Laboratório 4
- CPU MIPS Multiciclo –

GRUPO 6
Dayanne Fernandes da Cunha, 13/0107191
Lucas Mafra Chagas, 12/0126443
Marcelo Giordano Martins Costa de Oliveira, 12/0037301
Lucas Junior Ribas, 16/0052289
Caio Nunes de Alencar Osório, 16/0115132
Diego Vaz Fernandes, 16/0117925

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CiC 116394 - OAC - Turma A

1. Objetivos

- Treinar o aluno com a linguagem de descrição de *hardware Verilog*;
- Familiarizar o aluno com a plataforma de desenvolvimento *FPGA DE2* da *Altera* e o software *QUARTUS II*;
- Desenvolver a capacidade de análise e síntese de sistemas digitais usando uma Linguagem de Descrição de *Hardware*;
- Apresentar ao aluno a implementação de uma *CPU MIPS Multiciclo*.

2. Ferramentas

Todos os códigos escritos neste laboratório podem ser encontrados no repositório `https://github.com/Dayof/OAC172` do *GitHub*.

- FPGA DE2 da Altera
- QUARTUS-II
- Verilog HDL

3. Exercício 2. Análise do processador Multiciclo

4. Exercício 3. Teste do funcionamento das instruções da *ISA*

5. Exercício 4. Software de lançamento de bola de canhão na *FPGA*

6. Exercício 5. Demonstração dos cenários

7. Exercício 6. Novas instruções usando a *ISA MIPS*

Na Tabela 1 mostra a instrução *MUL* do tipo R que foi inserida na *ISA MIPS* Multiciclo.

INSTRUÇÃO	OPCODE (6)	RS (5)	RT (5)	RD (5)	SHAMT (5)	FUNCT (6)
MUL $t1, $t2, $t3	000000	x	x	x	00000	000010

Table 1. Componentes da instrução *MUL*.

Para preencher a Tabela 1 foi usada a informação que as instruções do tipo R sempre possuem o *OPCODE* em zero. A ordem dos componentes de *MUL* é *MUL RD RS RT*. *RS* e *RT* se referem aos argumentos da operação, no caso, $t2 e $t3 respectivamente e *RD* para o registrador destino, ou seja, $t1. Como *MUL* não é uma instrução que usa operações *SHIFT* então este campo também permanece com zeros. O componente *FUNCT* foi preenchido de acordo com a fonte [MIPS32^{TM}].

Na Tabela 1 mostra a instrução *JALR* do tipo R que foi inserida na *ISA MIPS* Multiciclo.

Instrução	OPCODE (6)	RS (5)	RT (5)	RD (5)	SHAMT (5)	FUNCT (6)
JALR $t1	000000	x	00000	11111	00000	001001
JALR $t1, $t2	000000	x	00000	x	00000	001001

Table 2. Componentes da instrução *JALR*.

Para preencher a Tabela 2 também foi usada a informação que as instruções do tipo *R* sempre possuem o *OPCODE* em zero. A ordem dos componentes de *JALR* é *JALR RD RS*. Na instrução com apenas *$t1* o componente *RD* é preenchido com 11111 para representar o *$ra*. Em ambas instruções da Tabela 2 o componente *RT* não é utilizado. E como *JALR* não é uma instrução que usa operações *SHIFT* então este campo também permanece com zeros. O componente *FUNCT* foi preenchido de acordo com a fonte [MIPS32^{TM}].

Para implementar estas instruções os arquivos *Parametros.v*, *ALUControl.v*, *Datapath_MULT.v* e *Control_MULT.v* foram modificados.

7.1. Parêmetros

O *OPCODE* de *MUL* e *JALR* não precisam de nenhuma adição no arquivo já que são do tipo R.

Os componentes *FUNCT* foram adicionados no arquivo como é mostrado na Figura 1.

```
87          FUNMUL      = 6'b000010,        // 2017/2
88          FUNJALR     = 6'b001001,        // 2017/2
```

Figure 1. Parâmetros para o componente *FUNCT* de cada instrução.

Os parâmetros para a máquina estado do multiciclo foram adicionados conforme a Figura 2.

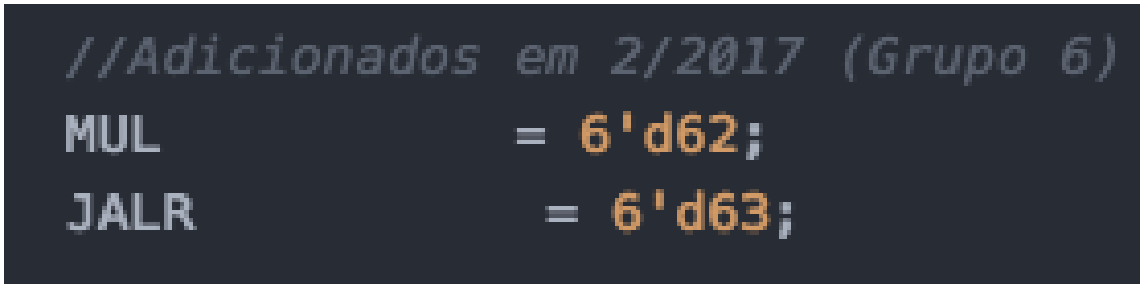

Figure 2. Parâmetros para a máquina de estados multiciclo.

A operação *MUL* não foi adicionada no arquivo pois o *OPMULT* implementada no arquivo *ALU.v* já implementa o resultado esperado para *MUL*. No caso seria colocar em *HI* e *LO* o resultado da operação *iA * iB*, sendo que em *HI* estaria os 32 bits mais significativos e *LO* os 32 menos significativos. Como *oALUresult* recebe *LO* em *OPMULT* então já temos o esperado para *MUL*, sendo *oALUresult* nossa representação do *RD*.

7.2. Bloco de Controle da ULA

Conforme explicado acima sobre a operação *MUL*, foi adicionado no arquivo *ALUControl.v* o mapeamento da *FUNMUL* para *OPMULT* já que a versão de teste para *OPMULT* com *oALUresult* recebendo *LO* já implementa a saída esperada para um *OPMUL* (Figura 3).

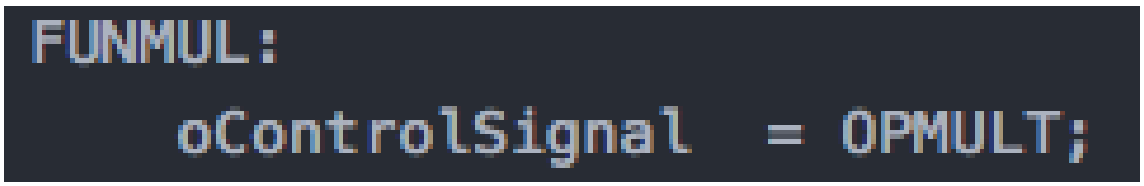

Figure 3. Mapeamento de *FUNMUL* para *OPMULT*.

7.3. Caminho de dados

7.4. Bloco de controle

7.5. Teste das novas instruções

References

[MIPS32™] MIPS32™. Mips32™ architecture for programmers volume ii: The mips32tm instruction set. `https://www.cs.cornell.edu/courses/cs3410/2008fa/MIPS_Vol2.pdf`. [Online; acessado 15-Novembro-2017].