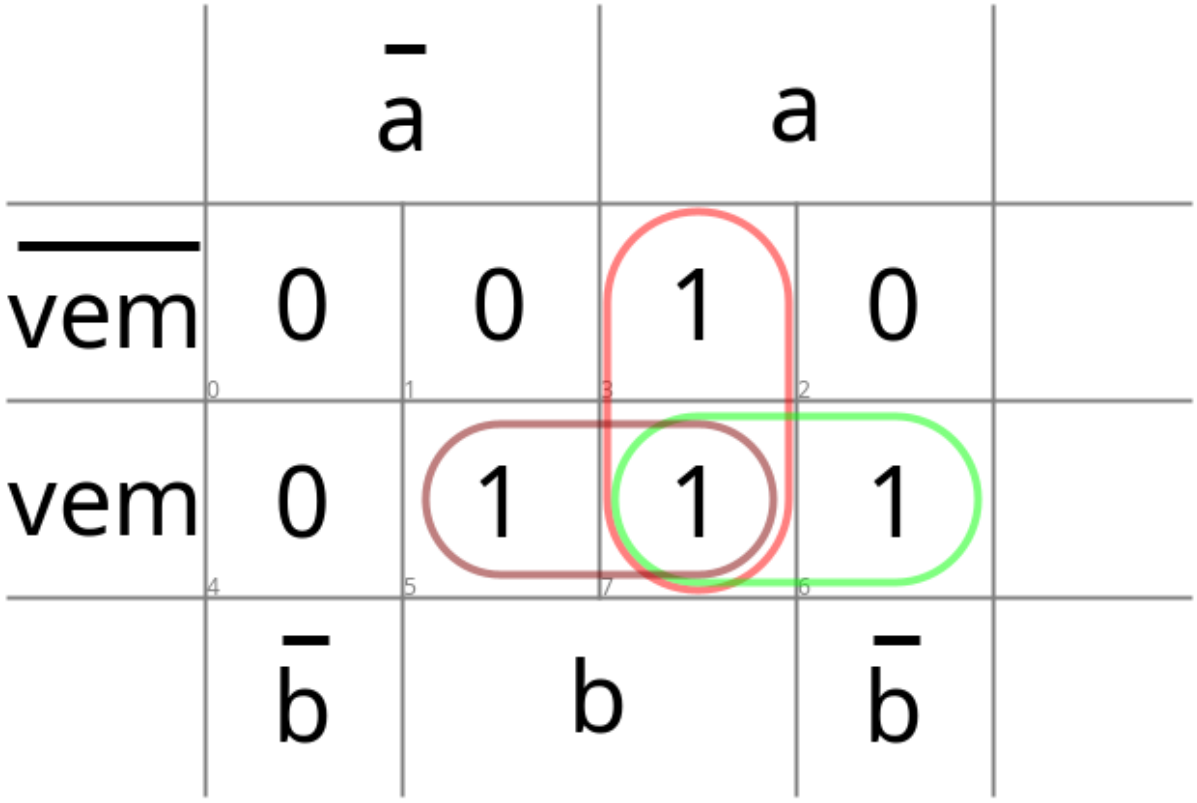

Figura: Mapa de Karnaugh do sinal *vai*.

Circuito somador

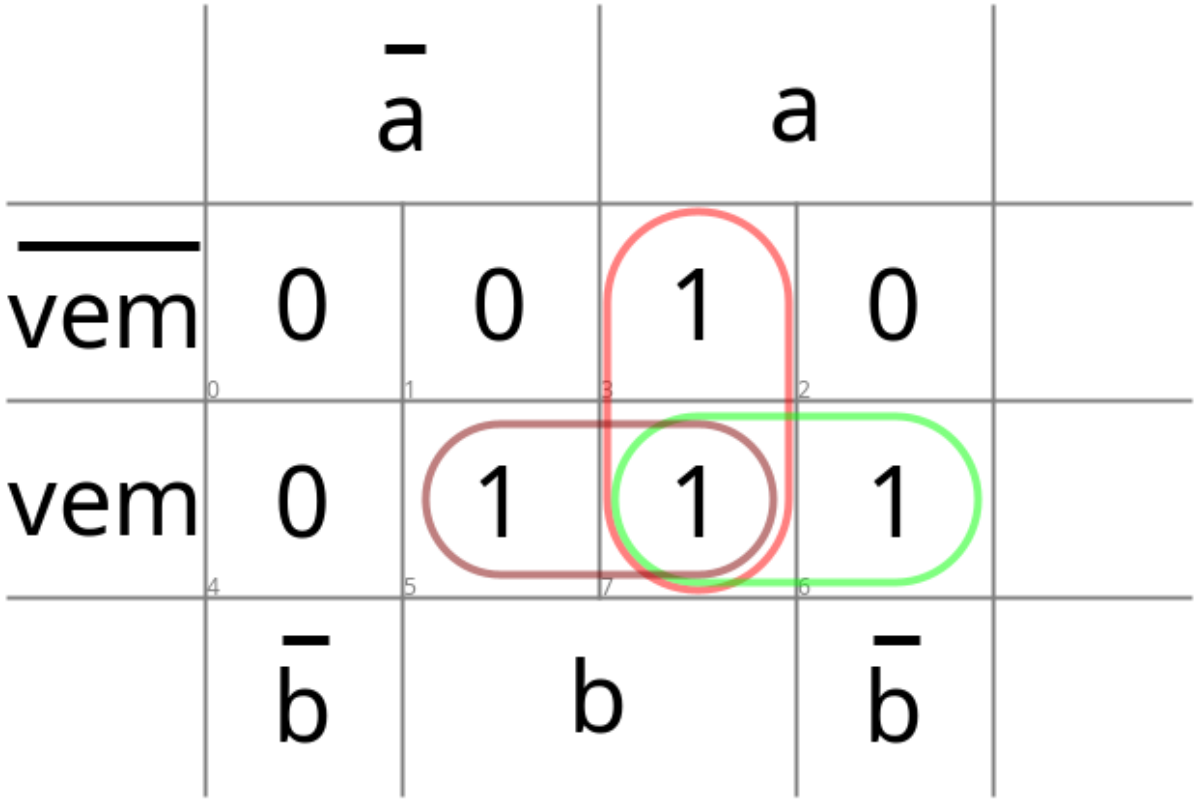

Figura: Mapa de Karnaugh do sinal *vai*.

$$\text{vai} = (a \wedge b) \vee (a \wedge \text{vem}) \vee (b \wedge \text{vem})$$

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Circuito somador

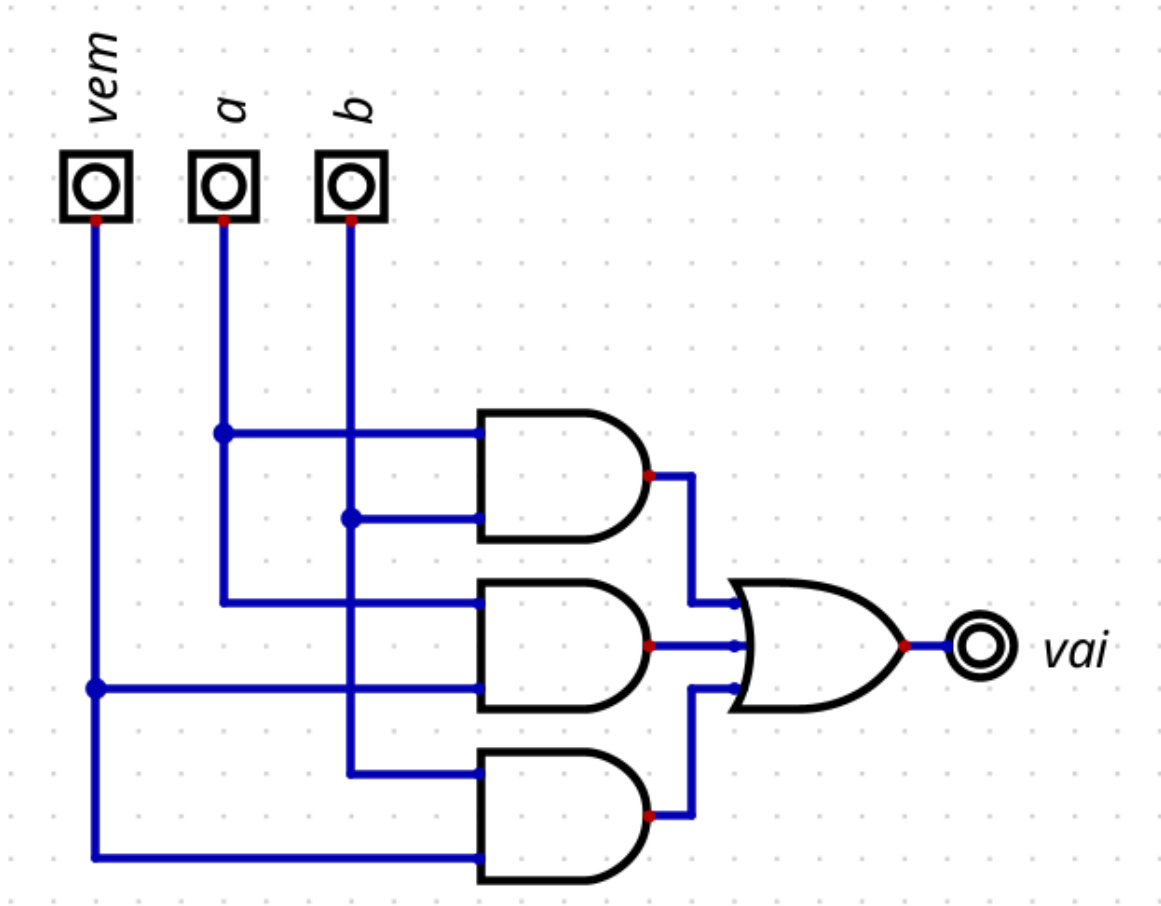

Figura: Circuito do sinal *vai* do somador de um bit.

Circuito somador

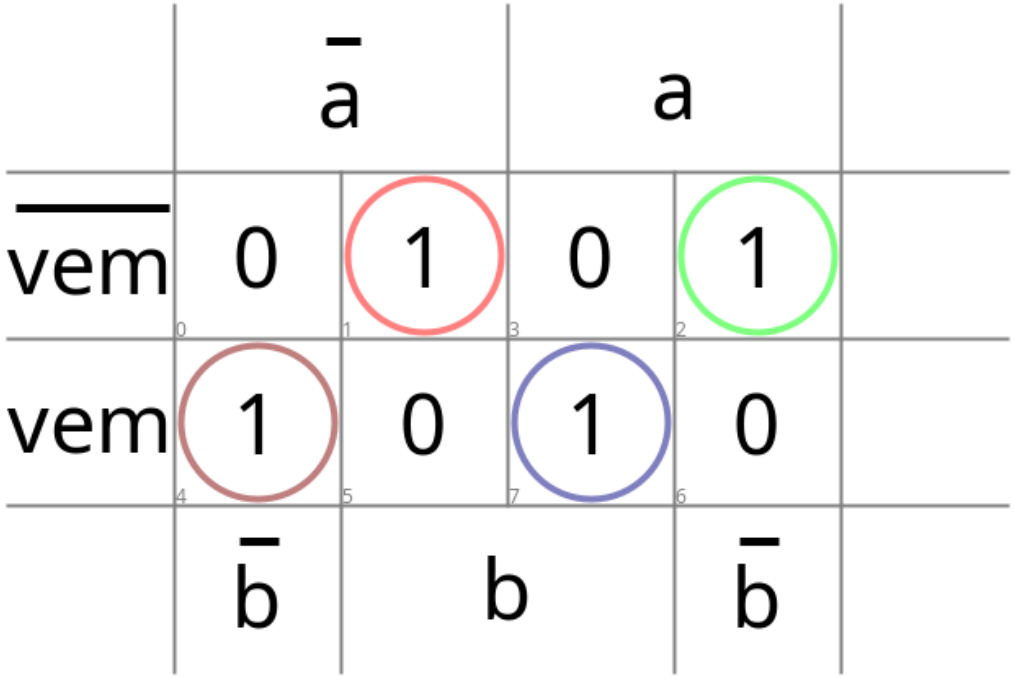

Figura: Mapa de Karnaugh do sinal *s*.

Circuito somador

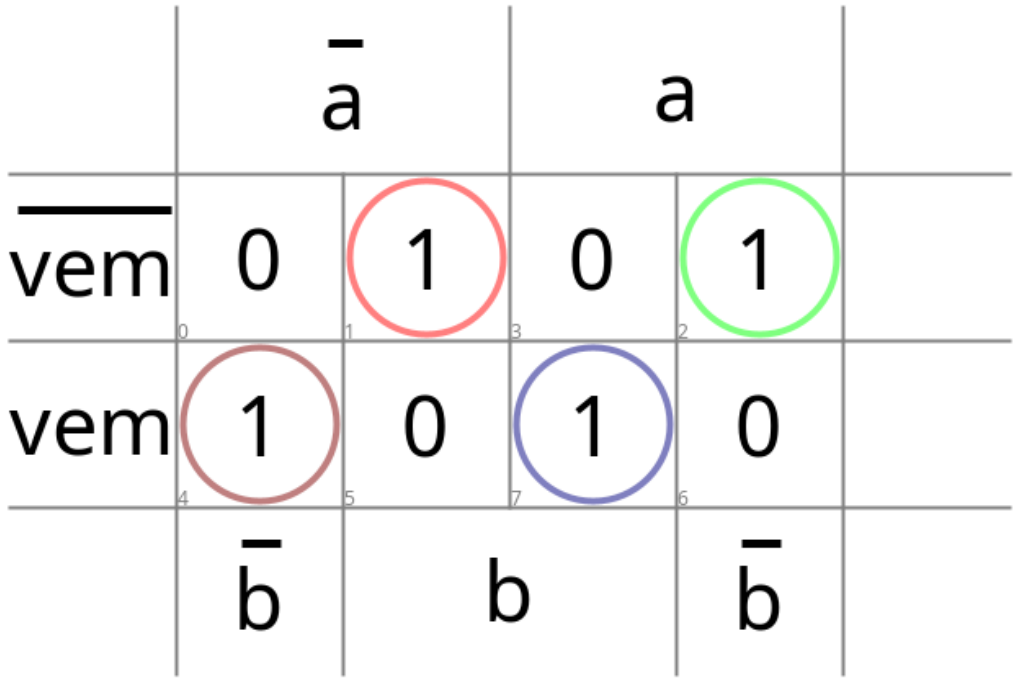

Figura: Mapa de Karnaugh do sinal s.

$$s = (\overline{a} \wedge b \wedge \overline{\text{vem}}) \vee (a \wedge \overline{b} \wedge \overline{\text{vem}}) \vee (\overline{a} \wedge \overline{b} \wedge \text{vem}) \vee (a \wedge b \wedge \text{vem})$$

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Circuito somador

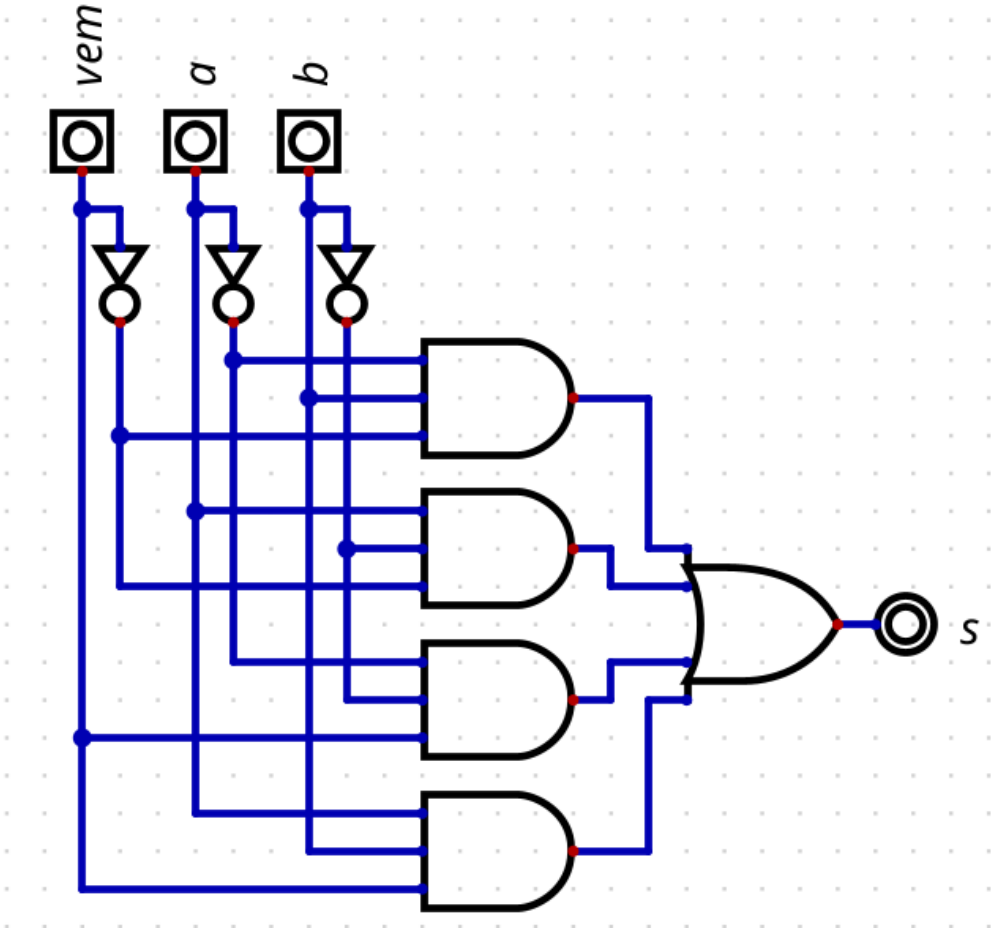

Figura: Circuito do sinal s do somador de um bit.

Circuito somador

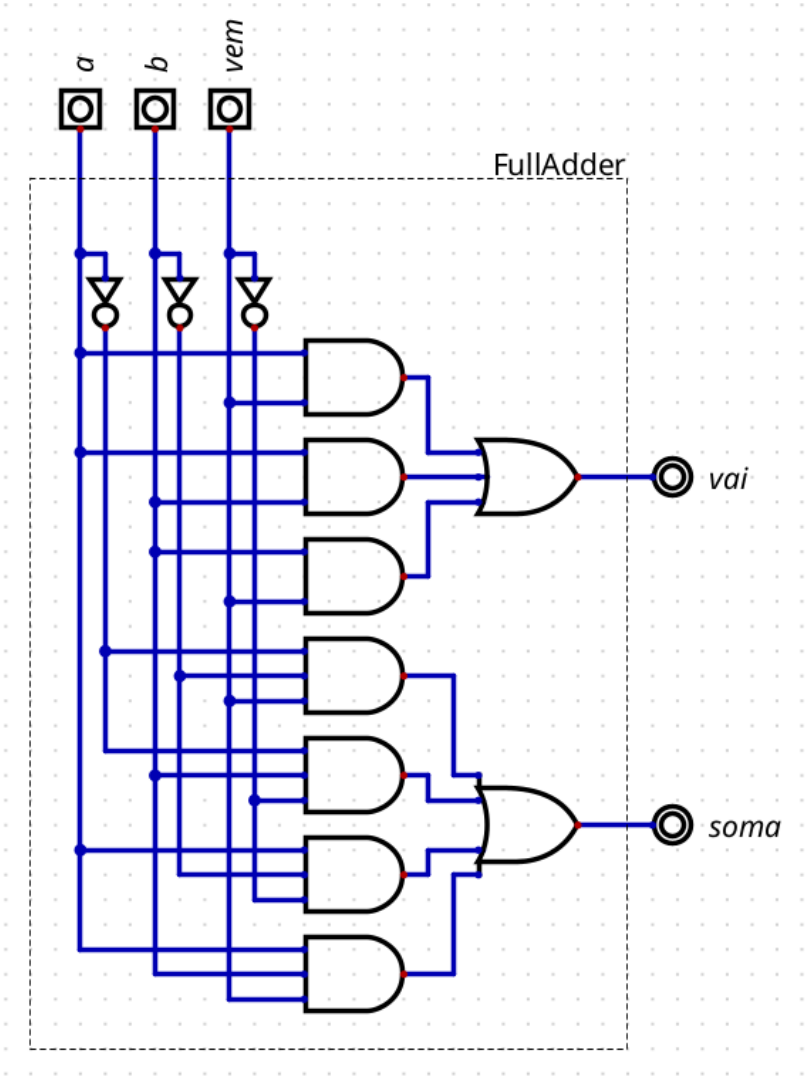

Figura: Circuito do somador de um bit.

Circuito somador

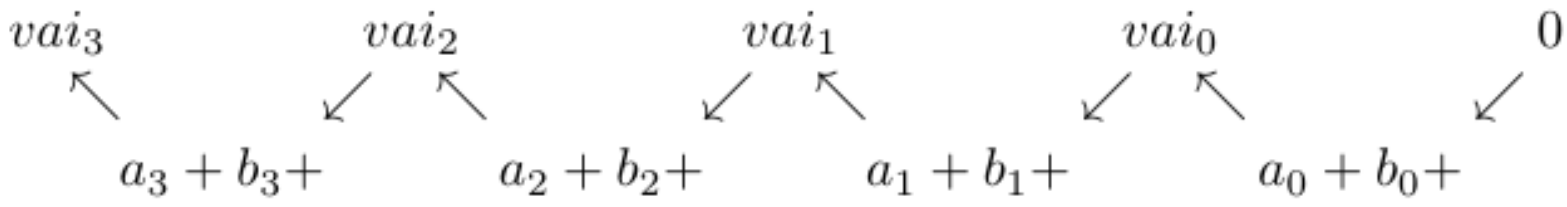

Figura: Adição de dois números de 4 bits. Fonte: RH

Circuito somador

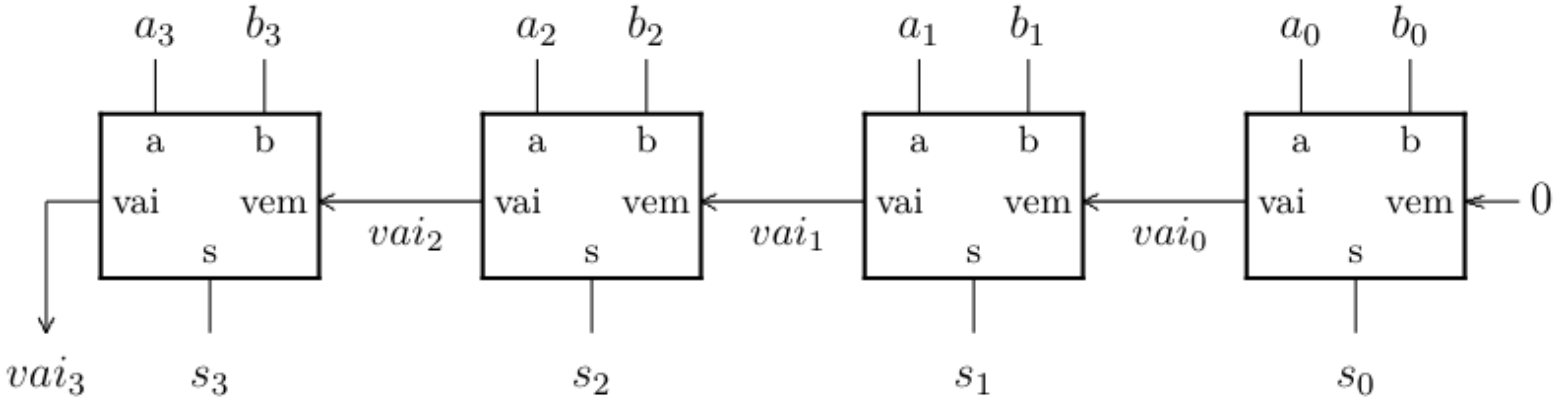

Figura: Circuito somador de 4 bits. Fonte: RH

Circuito somador e subtrator

Vimos há pouco que a subtração na verdade é uma soma envolvendo um número negativo (representado em complemento de 2)

Portanto, podemos usar o mesmo circuito da soma, com pequenas modificações para calcular o complemento de 2 de um operando antes de iniciar a soma

Circuito somador e subtrator

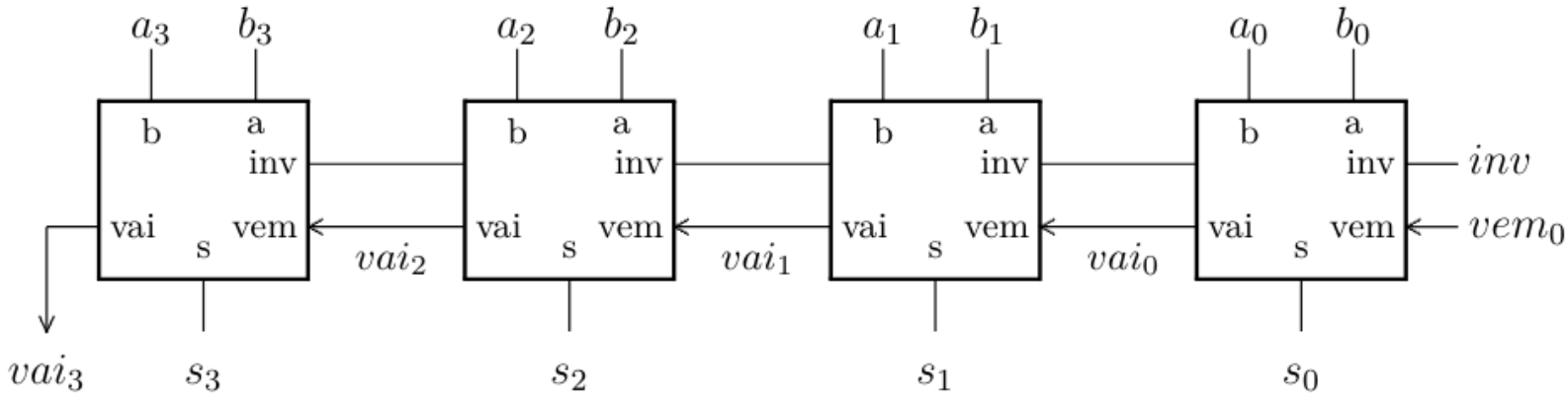

Figura: Circuito que efetua somas e subtrações. Fonte: RH

O sinal *inv* indica que cada bit de B deve ser invertido e o sinal vem_0 é ligado a 1

Circuito somador e subtrator

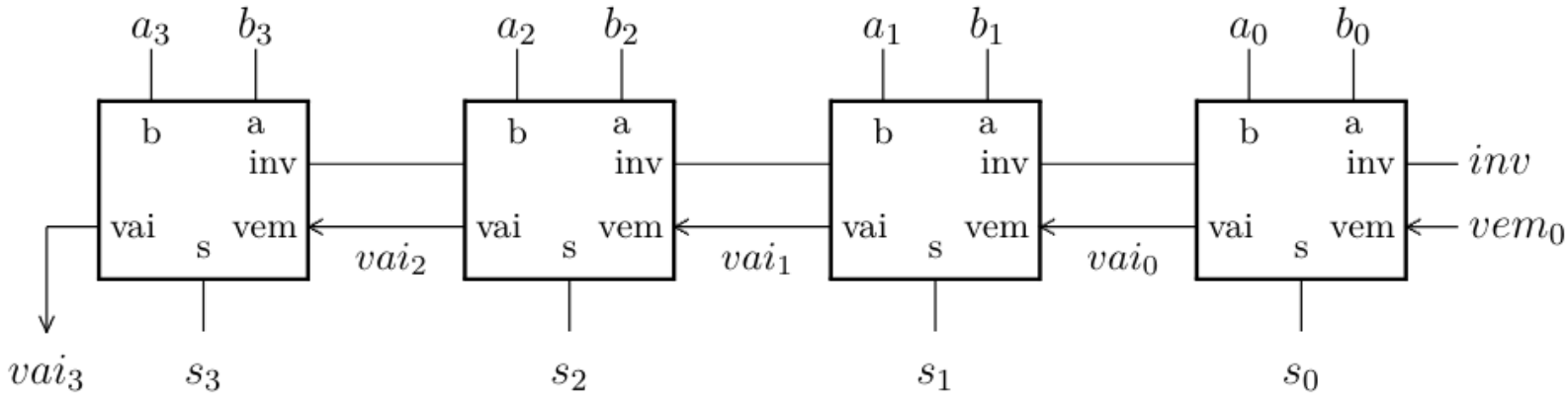

Figura: Circuito que efetua somas e subtrações. Fonte: RH

O sinal *inv* indica que cada bit de B deve ser invertido e o sinal vem_0 é ligado a $1 \rightarrow \overline{B} + 1$

Deslocamentos

Deslocamentos para esquerda ou para direita em uma tupla de bits são equivalentes a multiplicação ou divisão inteira por potências de 2

00110			seis
$00110 >> 1$	$=$	00011	três
$00110 << 1$	$=$	01100	doze
$00110 >> 2$	$=$	00001	um
$00110 << 2$	$=$	11000	vinte e quatro

Note que, se usarmos números negativos em complemento de 2, um deslocamento pode causar problemas

Deslocamentos

00110			seis
00110 >> 2	=	00001	um
00110 << 2	=	11000	-8 (errado)
11100			-4
11100 >> 2	=	00111	sete (errado)
11100 >> 2	=	11111	-1 (certo)

O deslocamento para direita deve replicar o bit de sinal!

Deslocamentos

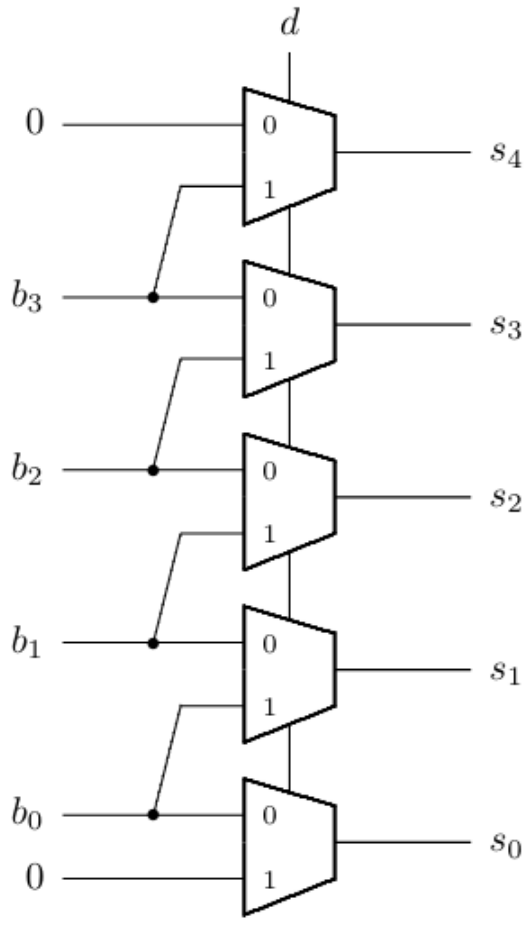

Figura: Deslocamento de uma posição para esquerda. Fonte: RH

Deslocamentos

Podemos combinar deslocadores

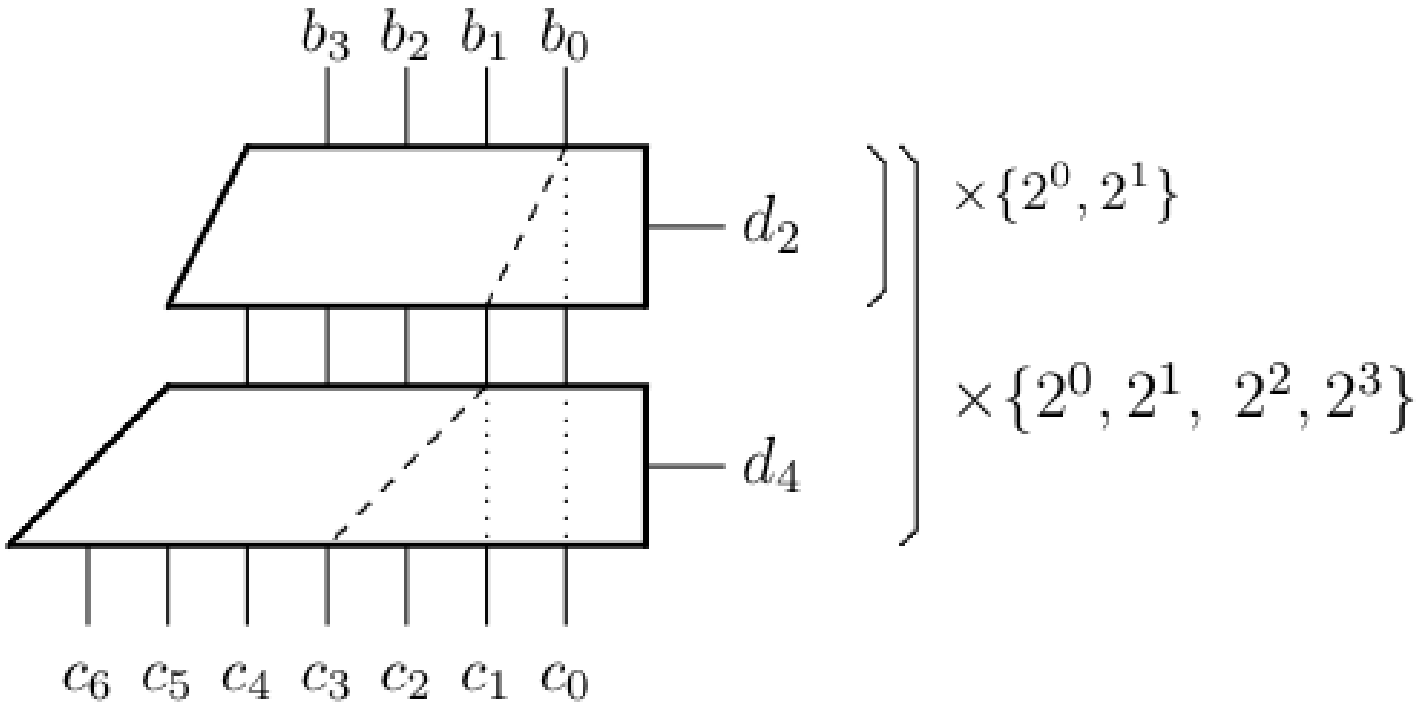

Figura: Deslocamento de 4 bits e três posições. Fonte: RH

Sumário

Operações aritméticas

Operações de soma, subtração, multiplicação, divisão, comparação de igualdade, comparação de magnitude são ditas *operações de aritmética*

Operações aritméticas

Operações de soma, subtração, multiplicação, divisão, comparação de igualdade, comparação de magnitude são ditas *operações de aritmética*

```
x = x + 1
```

```
if (a == b) ...
```

```
r1 = -b/(4 * a) + sqrt(b * b - 4 * a * c);
```

Operações lógicas

Operações como *and*, *or*, *xor*, *not* são chamadas *operações de lógica*

Operações lógicas

Operações como *and*, *or*, *xor*, *not* são chamadas *operações de lógica*

```
x = x | 1
```

```
if (a && b) ...
```

ULA

A ULA é um circuito que efetua *todas* as operações ao mesmo tempo, a depender do comando em C/C++, escolhe uma dentre as saídas $\rightarrow$ mais fácil e barato para construir o circuito

ULA

A ULA é um circuito que efetua *todas* as operações ao mesmo tempo, a depender do comando em C/C++, escolhe uma dentre as saídas $\rightarrow$ mais fácil e barato para construir o circuito

Comandos em C/C++ são traduzidos pelo compilador para *linguagem de máquina* que possui uma sequência de *instruções* simples para executar no processador

ULA

`x = x + 1` é traduzido para a instrução
`ADD x, x, 1`

A ULA vai executar todas as instruções, mas vai escolher a saída do circuito somador como resultado final

ULA

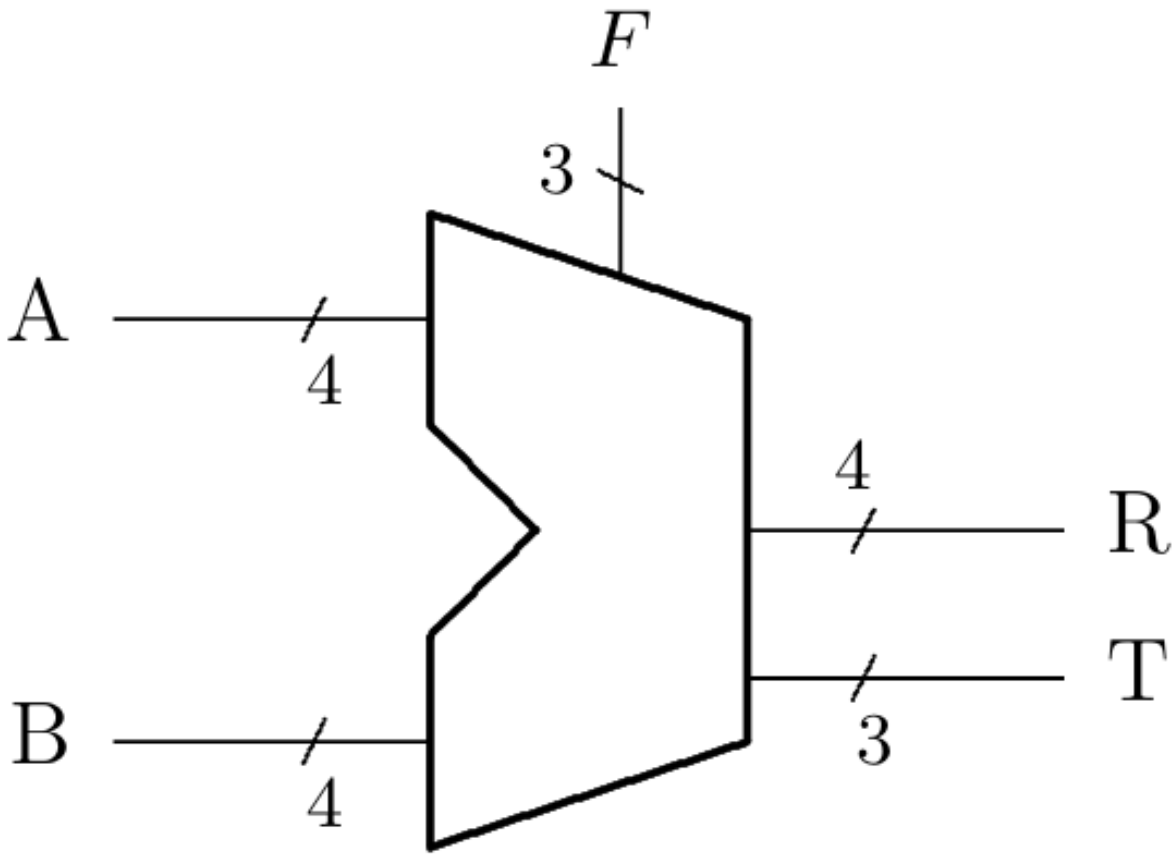

Figura: Unidade de lógica e aritmética. Fonte: RH

ULA

Essa ULA possui operandos e um resultado de 4 bits. A ULA recebe um sinal F de 3 bits para indicar qual a operação desejada. Além disso, a ULA gera um sinal T de 3 bits contendo o *status* da operação (negativo, zero, vai-um)

ULA

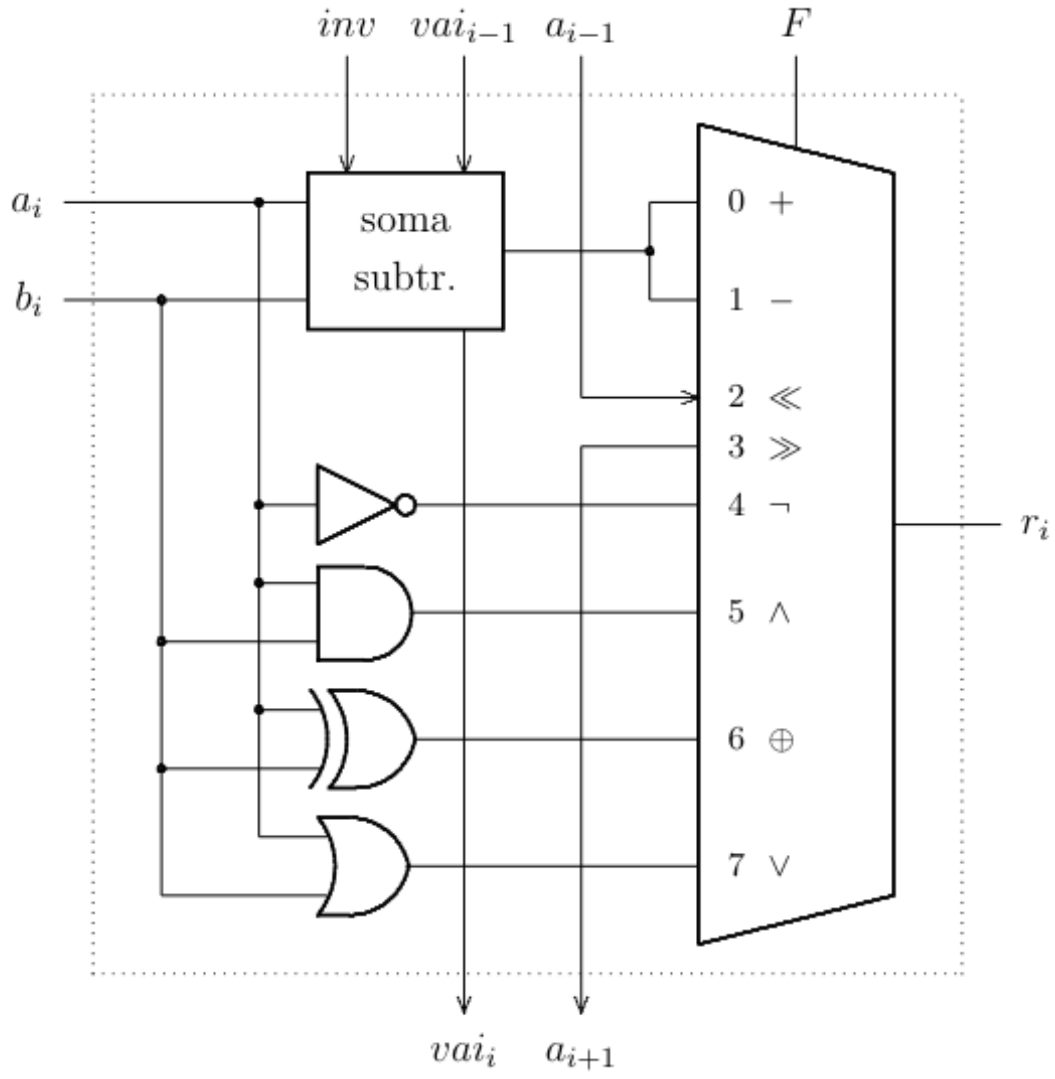

Figura: Fatia de um bit da ULA. Fonte: RH