Grupo 1

Desenvolvimento:
Elias da Silva Barroso Soares
Nivaldo Roberto Teixeira Junior
Vitor Guilherme Ribeiro Lopes

Testes:
Douglas Sales Silva
Manasses Ferreira Neto

Unidade Lógica e Aritmética (ALU):

A unidade lógica aritmética (ALU – Arithmetic Logic Unit) é o módulo responsável por operações lógicas, como AND e OR, e também por operações aritméticas, como soma e subtração. O módulo implementado possui dois arrays de 32 bits, ***A*** e ***B***, que servirão como entrada dos operandos e o resultado da operação será armazenado em um array de 32 bits chamado ***aluout***. Além disso, terá como entrada um array de 3 bits ***op*** que servirá para receber os códigos da operação a ser realizada de acordo com a tabela abaixo:

Código	Operação
000	A and B
001	A or B
010	A + B
100	A nor B
101	A xor B
110	A - B
011 e 111	Não possuem operações

Este módulo ainda terá presente o sinal de saída ***compout*** que será codificado como 1 quando o operando *A* for menor que o operando *B* e 0 caso contrário. Também estará presente o flag ***unsig*** que será codificado como 1 se o comparador anterior deverá considerar as entradas com sinal e 0 caso contrtário. Além disso, há o flag de ***overflow*** que indica quando algumas das operações da ALU resulta em um resultado inesperado devido ao numero insuficiente de bits para representá-lo. Abaixo, uma tabela que ilustra casos de possíveis overflows:

Operação	Operando A	Operando B	Resultado
A + B	>= 0	>= 0	< 0
A + B	< 0	< 0	>= 0
A - B	>= 0	< 0	< 0
A - B	< 0	>= 0	>= 0

Inicialmente, o módulo compara os códigos de operações, executando o que cada um determina. Dentro das operações de adição e subtração, é feita a verificação de overflow de acordo com a tabela anterior. Por fim, é feita a comparação entre os operando e este valor é guardado na saída de *compout.* Se caso o código de operação não for reconhecido, foi tomada a decisão de projeto de não executar quaisquer operação sobre os operandos, apenas comparando seus valores e

atualizando a saída *compout.*

Shifter:

O shifter é a unidade resposável pelo deslocamento de bits, que pode ser lógico ou aritmético e para esquerda ou para a direita. Sua entrada é um valor de 32 bits que pode ser deslocados por até 31 posições. Este módulo possui um array de 32 bits chamado ***in*** que será usado para abrigar o valor de entrada a ser deslocado, um array de 2 bits chamado ***shiftop*** que será usado para indicar qual operação de deslocamento será realizada (00 shift lógico para a direita, 01 shift aritmético para a direita, 10 shift lógico para a esquerda), um array de 4 bits chamado ***shiftamt*** que indicará a quantidade de bits a serem deslocados e, finalmente, um array de 32 bits chamado ***result*** que indicará o resultado da operação.

Este módulo foi implementado através de um seletor onde reconhece qual tipo de deslocamento será calculado e o executa no operando de entrada. Este operando é deslocado de acordo com o indicado em shiftamt. Quando não houver a codificação correta do desclocamento, o resultado reflete apenas o que foi dado no operando de entrada.

Comparador:

O comparador é um módulo que compara dois valores tem como saída o resultado dessa comparação que pode ser utilizada em instruções de desvio. Este módulo possui dois arrays com 32 bits chamados de ***A*** e ***B*** que serão usados para abrigar os valores a serem comparados, além de um array com 3 bits chamado ***op*** que será usado como controle que indicará qual operação de comparação a ser realizada de acordo com a tabela abaixo e uma saída chamada ***compout*** que retornará 1 caso a comparação realizada seja verdadeira e 0 caso contrário.

Código	Comparação
000	Se A = B
001	Se A >= B
010	Se A <= B
011	Se A > B
100	Se A < B
101	Se A != B
110 e 111	Não possuem operações

O módulo foi implementado como um seletor, onde o código de comparação é verificado de acordo com os códigos anteriormente citados e o valor da comparação entre os operandos *A* e *B* é retornado na saída *compout.* Por padrão, é retornado o valor 0 caso o código de comparação não seja reconhecido.