**Projeto**

**Projeto e Teste de uma Unidade Lógica Aritmética**

**(Prazo: 1 semana)**

**Parte 1 - Projetando a ULA**

Vamos projetar uma ULA capaz de realizar um subconjunto das operações de uma ULA MIPS completa. Você pode consultar o Apêndice B do livro H&H para ver o conjunto completo de operações que o MIPS pode suportar. Neste projeto, desenvolveremos uma ULA que recebe duas entradas A e B de 32 bits e executa as sete instruções a seguir:

add, sub, slt, and, or, xor, nor

A ULA gera uma saída de 32 bits que chamaremos de 'Result' e um bit sinalizador adicional 'Zero' que será definido como 1 se todos os bits de 'Result' forem 0. As diferentes operações serão selecionadas por um sinal de controle de 4 bits chamado 'AluOp' de acordo com a tabela a seguir.

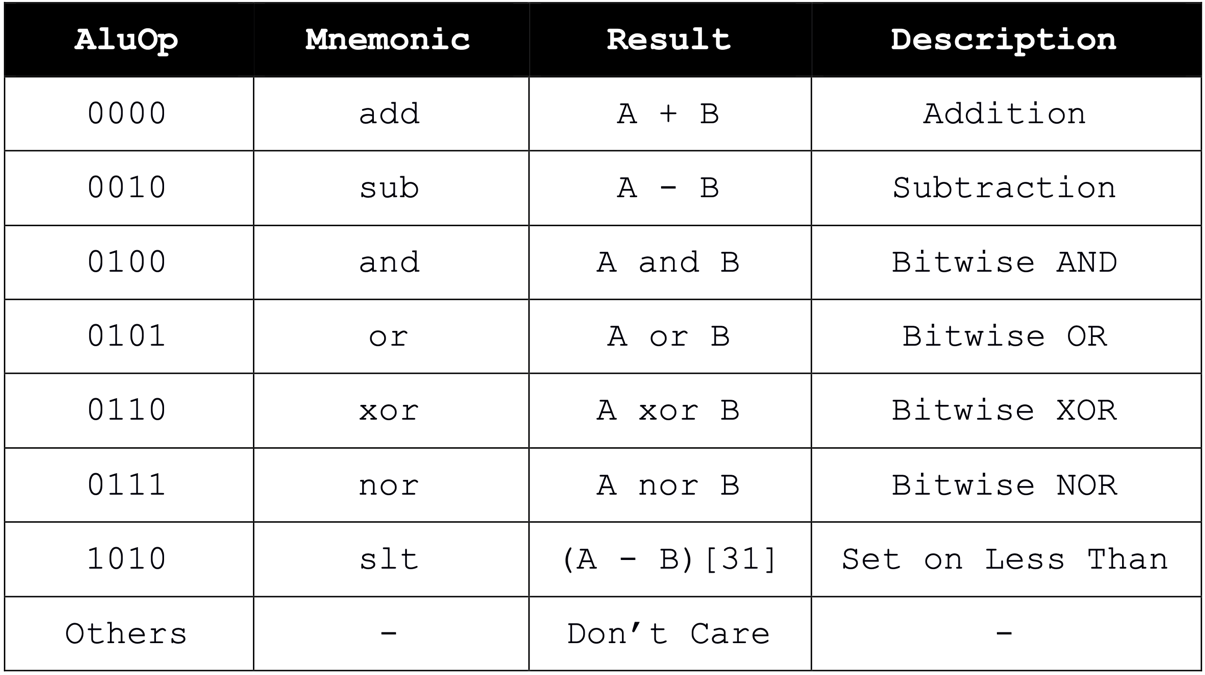


Tabela 1: Operações da ULA

Nota: Você deve estender o resultado de slt para 32 bits (ou seja, 32'b00 ou 32'b01)

Muitos valores de 'AluOp' não correspondem a nenhuma operação, e não é importante o que o circuito faz quando 'AluOp' tem esses valores, pois o resultado será simplesmente ignorado em tais casos.

**Projetando o diagrama de blocos**

Primeiramente, você precisa desenhar um diagrama de blocos para a ULA, análogo ao apresentado na Figura 5.15 do livro H&H. Você é livre para apresentar suas próprias ideias, mas aqui vai algumas dicas.

Primeiro examine as diferentes operações. Observe que temos dois tipos de instruções. As três instruções **add**, **sub** e **slt** requerem operações aritméticas, enquanto as quatro restantes **and**, **or**, **xor** e **nor** são operações lógicas bit a bit. Agora observe a tabela considerando os dois grupos de operações e determine para quais valores de AluOp realizamos uma operação de cada grupo. Deve ficar claro que quando AluOp[2] é 0 uma operação aritmética é selecionada, ao passo que quando AluOp[2] é 1, selecionamos uma operação lógica. Isso significa que a saída de qualquer um dos tipos pode ser selecionada por um multiplexador de 2 entradas controlado por AluOp[2]. A figura 1 abaixo representa um projeto de ULA que inclui um bloco lógico separado para a parte aritmética e outro para a parte lógica.

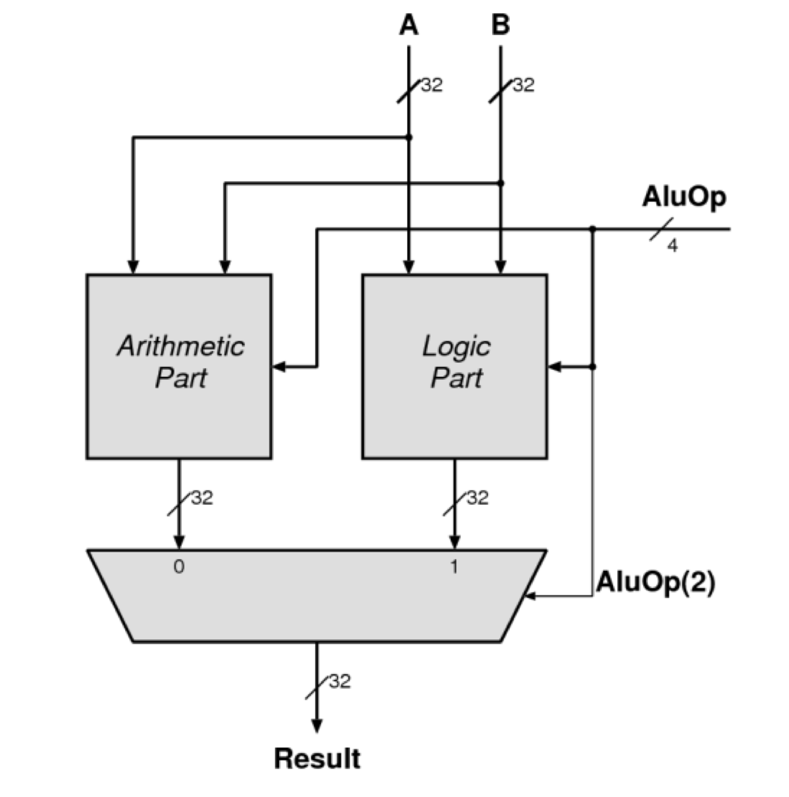


Figura 1: Possível particionamento da ULA.

Agora que estão separados, podemos dar uma olhada nos dois grupos individualmente. Para a parte lógica, AluOp[1:0] seleciona uma das 4 operações bit a bit. Na parte aritmética, veja que temos uma adição (add) ou uma subtração (sub, slt). Podemos ver que AluOp[1] é 0 para adições e 1 para subtrações. Isso nos permitir construir uma estrutura como a da Figura 5.15 do livro H&H para projetar um somador-subtrator (controlado com AluOp[1] em vez de F[2]). A Figura 2 mostra esse projeto.

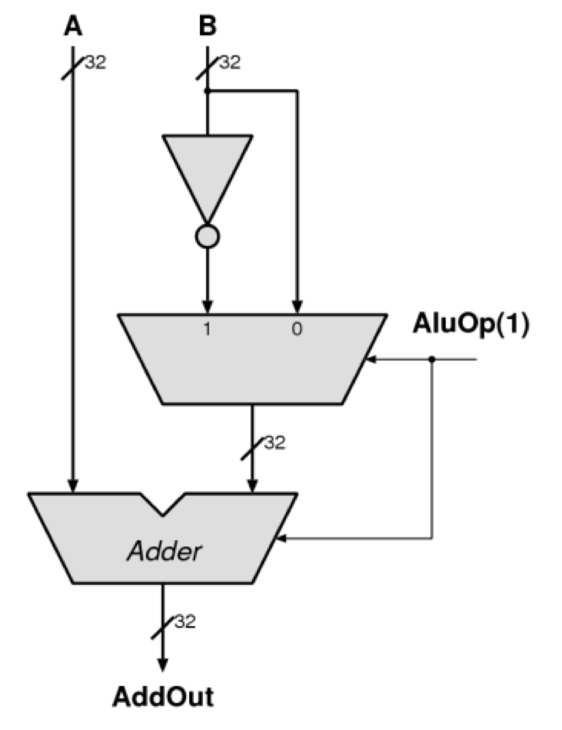


Figura 2: Possível Organização para o somador-subtrator.

Resta mais uma coisa, com AluOp[3] podemos selecionar se pegamos apenas o bit mais significativo (1, instrução slt) ou pegamos a saída normal. Uma possibilidade para esta parte está na Figura 3.

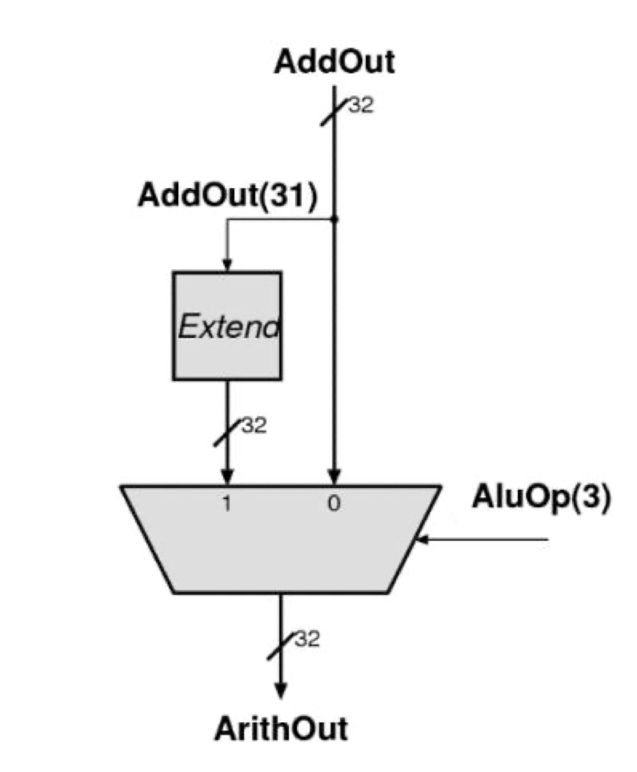


Figura 3: Possível Organização para implementar SLT

Seu trabalho nessa primeira parte é desenhar um diagrama de blocos que implemente as operações da ULA listadas na Tabela 1. Você é livre para decidir como implementar a ULA e não precisa seguir exatamente o diagrama de blocos explicado acima. Você pode usar somadores de tamanho arbitrário, multiplexadores, portas lógicas, extensão de zero/sinal, comparadores e shifters, da forma que desejar.

**Parte 2 – Implementação da ULA**

Com um bom diagrama de blocos, fica relativamente fácil implementar o circuito em VHDL. Crie um projeto no quartus e substitua cada bloco por uma entity VHDL. Seja fiel ao seu projeto no que se refere a nomenclatura dos blocos e dos sinais.

Infelizmente, após a implementação, não saberemos se o circuito funciona corretamente. Isso porque não é possível verificar se o circuito funciona testando-o diretamente, pois há muitos bits de entrada. Em vez disso, usaremos um testbenchs para verificar a funcionalidade.

1. **Procedimento de testagem da ULA**

Baixe o arquivo testvector\_hex.txt disponível no site da disciplina. De acordo com a funcionalidade esperada para a ULA, determine o resultado correto para cada uma das entradas e atualize o arquivo 'testvectors\_hex.txt' com os valores determinados. Em seguida, crie 5 vetores de teste adicionais à sua escolha e acrescente no arquivo.

Lembre-se que a ULA tem uma saída adicional de um bit, chamada ‘zero’. O valor esperado dessa saída pode ser facilmente determinado com base no resultado. Assim, podemos definir diretamente seu valor esperado dentro do testbench. Se o valor esperado do resultado for zero, o valor esperado do sinal ‘zero’ deve ser 1.

Crie um módulo testbench em VHDL e implemente a lógica necessária para testar a ULA usando o arquivo testvector\_hex.txt. Faça a simulação dos testes usando o simulador Questa Sim Intel, instalado no laboratório do curso. Examine os resultados da simulação para garantir que o circuito está funcionando corretamente. No relatório indique todos as possíveis correções que você precisou fazer.