Universidade Federal de Viçosa

Campus Florestal

Implementação Simplificada do Caminho de dados MIPS

Lucas Gomes Soares, 3003

Paulo Roberto Oliveira Brandão Alves, 3008

Florestal/MG, 2018

Implementação Simplificada do Caminho de dados MIPS

Lucas Gomes Soares, 3003

Paulo Roberto Oliveira Brandão Alves, 3008

Este relatório apresenta a metodologia e os resultados obtidos no 2º trabalho prático da disciplina de Organização de Computadores I, orientado pelo Professor José Augusto Nacif.

Florestal/MG, 2018

RESUMO

O trabalho consiste na síntese de uma versão simplificada do caminho de dados do MIPS, que está representado na figura abaixo, utilizando uma field-programmable gate array (FPGA). A FPGA é um dispositivo lógico programável que suporta a implementação de circuitos digitais. Sua configuração é geralmente especificada usando uma linguagem de descrição de hardware (HDL), neste caso será feita em Verilog. A comprovação de resultados foi feita a partir da simulação de ondas.

Palavras-chave: FPGA, HDL, VERILOG, Caminho de dados, MIPS.

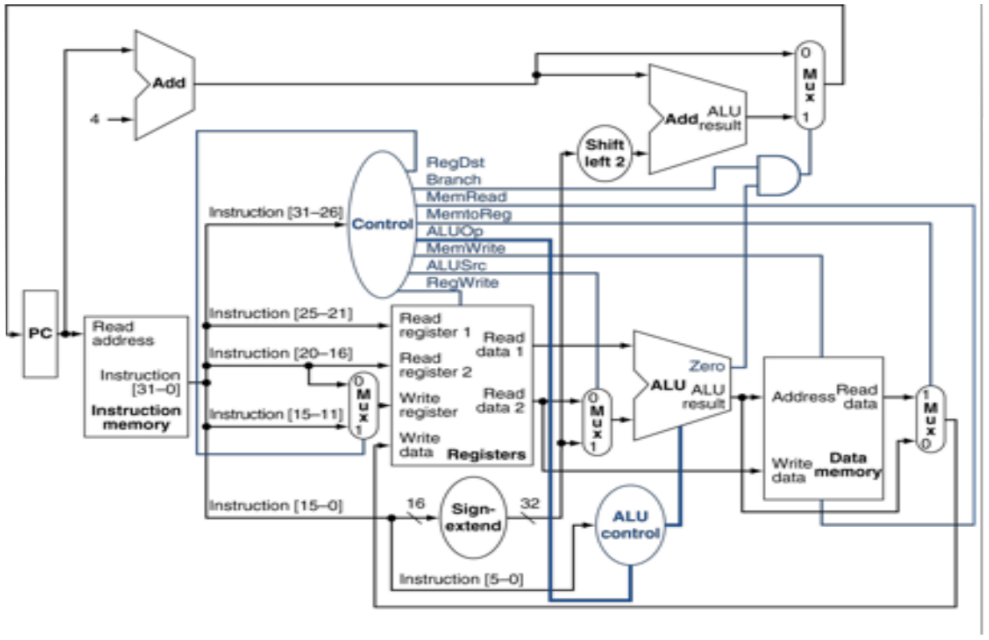
SUMÁRIO

INTRODUÇÃO

O Caminho de dados representa o conjunto por onde passa cada tipo de instrução, cada uma realizando uma sequência de ações única. Podemos observar na representação abaixo que as 2 primeiras etapas são idênticas para todas as instruções:

- Buscar a instrução na memória na posição indicada pelo *Program Counter* (PC);
- Ler os registradores indicados na instrução (um ou dois);

Após isso, ocorre u00ma mudança que varia de acordo com cada instrução.



1.0 - Caminho de dados MIPS

METODOLOGIA

INSTRUÇÕES

As instruções implementadas são descritas na categorização abaixo:

TIPO R		
Nome	Opcode	Ação
add	ADD rd,rs,rt	rd = rs + rt
substract	SUB rd,rs,rt	rd = rs - rt
and	AND rd,rs,rt	rd = rs & rt
or	OR rd,rs,rt	rd = rs | rt
Set on less Than	SLT rd,rs,rt	rd = rs < rt
LOAD STORE		
Load Word	LW rt, offset (rs)	rt = *(int*)(offset+rs)
STORE WORD		
Store Word	SW rt, offset(rs)	*(int*)(offset+rs)=rt
BRANCH EQUAL		
Branch On Equal	BEQ rs,rt,offset	if(rs==rt) pc+=offset*4

LEITURA DE OPERANDOS

Cada instrução possui 32 bits e a separação de cada grupo de binários é que define o tipo de instrução e a ação a ser executada. Abaixo, uma exemplificação da leitura de operandos das instruções implementadas no projeto.

TIPO R

O	rs	rt	rd	shamt	funct
31:26	25:21	20:16	15:11	10:6	5:0

LOAD/STORE

35 or 43	rs	rt	address
31:26	25:21	20:16	15:0

BRANCH

4 (opcode)	rs	rt	address
31:26	25:21	20:16	15:0

MIPS:CICLO DE EXECUÇÃO

Para implementação, o ciclo de execução foi definido em 5 etapas:

I. **Program Counter (PC) :** *endereço da instrução a ser buscada na memória. Atualiza o PC para receber a próxima instrução.*
II. **Leitura dos Registradores:** *busca de operando nos registradores.*
III. **Usar a Unidade Lógica Aritmética (ULA) para calcular:**
    A. *Resultado aritmético*
    B. *Endereço da memória para load/store*
    C. *Endereço de desvio condicional (branch)*
IV. **Acesso a memória de dados** (load/store): Uso da ULA para calcular o endereço. Em seguida, acessar a memória de dados para fazer um LW (Load Word) ou escrever o valor com SW (Store Word).
V. **Escrever resultado no registrador destino**

RESULTADOS

O caminho de dados foi feito utilizando a linguagem de descrição de hardware VERILOG. Como teste, foi compilado o arquivo teste.v, gerado o arquivo de ondas MIPS.vcd e testado no software GTKWave Analyzer v3.389 apresentando as seguintes simulações de onda a seguir:


Signals
Waves
Time
3 sec
6 sec
9 sec
Jump =0
MemRead =0
MemWrite =1
MemtoReg =0
RegDst =0
RegWrite =0
opcode[5:0] =101011
ALUControl[3:0] =0010
ALUResult[31:0] =00000000
value1[31:0] =00000000
value2[31:0] =00000000
zero =0
ALUCtl[3:0] =0010
ALUOp[1:0] =00
FuncCode[5:0] =110000
ALUResult[31:0] =00010010
addressShift[31:0] =00000000
addressSomador[31:0] =00010010
addressDesvio[31:0] =00000001
controland =0
in0[31:0] =00010010
in1[31:0] =00010010
DataMemory[31:0] =8
addressFin[31:0] =00000001
addressFinal[31:0] =xxxxxxxx
addressShiftJ[27:0] =10011010


Signals
Waves
Time
3 sec
6 sec
9 sec
clk =1
addressInPC[31:0] =20
InstructionOut[31:0] =10101110
ProxInstruction[31:0] =00010010
ALUOp[1:0] =00
ALUSrc =1
Branch =0
Jump =0
MemRead =0
MemWrite =1
MemtoReg =0
RegDst =0
RegWrite =0
opcode[5:0] =101011
ALUControl[3:0] =0010
ALUResult[31:0] =00000000
value1[31:0] =00000000
value2[31:0] =00000000
zero =0
ALUCtl[3:0] =0010
ALUOp[1:0] =00
FuncCode[5:0] =110000
ALUResult[31:0] =00010010
addressShift[31:0] =00000000
addressSomador[31:0] =00010010
addressDesvio[31:0] =00000001