Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

Organização de Computadores I

Lab 2 : Unidade Lógico-Aritmética

Professor: Antonio Otavio Fernandes.
Monitor: Omar Vidal Pino.

28 de Setembro de 2016

1 Introdução

A Unidade Lógico-Aritmética (**ULA**) de um processador é o módulo responsável pela realização de todas as operações lógicas e aritméticas definidas pelo seu conjunto de instruções. Entre as operações lógicas estão AND, OR, XOR e NOT, além daquelas relativas ao deslocamento à esquerda ou à direita do conteúdo de um registrador. As três primeiras são binárias (*operam sobre dois registradores*) e as de deslocamento e o NOT são unárias (*operam sobre um único registrador*). Entre as operações aritméticas, a adição e a subtração estão presentes em todos os processadores, sendo as operações de multiplicação e divisão restritas aos processadores de maior desempenho.
Outra operação inclui a comparação entre inteiros, com ou sem sinal, realizada através da subtração sem atualização de registradores, mas refletindo o resultado nos *sinalizadores* (IGUAL, MENOR, MAIOR).
O resultado de uma operação na ULA é tipicamente armazenado num registrador de destino e as condições resultantes da operação são armazenadas em sinalizadores (*flags*), tais como:

- **zero**: o resultado da operação na ULA produziu o valor 0 (zero);
- **positivo**: idem, produziu um valor positivo (inteiro com sinal);
- **negativo**: idem, produziu um valor negativo (inteiro com sinal);
- **estouro**: idem, produziu um valor que ultrapassa o limite máximo (inteiro com sinal);
- **vai-um (carry)**: idem, produziu um vai-um final do bit mais significativo (inteiro sem sinal);

- **paridade**: idem, indica se o número de 1's no resultado é par ou não.

Na arquitetura MIPS, como explicado no livro-texto, decidiu-se implementar apenas um sinalizador, o ZERO, sendo os outros derivados dele em função da realização de outras operações entre os operandos.

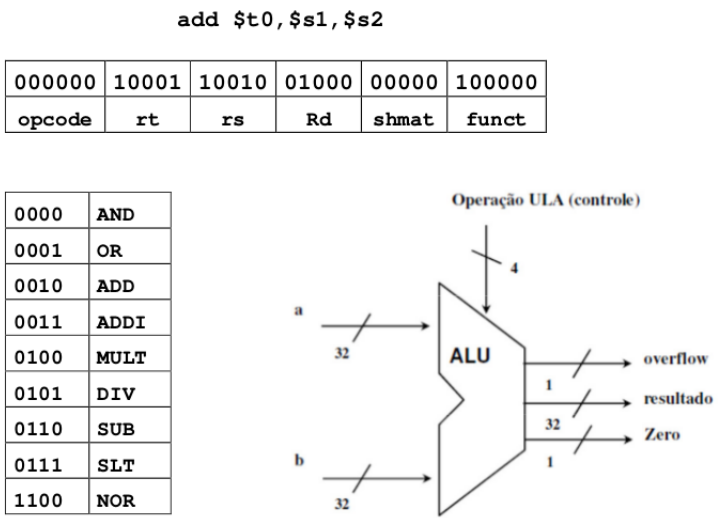

Figura 1.1: Exemplo de instruções , operações e módulo ULA

2 Objetivos

O objetivo desta aula prática é a implementação, simulação e teste de uma ULA que implemente as seguintes operações lógicas e aritméticas:

- AND, OR, NOR: AND, OR, NOR lógico entre dois operandos de 32 bits;
- ADD: adição de dois inteiros de 32 bits;
- SUB: subtração entre dois inteiros de 32 bits;
- SLT: comparação entre dois inteiros de 32 bits.

O projeto lógico deverá ser definido através da linguagem Verilog, já estudada nas disciplinas de ISL. No site da disciplina, há várias referências sobre a linguagem. Após a entrada do projeto lógico, ele deverá ser compilado e simulado, para verificação do seu correto funcionamento.

3 Parte Experimental

A parte experimental consiste em testar o projeto de uma ULA que suporte um subconjunto das instruções definidas para o processador MIPS. Para isto, conforme descrito no livro-texto, serão implementadas as funções dadas na tabela abaixo e que farão uso da ULA em seu processamento:

Classe	Operação	funct	Função da ULA	Controle da ULA
R-type	ADD	100000	somar	0010
R-type	SUB	100010	subtrair	0110
R-type	AND	100100	AND lógico	0000
R-type	OR	100101	OR lógico	0001
R-type	NOR	100111	NOR lógico	1100
R-type	set-on-less-than	101010	ativar se menor	0111
I-type	load upper immediate	001111	carregar imediato	1000

Figura 3.1: Funções ULA.

3.1 Atividade 1

Crie um novo projeto (module) para uma Unidade Lógico-Aritmética (ULA) básica , com apenas as operações: AND lógico, adição , set-less-than. Salve o modulo com o nome ULA.v. Um exemplo de código é mostrado abaixo. (Ver pasta *code-examples*.)

```
module ALU(ALUctl, A, B, ALUOut, Zero);
parameter wordSize = 32;
parameter OpSize = 4;
input [OpSize-1:0] ALUctl;
input [wordSize-1:0] A,B;
output reg [wordSize:0] ALUOut;
output Zero;
assign Zero = (ALUOut==0); //Zero is true if ALUOut is 0
always @(ALUctl, A, B) begin //reevaluate if these change
case (ALUctl)
0: ALUOut <= A&B;
default: ALUOut ¡= 0;
endcase
end
endmodule
```

3.2 Atividade 2

Nesta atividade, você deverá verificar o funcionamento do projeto da ULA através de sua simulação. Para isto, proceda como a seguir:

1. Crie um novo projeto: *File* $\longrightarrow$ *New Project Wizard*. Clique em *Next*.
2. Selecione o seu diretório de trabalho e entre com o nome "ULA". Selecione seu arquivo com o nome "ULA.v", clique em *Add* e em *Next*.
3. Selecione a família Cyclone II e o dispositivo do módulo DE2 (EP2C35F672C6N) e clique em *Next* e *Finish*.

4. Compile o projeto selecionando *Processing* $\longrightarrow$*Start Compilation* ou o ícone na barra de ferramentas.

5. Crie um arquivo de formas de onda selecionando File $\longrightarrow$ New $\longrightarrow$University Program VWF. Adicione os sinais de entrada e saída do projeto selecionando com o botão direito do mouse as opções *Insert* e *Insert Node or Bus* no menu suspenso, então *Node Finder* e, com a opção *Pins: all*, clique no botão *List* para listá-los. Ao colocar os sinais, selecione a base como binário, hexadecimal ou decimal em função do teste que será realizado.

6. Defina o tempo de duração da simulação (*Edit* $\longrightarrow$ *End Time*) e o tamanho da grade (*Edit* $\longrightarrow$*Grid Size*) e selecione *Fit in Window*, ou tecle CTRL-W, para visualizar toda a janela de simulação.

7. Para definir valores apropriados aos sinais de entrada, selecione uma região do sinal e a ferramenta *Arbitrary Value* no menu à esquerda. Os operandos A e B e a função F devem ser definidos em regiões semelhantes para produzir os resultados esperados.

8. Salve o arquivo de simulação e selecione *Simulation* $\longrightarrow$ *Run Functional Simulation.*

3.3 Atividade 3

A atividade 3 consiste em adicionar, ao código da atividade 1, as seguintes funções à ULA: OR, NOR, SUB e LUI, de acordo com as definições do conjunto de instruções do MIPS. Compile e verifique o funcionamento correto dessas instruções.

3.4 Atividade 4

A atividade 4 consiste em verificar o funcionamento da ULA utilizando o simulador ModelSim. Para isso, o projeto da ULA, como no caso do processador mais tarde, será considerado como um COMPONENTE de um projeto de testes, ilustrando a facilidade em utilizar o mesmo ambiente de desenvolvimento também na simulação. Os passos para a simulação no ModelSim estão delineados no documento "Tutorial: Simulação com o ModelSim".

4 Relatorio

Deve ser entregue uma pasta compactada contendo os seguintes arquivos :

- Arquivo fonte e código compilado (módulos e testbench), sem erros de execução.
- Arquivos de entrada e saída para uma execução exemplo.
- Relatório com os resultados das actividades 1-4 do laboratório.

Arquivos com vírus não serão avaliados. Todos os trabalhos plagiados/copiados serão desconsiderados (zerados).