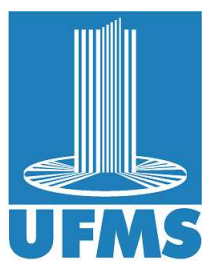
Projeto da ULA

Arquitetura de Computadores I
Bacharelado em Sistemas de Informação

Professor Rodrigo Kishi
rodrigo.kishi@ufms.br



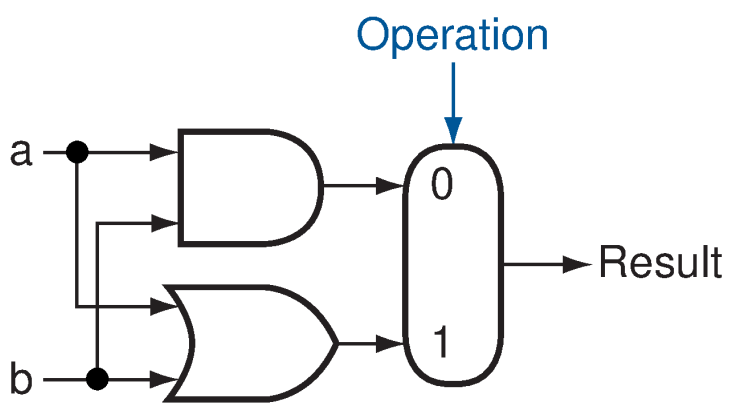
Campus de Três Lagoas - CPTL
Universidade Federal de Mato Grosso do Sul

Conteúdo

- Unidade lógica de 1 bit (AND e OR);
- Somador de 1 bit;
- ULA de 1 bit (AND, OR e Soma);
- ULA de 32 bits (AND, OR e Soma);
- ULA de 1 bit (AND, OR e Soma e Subtração);
- ULA de 1 bit (AND, OR e Soma, Subtração e `slt`);
- ULA de 1 bit (AND, OR e Soma, Subtração, `slt` e igualdade);
- ULA de 32 bits (AND, OR, Soma, Subtração, `slt` e igualdade).

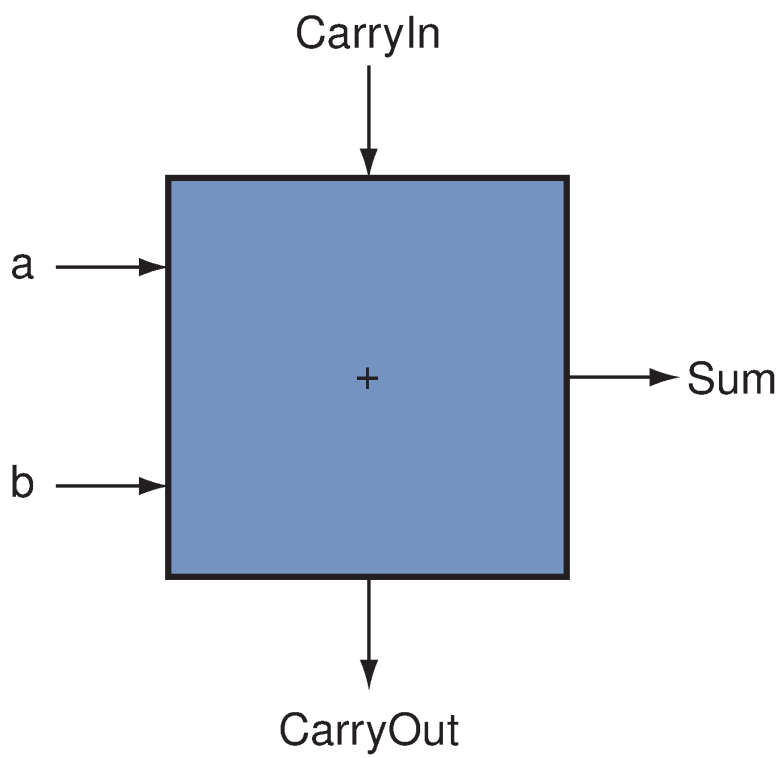
Unidade lógica de 1 bit (AND e OR)

- Entradas:
  - a e b: dois dados de 1 bit a serem operados;
  - $Operation$: sinal de controle (1 bit);
- Saída:
  - $Result$: dado de 1 bit resultante da operação.
- Multiplexador de 2 entradas de 1 bit (seleciona resultado);
  - Se $Operation = 0 \rightarrow Result = a$ AND b
  - Se $Operation = 1 \rightarrow Result = a$ OR b



Somador de 1 bit

- Entradas:
  - a e b: dois dados de 1 bit a serem somados;
  - $CarryIn$: sinal de vai um “anterior” ;
- Saídas:
  - Sum: dado de 1 bit resultante da soma;
  - $CarryOut$: sinal de vai um.



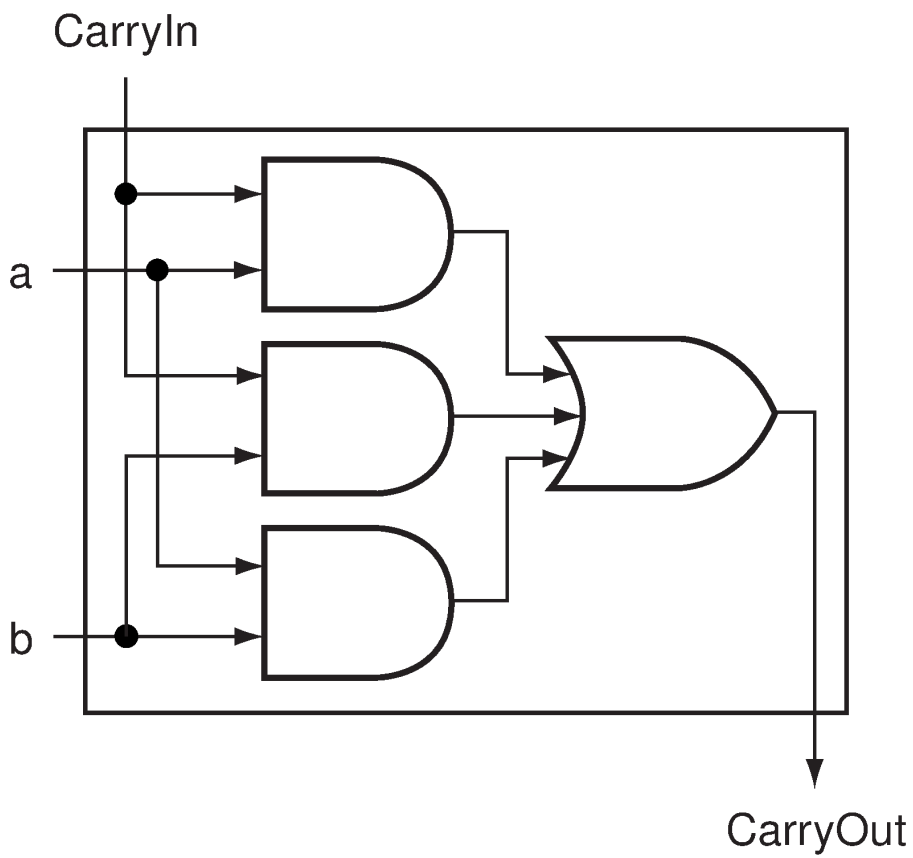
Somador de 1 bit

- Tabela verdade do somador de 1 bit:

a	*b*	*CarryIn*	*CarryOut*	*Soma*
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

Somador de 1 bit

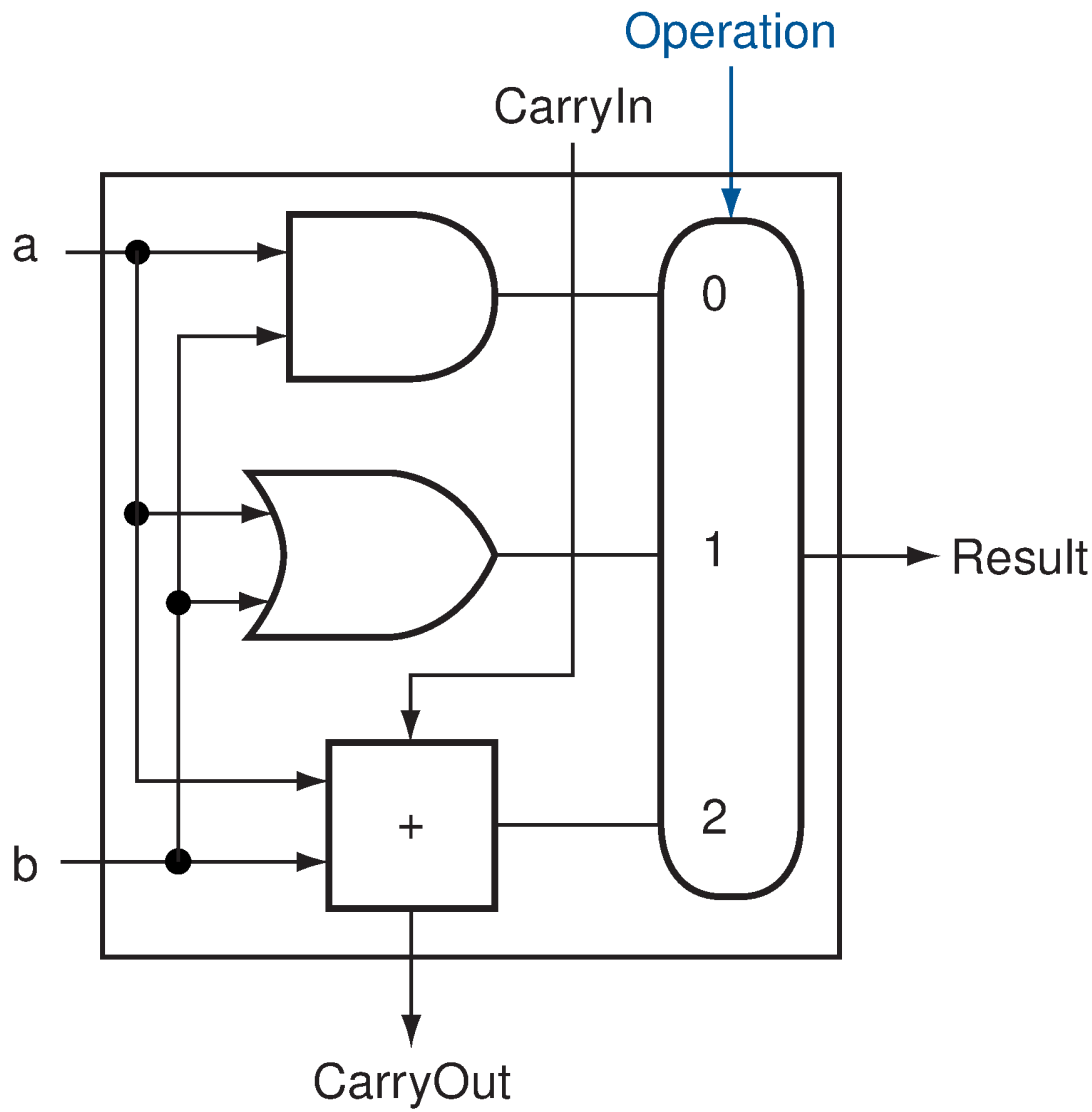
- Soma de produtos:
  - $CarryOut = (b.CarryIn) + (a.CarryIn) + (a.b) + (a.b.CarryIn)$
  - $Soma = (a.\overline{b}.\overline{CarryIn}) + (\overline{a}.b.\overline{CarryIn}) + (\overline{a}.\overline{b}.CarryIn) + (a.b.CarryIn)$
- Circuito da saída $CarryOut$ do somador de 1 bit:



ULA de 1 bit (AND, OR e soma)

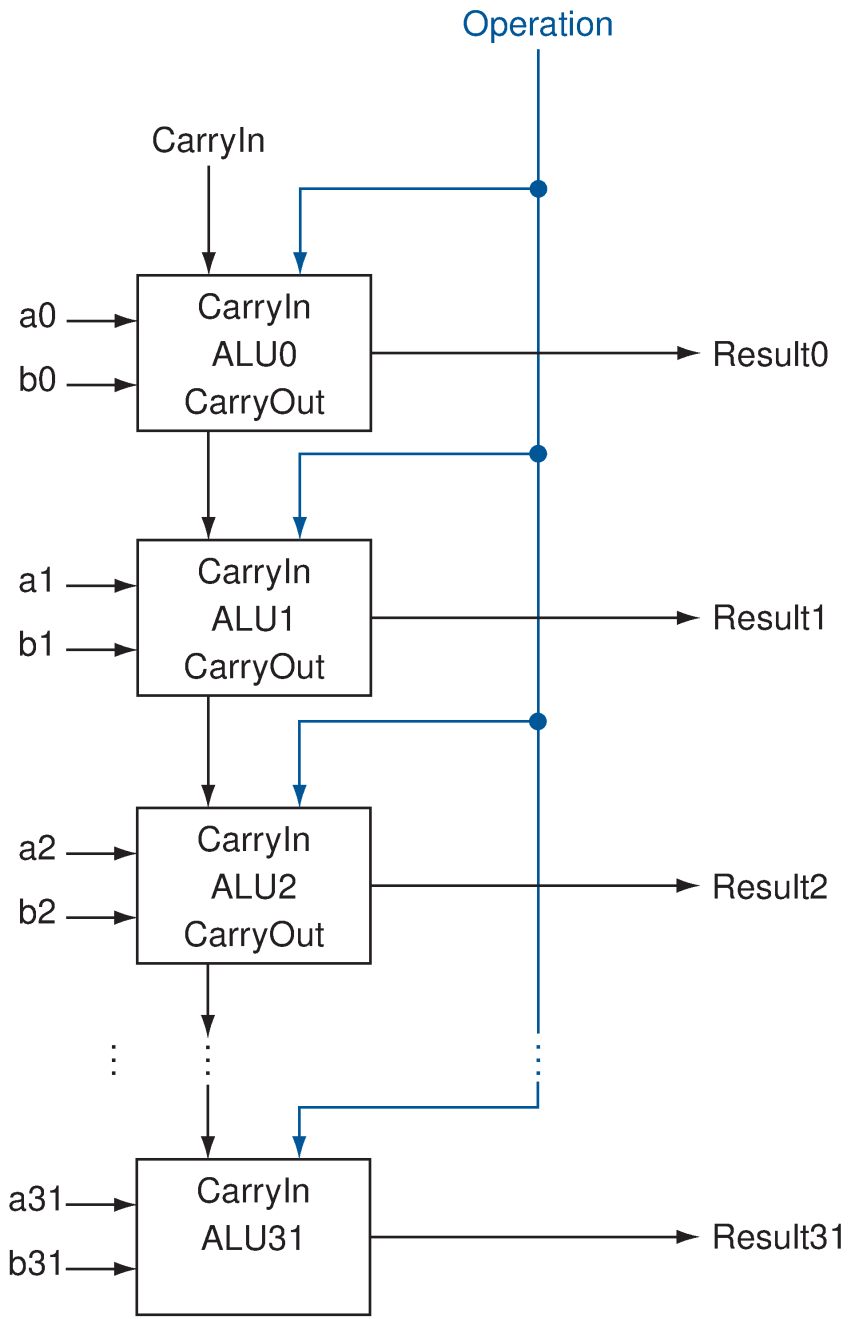
- Entradas:
  - a e b: dois dados de 1 bit a serem operados;
  - $CarryIn$: sinal de vai um “anterior”;
  - $Operation$: sinais de controle (2 bits);
- Saídas:
  - $Result$: dado de 1 bit resultante da operação;
    - Se $Operation = 00 \rightarrow Result = a$ `AND` b;
    - Se $Operation = 01 \rightarrow Result = a$ `OR` b;
    - Se $Operation = 10 \rightarrow Result = a + b$.
  - $CarryOut$: sinal de vai um.

ULA de 1 bit (AND, OR e soma)



ULA de 32 bits (AND, OR e soma)

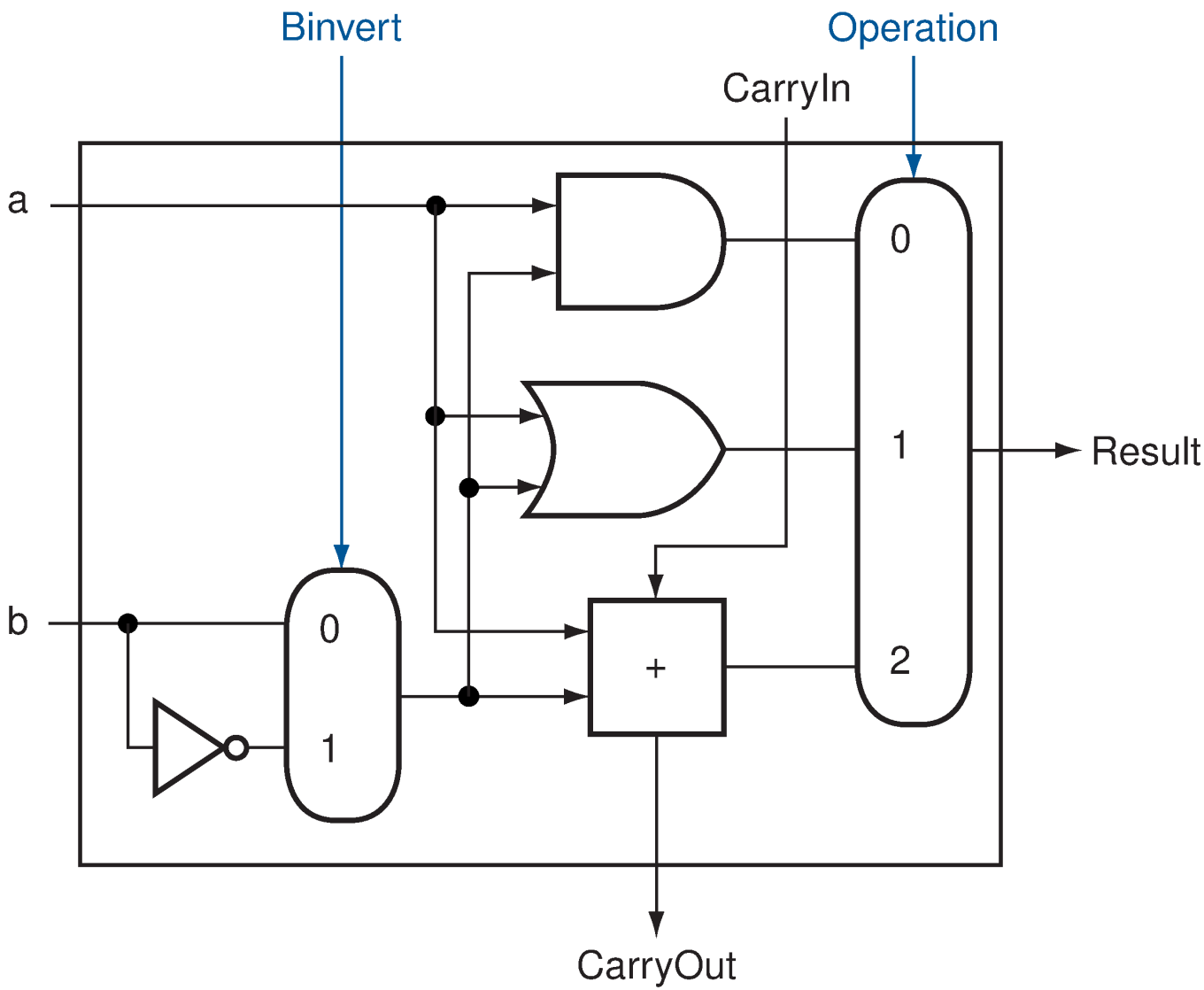
- Somador com carry em cascata (*Ripple carry adder*):



ULA de 1 bit (AND, OR, soma e subtração)

- Subtração: $a - b = a + (-b) = a + \texttt{NOT}(b) + 1$
- Entradas:
  - a e b: dois dados de 1 bit a serem operados;
  - $CarryIn$: sinal de vai um “anterior”;
  - $BInvert$: sinal de controle para inversão (NOT) da entrada;
    - Se $BInvert = 1$, o dado é negado antes de ser operado.
  - $Operation$: sinais de controle (2 bits);
- Saídas:
  - $Result$: dado de 1 bit resultante da operação;
    - Se $Operation = 00 \rightarrow Result = a$ AND b;
    - Se $Operation = 01 \rightarrow Result = a$ OR b;
    - Se $Operation = 10 \rightarrow Result = a + b$ ou$a - b$.
  - $CarryOut$: sinal de vai um.

ULA de 1 bit (AND, OR, soma e subtração)



ULA de 1 bit (AND, OR, soma e subtração)

- Multiplexador de 2 entradas de 1 bit: seleciona segundo dado de entrada;
  - Se $BInvert = 0 \rightarrow$ segundo dado de entrada $= b$;
  - Se $BInvert = 1 \rightarrow$ segundo dado de entrada $= \texttt{NOT}(b)$;
- Para subtração:
  - $BInvert = 1$ e $CarryIn$ da ULA correspondente ao bit menos significativo $= 1$.

ULA de 1 bit (AND, OR, soma, subtração e `slt`)

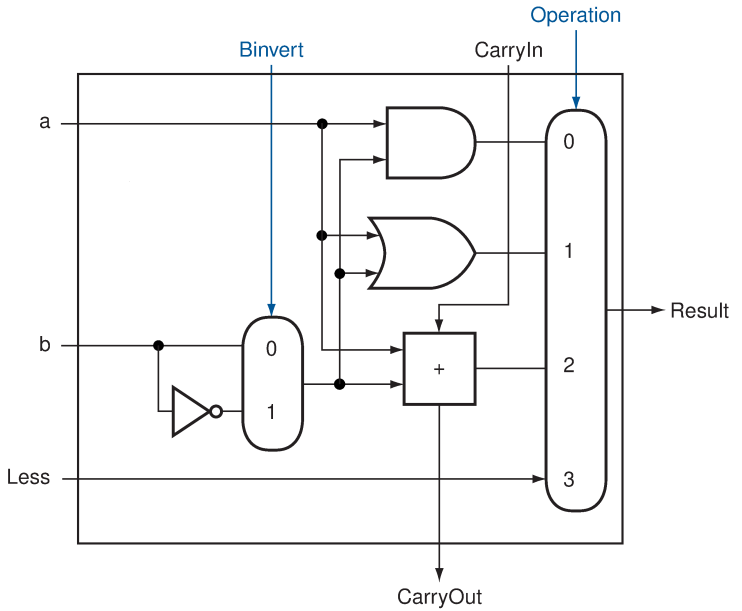
- `slt rd, rs, rt`
  - Se $rs < rt$ então $rd = 1$, senão $rd = 0$;
  - Saída de 32 bits terá todos os bits em 0, exceto pelo bit menos significativo, que é setado de acordo com a comparação.
- Idéia: $a < b \rightarrow a - b < 0$
- Entradas:
  - a e b: dois dados de 1 bit a serem operados;
  - $CarryIn$: sinal de vai um "anterior";
  - $BInvert$: sinal de controle para inversão (`NOT`) da entrada b;
  - $Less$: sinal resultado da operação `slt`;
  - $Operation$: sinais de controle (2 bits).

ULA de 1 bit (AND, OR, soma, subtração e `slt`)

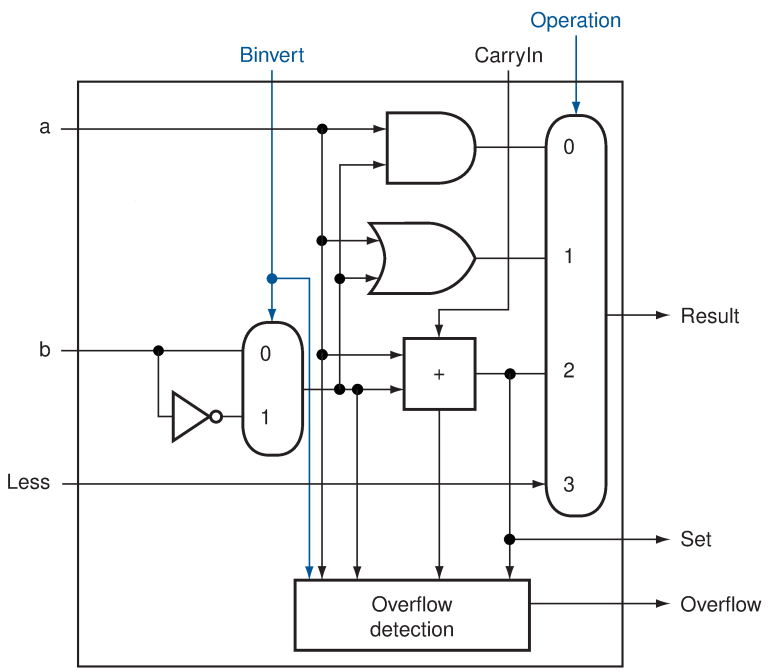
- Saídas:
  - $Result$: dado de 1 bit resultante da operação;
    - Se $Operation = 00 \rightarrow Result = a$ `AND` b;
    - Se $Operation = 01 \rightarrow Result = a$ `OR` b;
    - Se $Operation = 10 \rightarrow Result = a + b$ ou$a - b$.
    - Se $Operation = 11 \rightarrow Result =$ resultado da operação `slt`.
  - $CarryOut$: sinal de vai um.
  - Apenas para ULA do bit mais significativo:
    - Set: dado de 1 bit resultante do somador (é o bit de sinal);
    - $Overflow$: sinal de 1 bit que indica se ocorreu *overflow*.

ULA de 1 bit (AND, OR, soma, subtração e `slt`)

31 bits menos significativos:



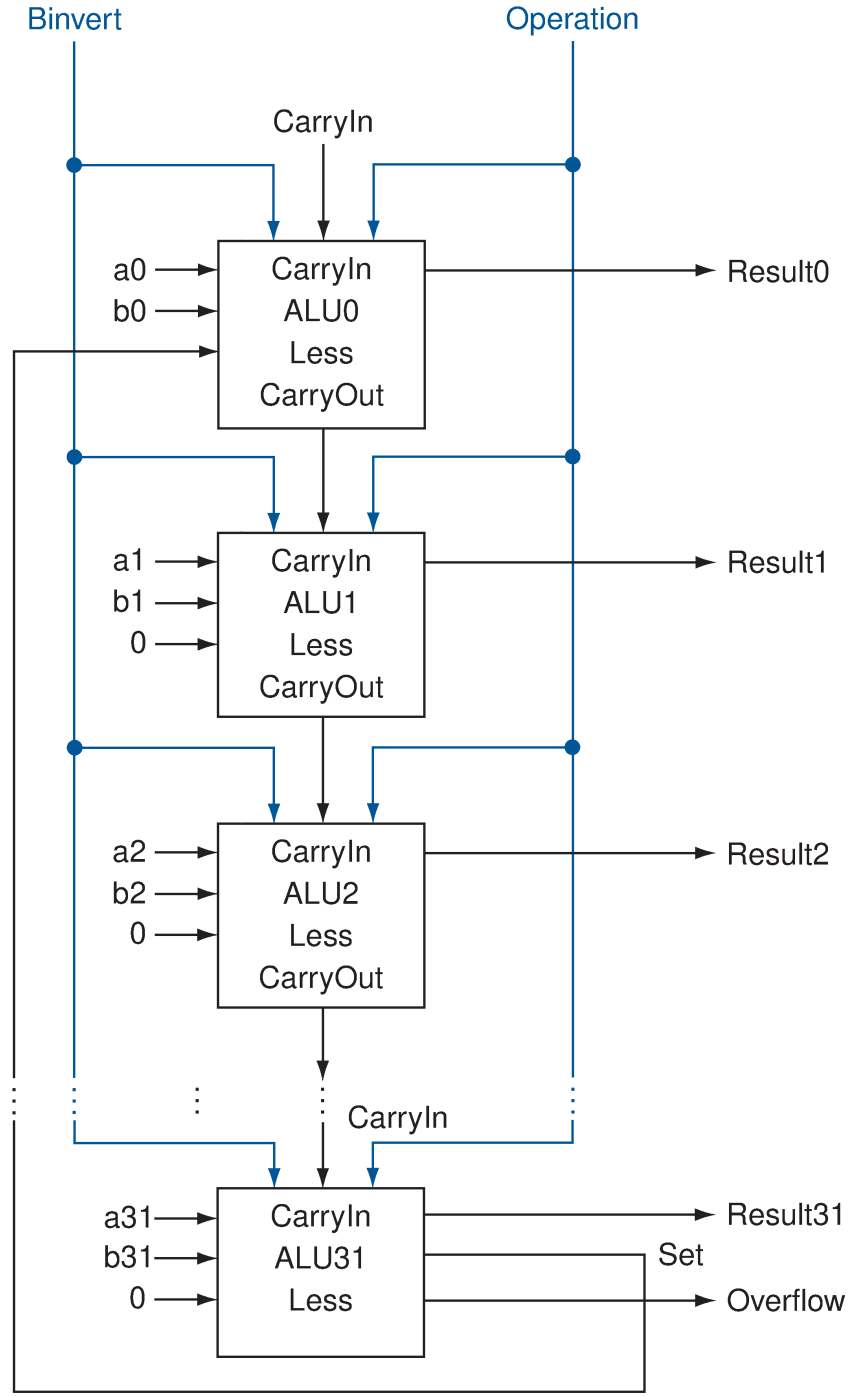
bit mais significativo:



ULA de 1 bit (AND, OR, soma, subtração e `slt`)

- Para `slt`:
  - Entrada $Less$ é 0 para todas as ULAs, exceto para a ULA do bit menos significativo;
  - Entrada $Less$ da ULA do bit menos significativo é o resultado do somador da ULA do bit mais significativo (bit de sinal).

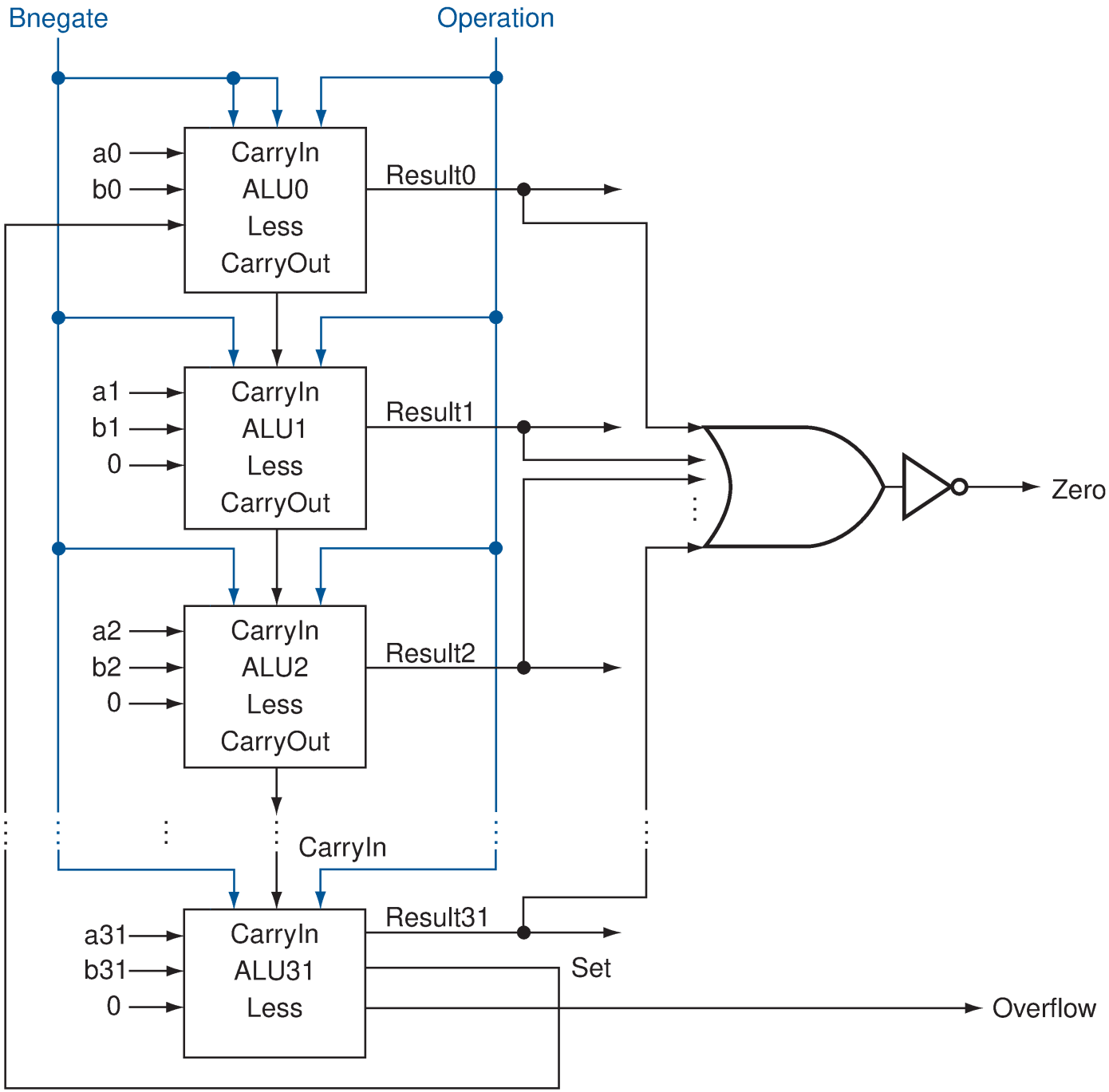
ULA de 32 bits (AND, OR, soma, subtração e `slt`)



ULA de 1 bit (AND, ... e igualdade)

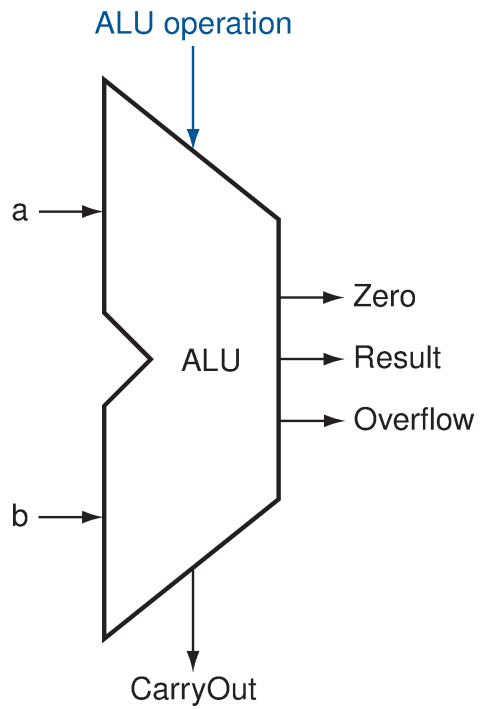
- Utilizada nas instruções `beq` e `bne`;
- Idéia: se $a = b \rightarrow a - b = 0$;
  - Faz o `OR` do resultado da subtração das 32 ULAs e inverte: $Zero = \texttt{NOT}(Result31\ \texttt{OR}\ \ldots\ \texttt{OR}\ Result1\ \texttt{OR}\ Result0)$
- Entradas:
  - a e b: dois dados de 1 bit a serem operados;
  - $BNegate$: combinação de $BInvert$ e $CarryIn$;
  - $Less$: sinal resultado da operação `slt`;
  - $Operation$: sinais de controle (2 bits).
- Sinal de saída $Zero$: se $a = b$ então $Zero = 1$.

ULA de 32 bits (AND, ... e igualdade)



ULA de 32 bits (AND, ... e igualdade)

Representação da ULA de 32 bits



Entrada de controle (junção de $BNegate$ e $Operation$).

Linhas de controle	Função
000	`and`
001	`or`
010	`+`
110	`-`
111	`slt`

Detecção de *overflow*

- *Overflow*: ocorre quando o resultado da operação aritmética não pode ser representado com o número de bits da palavra;
- Quando ocorre *overflow*, o processador causa uma exceção:
  - Execução do programa é suspensa;
  - Execução é desviada para um endereço pré-definido, onde uma rotina de tratamento é chamada;
- *Overflow* ocorre quando (a e b são dados de 32 bits);

Operação	a	b	Resultado
$a + b$	≥ 0	≥ 0	< 0
$a + b$	< 0	< 0	≥ 0
$a - b$	≥ 0	< 0	< 0
$a - b$	< 0	≥ 0	≥ 0