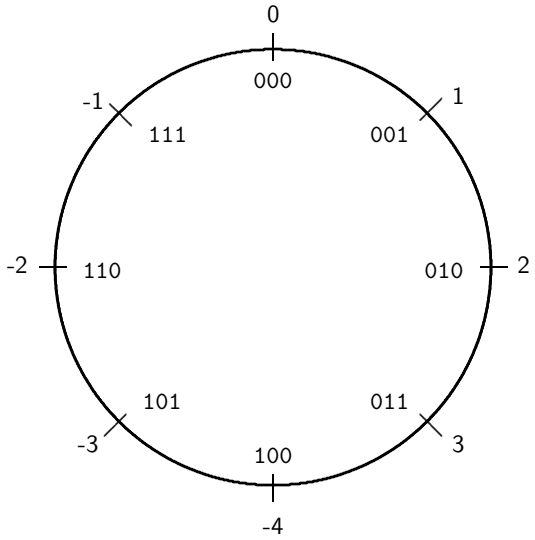
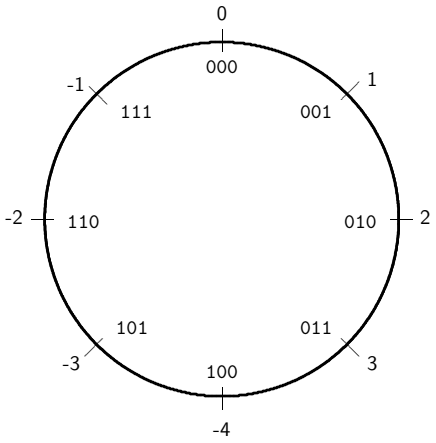
Representação em Ponto Fixo – Complemento de 2



Representação em Ponto Fixo (cont)

Representação em complemento de dois
círculo da representação + overflow

$$
\begin{aligned}
x + \overline{x} &= -1 \\
x + \overline{x} + 1 &= 0 \\
\overline{x} + 1 &= -x
\end{aligned}
$$



Representação em Ponto Fixo (cont.)

$$
\begin{aligned}
(x_{31} \cdot -2^{31}) &+ [\, x_{30} \cdot 2^{30} + \ldots + x_0 \cdot 2^0 \,] \\
(0 \cdot -2^{31}) &+ [\ldots] \quad \rightarrow 0001 \\
(1 \cdot -2^{31}) &+ [\ldots] \quad \rightarrow 1111
\end{aligned}
$$

$$
\begin{aligned}
(0 \cdot -2^2) + 1 \cdot 2^1 + 1 \cdot 2^0 &= (x_2 \cdot -2^2) + x_1 \cdot 2^1 + x_0 \cdot 2^0 \\
0 + 2 + 1 = 3 &= (0 \cdot -2^2) + 1 \cdot 2^1 + 0 \cdot 2^0
\end{aligned}
$$

$$
\begin{aligned}
(1 \cdot -2^2) + 1 \cdot 2^1 + 1 \cdot 2^0 &= (x_2 \cdot -2^2) + x_1 \cdot 2^1 + x_0 \cdot 2^0 \\
-4 + 2 + 1 = -1 &= (x_2 \cdot -2^2) + x_1 \cdot 2^1 + x_0 \cdot 2^0
\end{aligned}
$$

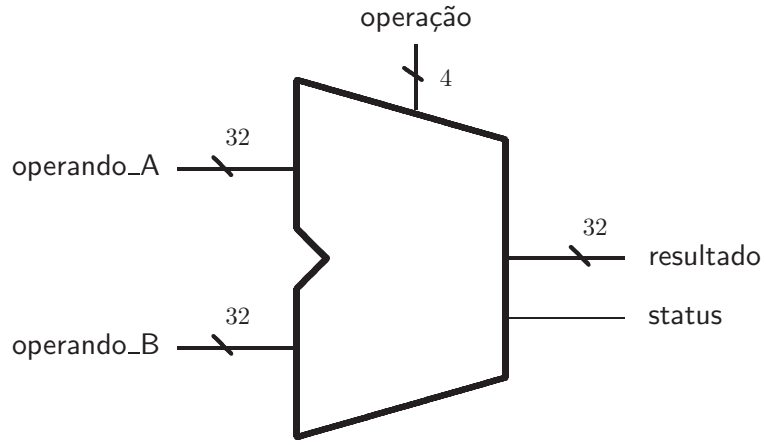
Representação em Ponto Fixo (cont.)

$$
\begin{aligned}
(x_{31} \cdot -2^{31}) &+ [\, x_{30} \cdot 2^{30} + \ldots + x_0 \cdot 2^0 \,] \\
(0 \cdot -2^{31}) &+ [\ldots] \quad \rightarrow 0\,001 \\
(1 \cdot -2^{31}) &+ [\ldots] \quad \rightarrow 1\,111
\end{aligned}
$$

	–128	64	32	16	8	4	2	1
$+1 = 0 + 1$	0	0	0	0	0	0	0	1
$-1 = -128 + 127$	1	1	1	1	1	1	1	1
$-125 = -128 + 3$	1	0	0	0	0	0	1	1
$-3 = -128 + 125$	1	1	1	1	1	1	0	1

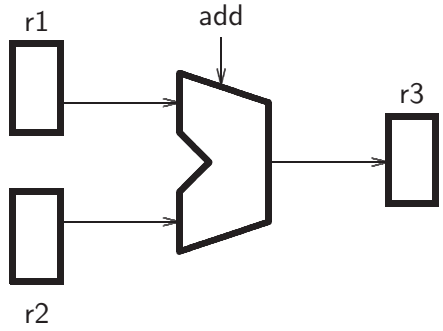
Unidade de Lógica e Aritmética

Operações lógicas: E, OU, OU-EXCL, INV
Operações aritméticas: $+$, $-$, $*$, $\div$, $=$, $<$



ULA + registradores

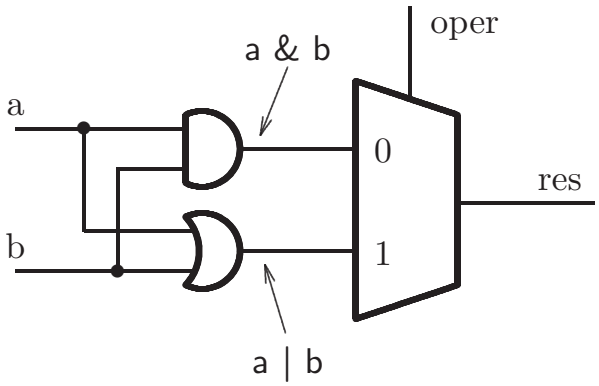
```
add r3,r1,r2 # r3 ← r1+r2
```



busca instrução;
decodifica, acessa regs;
executa;
grava resultado;

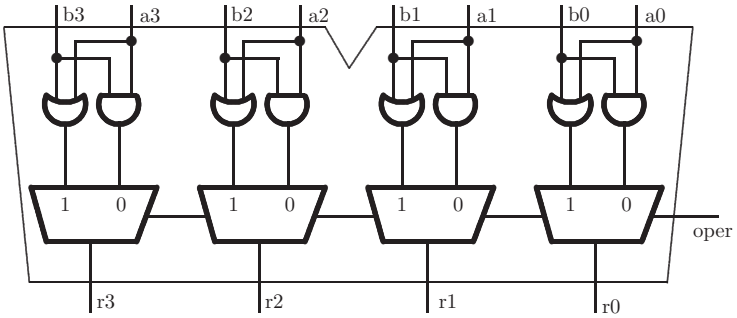
Operações lógicas: E, OU

Circuito para um bit



ULA de 32 bits

Para contruir uma ULA de 32 bits
contrói-se uma de 1 bit e usa-se 32 delas (ou quatro...)



Somador Completo de Um Bit

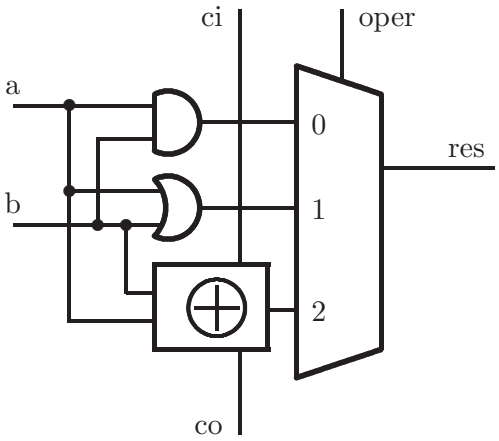
SOMA

a	b	ci	co	s
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

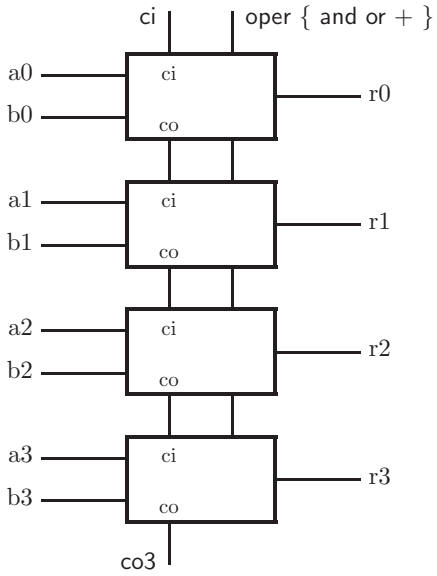
$$s = a \oplus b \oplus ci$$

$$co = (\overline{a} \wedge b \wedge ci) \vee (a \wedge \overline{b} \wedge ci) \vee (a \wedge b)$$

Somador Completo II



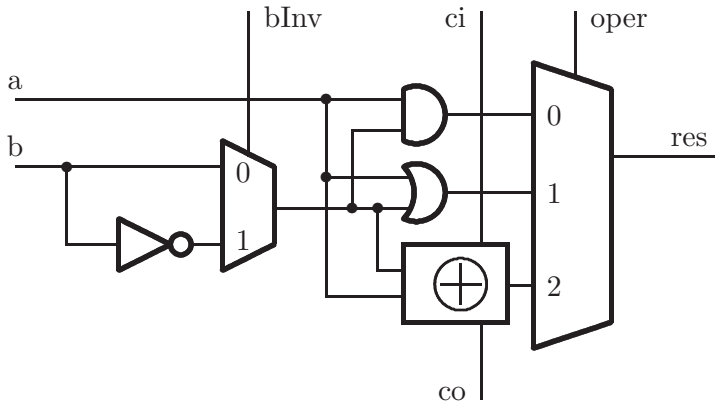
ULA de 4 bits



Subtração e Negação

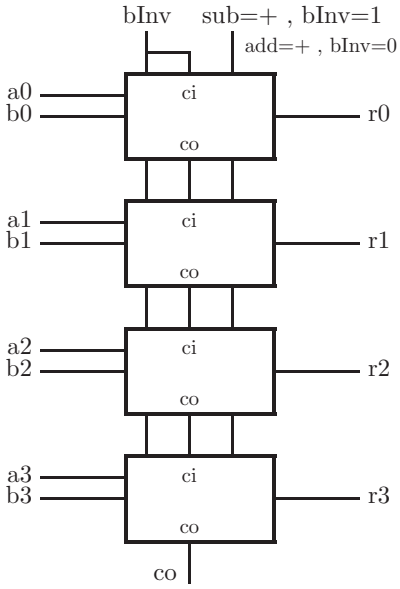
subtração

$$\boldsymbol{a - b} \;\Leftrightarrow\; \boldsymbol{a + (-b)} \;\Leftrightarrow\; \boldsymbol{a + (\overline{b} + 1)} \;\Leftrightarrow\; \boldsymbol{a + \overline{b} + 1}$$



Subtração II

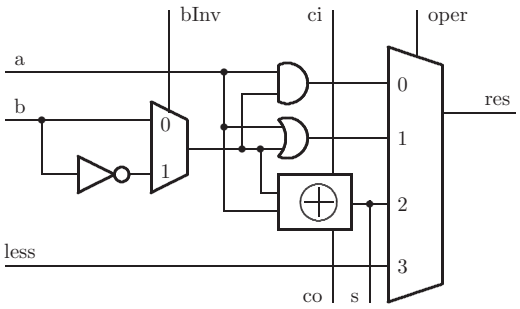
$$a - b \Leftrightarrow$$
$$a + (-b) \Leftrightarrow$$
$$a + (\overline{b} + 1) \Leftrightarrow$$
$$a + \overline{b} + 1$$



set-on-less-than

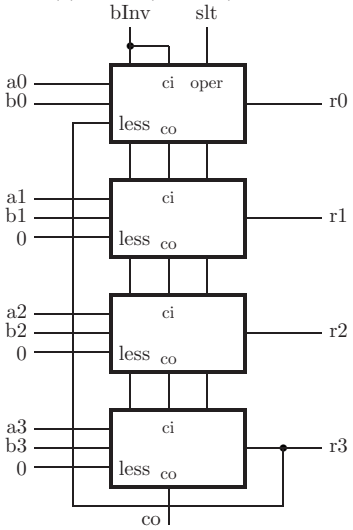
`slt rd, rs, rt` $\equiv$
`rd` $\leftarrow$ `(rs` $<$ `rt)`
rd recebe 1 se (rs $<$ rt)

$$\begin{aligned} & (rs - rt) < 0 \\ \Rightarrow\ & (rs - rt) + rt < (0 + rt) \\ \Rightarrow\ & (rs + 0) < (0 + rt) \\ \Leftrightarrow\ & rs < rt \end{aligned}$$



set-on-les-than II

$$\texttt{r} \leftarrow (a < b) \ \equiv\ \texttt{r} \leftarrow ((a - b) < 0)$$



Igualdade

$a = b \;\Rightarrow\; a - b = 0$

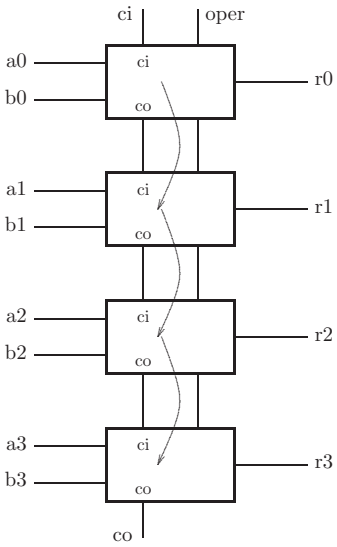
zero $= \overline{r0} \wedge \overline{r1} \wedge \overline{r2} \wedge \overline{r3}$

Melhora no somador

Esta ULA de 4 bits é MUITO lenta

o ci_n somente é conhecido após todos os ci_m, $m < n$ serem computados, um após o outro

Qual o número de portas lógicas entre 'ci' e 'co'?



Primeira solução

antecipar cálculo do ci em função dos bits menos significativos

$$
\begin{aligned}
ci_1 &= (b_0 \wedge ci_0) \vee (a_0 \wedge ci_0) \vee (a_0 \wedge b_0) \\
ci_2 &= (b_1 \wedge ci_1) \vee (a_1 \wedge ci_1) \vee (a_1 \wedge b_1) \\
&= (b_1 \wedge a_0 \wedge b_0) \vee (b_1 \wedge a_0 \wedge ci_0) \vee (b_1 \wedge b_0 \wedge ci_0) \vee \\
&\quad (a_1 \wedge a_0 \wedge b_0) \vee (a_1 \wedge a_0 \wedge ci_0) \vee (a_1 \wedge b_0 \wedge ci_0) \vee \\
&\quad (a_1 \wedge b_1) \\
ci_3 &= \cdots \qquad \textbf{muito complicado}
\end{aligned}
$$

Segunda solução

Antecipar cálculo do ci em função dos bits menos significativos

a	b	ci	co	s
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

$$g_i = a_i \wedge b_i \qquad \textit{gera vai-um}$$
$$p_i = a_i \vee b_i \qquad \textit{propaga vem-um}$$

Segunda solução II

Antecipar cálculo do ci em função dos bits menos significativos

$$g_i = a_i \wedge b_i \qquad \textit{gera vai-um}$$
$$p_i = a_i \vee b_i \qquad \textit{propaga vai-um}$$

$$\begin{aligned}
ci_1 &= g_0 \vee (p_0 \wedge ci_0) \\
ci_2 &= g_1 \vee (p_1 \wedge g_0) \vee (p_1 \wedge p_0 \wedge ci_0) \\
ci_3 &= g_2 \vee (p_2 \wedge g_1) \vee (p_2 \wedge p_1 \wedge g_0) \vee (p_2 \wedge p_1 \wedge p_0 \wedge ci_0) \\
ci_4 &= g_3 \vee (p_3 \wedge g_2) \vee (p_3 \wedge p_2 \wedge g_1) \vee (p_3 \wedge p_2 \wedge p_1 \wedge g_0) \\
&\quad \vee (p_3 \wedge p_2 \wedge p_1 \wedge p_0 \wedge ci_0)
\end{aligned}$$

Adiantamento de vai-um

Este circuito se chama "carry lookahead"
calcula os quatro ci's em paralelo
agrupa-se quatro grupos de quatro para somador de 16 bits
agrupa-se quatro grupos de dezesseis para somador de 64 bits