

Introdução à Arquitetura e Organização de Computadores

Professor: Diego de Assis Monteiro Fernandes

Aula 9 - Aritmética de Computadores 2

1 Construindo uma Unidade Lógica e Aritmética

A **unidade lógica e aritmética** ou **ULA** é o dispositivo que realiza operações aritméticas como adição ou operações lógicas como **and** (E-lógico). Essa aula tem como objetivo construir uma ULA a partir de quatro elementos básicos de construção mostrados na Figura 1. Apesar da Figura 1 mostrar as portas lógicas e o multiplexador com somente duas entradas, algumas vezes esses componentes são utilizados durante a aula em versões com mais entradas. Como a arquitetura MIPS trabalha com palavras de 32 *bits*, é fácil concluir que sua ULA também trabalha com dados de 32 *bits*.

Uma forma de construir uma ULA que manipule dados de 32 bits é projetar 32 ULAs que são capazes de manipular um 1 *bit* e uni-las. A justificativa para esse tipo de projeto é a de que a construção de um circuito que executa as operações básicas para um único bit é simples de ser projetado. Da mesma forma, a tarefa de conectá-las também é simples. A seção a seguir mostra o projeto de uma ULA de um *bit*.

1. AND gate ($c = a \cdot b$)



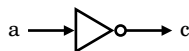
a	b	c = a · b
0	0	0
0	1	0
1	0	0
1	1	1

2. OR gate ($c = a + b$)



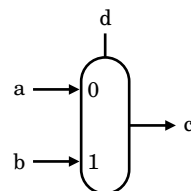
a	b	c = a + b
0	0	0
0	1	1
1	0	1
1	1	1

3. Inverter ($c = \bar{a}$)



a	c = \bar{a}
0	1
1	0

4. Multiplexor
(if $d = 0$, $c = a$;
else $c = b$)



d	c
0	a
1	b

Figura 1: Elementos utilizados na construção da ULA.

2 ULA de 1 bit

A seguir estão descritos os projetos de algumas operações que a ULA deve desempenhar.

2.1 Operações Lógicas

As operações lógicas executadas pelas instruções `and` e `or` presentes na arquitetura MIPS são simples de serem implementadas, pois são mapeadas diretamente dos componentes básicos, como mostrado na Figura 2. O multiplexador, incluído no circuito, é utilizado para selecionar a operação a ser realizada.

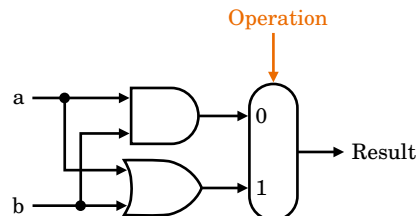


Figura 2: Unidade lógica de 1 bit para AND e OR.

2.2 Operação de soma

A próxima função a ser implementada é a adição. Mas para implementá-la, quais são as entradas e saídas em que um somador de um 1 *bit* deve possuir? Primeiro, deve possuir duas entradas para os operandos e uma única saída para o resultado. Porém, essa saída não é a única que um circuito de soma deve prover pois, além disso, deve existir uma saída para indicar o *bit* referente ao *carry* (vai-um). Por fim, como esse somador de 1 bit vai estar integrado a outros somadores, uma vez que a ULA completa da arquitetura MIPS trabalha com dados de 32 bits, ele deve incluir uma entrada referente ao *carry* produzido por um somador anterior. A Figura 3 resume o projeto descrito, destacando as entradas e saída que um somador completo de um *bit* deve possuir.

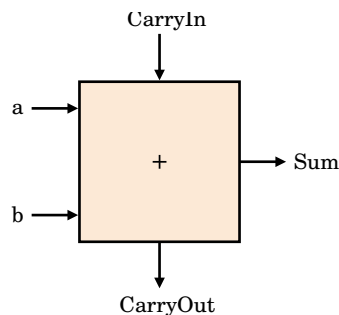


Figura 3: Somador de 1 bit.

A equação lógica para o *bit* “vai-um” de saída é a seguinte: $CarryOut = (b.CarryIn) + (a.CarryIn) + (a.b)$. A Figura 4 mostra a implementação com portas lógicas dessa equação. A equação booleana para o resultado da soma é a seguinte: $Sum = (a.b.CarryIn) + (\bar{a}.b.CarryIn) + (\bar{a}.\bar{b}.CarryIn) + (a.b.CarryIn)$

2.3 Combinação das operações

A Figura 5 mostra uma ULA de 1 *bit*, combinando o circuito do somador completo com o circuito das operações lógicas projetado anteriormente. É importante notar que no circuito o multiplexador passa a ter

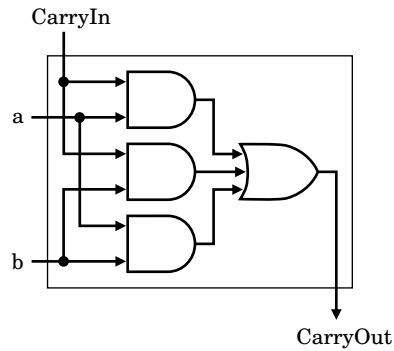


Figura 4: Hardware para o cálculo do vai-um.

três entradas, pois a ULA pode realizar tanto a operação de soma, E-lógico ou OU-lógico. A função do multiplexador é a de selecionar resultado da operação que deseja-se desempenhar.

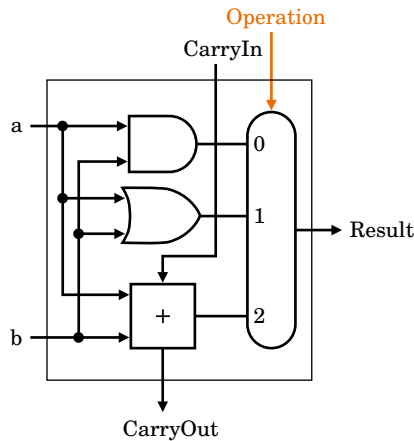


Figura 5: Uma ULA de 1 bit que realiza AND, OR e adição.

3 ALU de 32 bits

A partir da ULA de 1 *bit* apresentada, pode-se construir uma ULA de 32 *bits* conectando diversas ULAs triviais. A Figura 6 mostra a integração das ULAs triviais formando uma ULA de 32-bits, onde a_i e b_i significam, respectivamente, o i -ésimo *bit* de a e b . É possível perceber que as ULAs são integradas pelas entradas e saídas referentes ao *carry* do circuito de soma. Mais ainda, a entrada que informa a operação a ser executada é compartilhada por todas as ULAs e o resultado é a composição dos bits de saída da ULA que totalizam o tamanho de uma palavra da arquitetura.

3.1 Operação de subtração

A próxima operação importante a ser adicionada é a subtração. A operação de subtração $a - b$ é semelhante a adição de a com a versão negativa de b , ou seja, o operando a ser subtraído. Isso permite que o circuito de soma projetado anteriormente seja utilizado também para realizar a subtração.

O próximo passo para o projeto da subtração é obter a versão negativa do operando b . Para isso, é preciso aplicar a transformação do complemento de 2, que é forma utilizada pelo computador para obter a versão

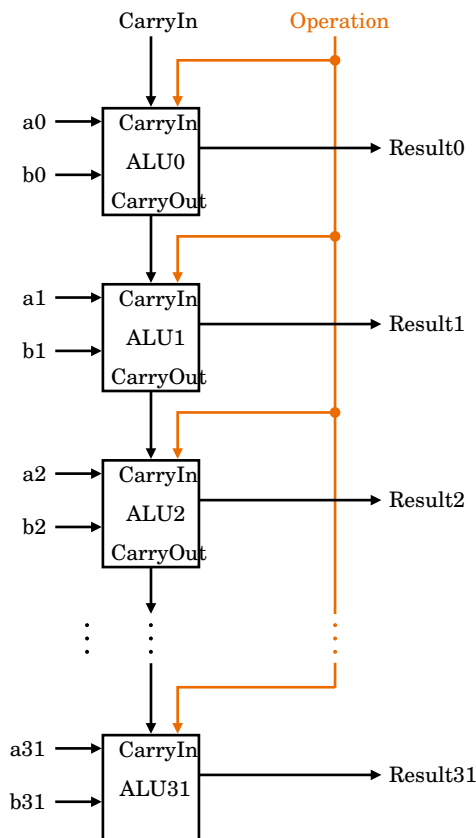


Figura 6: Uma ULA de 32 bits.

negativa de um número binário. Como já foi visto, o complemento de 2 de um número é obtido através da inversão de cada *bit* e da soma com 1. Para inverter cada *bit* do operando b , basta adicionar na ULA um multiplexador que seleciona b ou \bar{b} como entrada. A integração dessa entrada na ULA de 1 bit é mostrada na Figura 7.

Esse é o primeiro passo para a negação do número, faltando ainda realizar a soma de 1 à inversão do número b ou \bar{b} . Esse valor, que necessita de um único *bit* para ser representado, pode ser adicionado utilizando o *CarryIn* (vai-um de entrada) do *bit* menos significativo (ver Figura 6). Com essa transformação, o somador então vai calcular a soma $a + \bar{b} + 1$, ou seja, $a + \bar{b} + 1 = a + (\bar{b} + 1) = a + (-b) = a - b$

3.2 Estendendo a ULA de 32 bits para a operação de comparação

O conjunto de operações - adição, subtração, and e or - é desempenhado na ULA de quase todos os computadores. Entretanto, existem outras operações importantes ainda não implementadas, basta observar o conjunto de instruções estudado da arquitetura MIPS para constatar tal fato. Uma instrução importante que ainda não foi implementada é a `slt` (*set on less than*), que realiza uma comparação. A operação produz 1 se $rs < rt$ ou 0 caso contrário.

Portanto, é fácil perceber que a instrução `slt` vai atribuir 0 para todos os *bits* de resultado, exceto ao *bit* menos significativo que vai assumir o seu valor de acordo com a comparação. Para que a ULA execute a instrução `slt`, o multiplexador que determina a operação a ser realizada deve ser expandido. Uma entrada (`Less`) vai ser usada na expansão e vai possuir o valor 0 para todas as entradas exceto para o bit menos significativo de resultado.

Mas como a comparação $a < b$ pode ser implementada pela ULA? Se b for subtraído de a e a diferença

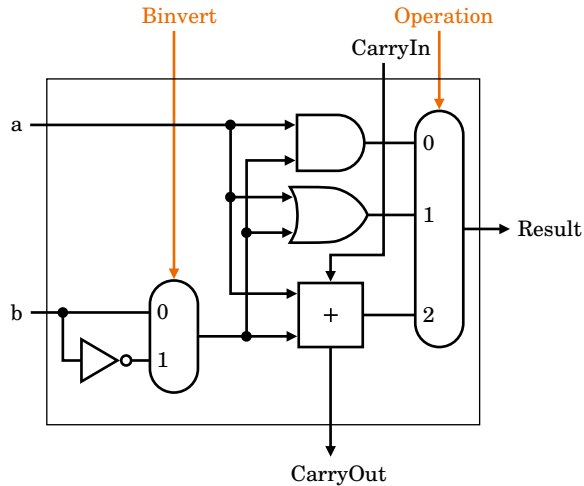


Figura 7: Alteração da ULA de 1 bit para a incorporação da subtração.

for negativa, o seguinte pode ser concluído:

$$(a - b) < 0 \Rightarrow ((a - b) + b) < (0 + b) \Rightarrow a < b \quad (1)$$

Isso significa que o bit menos significativo do resultado da operação deve ser 1 se a operação $a - b$ fornecer um número negativo e 0 se positivo. É fácil ver que o valor que deve ser atribuído ao bit menos significativo de resultado corresponde exatamente ao *bit* mais significativo (que representa o sinal) do valor obtido com a subtração. Seguindo esse raciocínio, basta ligar o *bit* de sinal do resultado do somador à entrada Less do *bit* menos significativo. Isso é permitido através da adição de uma saída *Set* na ULA do *bit* mais significativo que é ligada a entrada Less do *bit* menos significativo. A incorporação da entrada *Less* e da saída *Set* no projeto da ULA pode ser observada na Figura 8

Note que sempre que a ULA for subtrair, os sinais *CarryIn* e *Binvert* devem ser 1. Já para a adição e as operações lógicas, essas linhas de controle devem ser 0. Por isso, tais linhas de entrada são combinadas em uma única linha chamada **Bnegate**.

3.3 Desvio Condicional

Para estender ainda mais a ULA, ela deve suportar instruções de desvio condicional. Essas instruções realizam o desvio se dois registradores são iguais ou diferentes, dependendo do caso. O modo mais simples de verificar a igualdade entre a e b é subtrair b de a e verificar se o resultado é 0, ou seja, verificar se $a - b = 0$.

O modo mais simples de verificar se o resultado de uma operação é 0 é mostrado pela equação booleana a seguir.

$$Zero = \overline{(Result_{31} + Result_{30} + \dots + Result_1 + Result_0)} \quad (2)$$

A Figura 8 mostra também a saída *Zero* na ULA de 32 *bits*. É possível observar também que na ULA trivial do *bit* mais significativo existe a saída *overflow*. Como o próprio nome sugere, essa saída tem como objetivo avisar a possível ocorrência de um *overflow* em uma operação aritmética. As técnicas para a detecção de *overflow* já foram mostrada na Aula 8.

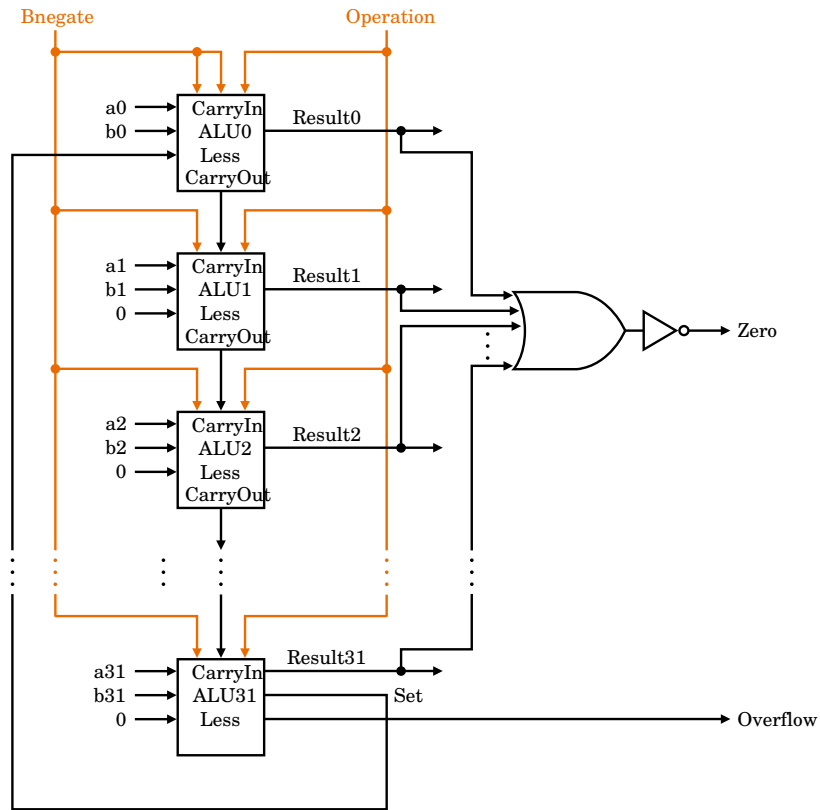


Figura 8: Uma ULA de 32 bits.

4 Símbolo da ULA

Por fim, a Figura 9 mostra o símbolo comumente utilizado para representar uma ULA completa.

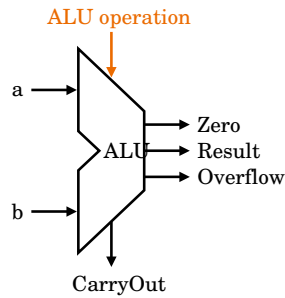


Figura 9: Símbolo de uma ULA.