# 计算机组成原理实验报告

19373106 裴宝琦

## 一、CPU设计方案综述

### （一）总体设计概述

本CPU为Logisim实现的单周期MIPS - CPU，支持的指令集包含{addu,subu,ori,lui,beq,nop,lw,sw,j}。为了实现这些功能，CPU主要包含了IFU、GRF,DM,IM,ALU,control等模块，有些模块的下层还包含一些用来计算的子模块电路。

### （二）关键模块定义

#### 1. GRF

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RESET | I | 异步复位信号 |
| WE | I | 为1时，时钟上沿将WD写入对应寄存器 |
| WD | I | 回写寄存器值 |
| A1 | I | 第一个寄存器编号 |
| A2 | I | 第二个寄存器编号 |
| A3 | I | 回写寄存器编号 |
| CLK | I | 时钟信号 |
| RD1 | O | 第一个寄存器读出的值 |
| RD2 | O | 第二个寄存器读出的值 |
| PC4 | I | PC+4的值 |
| Jal | I | 判断当前指令是否是jal |
| Reg | O | 回写rs寄存器中值（针对JR） |

#### 2. DM

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Address | I | 地址输入 |
| WD | I | 数据写入 |
| WE | I | 为1时，时钟上沿将WD写入对应地址单元 |
| CLK | I | 时钟信号 |
| RE | I | 异步复位信号 |
| LOAD | I | 为1时，时钟上沿读出数据 |
| RD | O | 读出数据 |
| PC4 | I | Pc+4 |

#### 3. ALU

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| In1 | I | 第一个操作数 |
| In2 | I | 第二个操作数 |
| ALUop | I | 控制信号 |
| ZERO | O | 判断两数是否相等 |
| Result | O | 输出操作结果 |

#### 4.IFU

这个模块里包括了PC,IM等模块以及相关逻辑，该模块的目的是在每个周期都输出正确的指令。

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Imm | I | 针对beq的imm\_ext |
| J\_addr | I | 28位j指令 |
| CLK | I | 时钟信号 |
| Branch | I | 判断是否转到beq所指向的地址 |
| j/jr/jal | I | 判断是否是对应指令 |
| Reset | I | 异步复位信号 |
| Instruction | O | 输出指令机器码 |
| PC4 | O | PC+4 |
| Reg | I | 针对jr指令写PC |

#### 5.MUX群

有三个MUX 针对alu/WD/rd。

第一个决定立即数或者寄存器值写1

第二个决定DM输出/ALU输出/PC+4写1

第三个决定rt/rd作为接口

#### 6.control

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Opcode | I | 操作码 |
| func | I | 对应的功能位 |
| RegDst | O | 失效选rt，有效写rd |
| RegWrite | O | 有效时把数据写入对应寄存器 |
| ALUSrc | O | 无效输入R[rt]，有效选择SignExt输出 |
| Branch | O | 判断是否是beq |
| Memread | O | 从DM中读数据 |
| MemWrite | O | 往DM中写数据 |
| Wdwrite | O | 写DM/ALU/PC+4 |
| ALUOp | O | ALU控制电路 |
| Exp | O | 扩展位方式 |
| j/jal/jr | O | 判断 |

### （三）重要机制实现方法

#### 1. PC值的确定

通过两个多路选择器确定下一个周期的PC值，一个选择器选择PC+4和PC+4+imm\_exp，一个选择器选择是否执行j指令通路。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | addu | subu | ori | lw | sw | beq | lui | nop | j | jal | jr |
| opcode | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 | 000000 | 000010 | 000011 | 000000 |
| func | 100001 | 100011 |  |  |  |  |  |  |  |  | 001000(last) |
| RegDst | 1 | 1 | 0 | 0 | 0 | 0 | 0 | x | x | 0 | 0 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | x | 1 | 0 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 | x | x | 0 | 0 |
| PCSrc | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCJump | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| MemRead | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Wdwrite | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALUctrl | + | - | or | + | + | - | + | x |  | 2 | 0 |
|  | 0 | 1 | 000002 | 000000 | 000000 | 000001 | 000000 | 000000 |  |  | 0 |
| EXP | 000000 | 000000 | 000000 | 000001 | 000001 | 000001 | 000002 |  |  | 000000 | 0 |
| jal |  |  |  |  |  |  |  |  |  | 000001 | 0 |
| jr |  |  |  |  |  |  |  |  |  |  | 1 |

#### 2. 控制器真值表及一些说明

## 二、测试方案

## ori $a0,$0,0x100

## ori $a1,$a0,0x123

## lui $a2,456

## lui $a3,0xffff

## ori $a3,$a3,0xffff

## addu $s0,$a0,$a2

## addu $s1,$a0,$a3

## addu $s4,$a3,$a3

## subu $s2,$a0,$a2

## subu $s3,$a0,$a3

## sw $a0,0($0)

## sw $a1,4($0)

## sw $a2,8($0)

## sw $a3,12($0)

## sw $s0,16($0)

## sw $s1,20($0)

## sw $s2,24($0)

## sw $s3,44($0)

## sw $s4,48($0)

## lw $a0,0($0)

## lw $a1,12($0)

## sw $a0,28($0)

## sw $a1,32($0)

## ori $a0,$0,1

## ori $a1,$0,2

## ori $a2,$0,1

## beq $a0,$a1,loop1

## beq $a0,$a2,loop2

## loop1: sw $a0,36($t0)

## loop2: sw $a1,40($t0)

## jal loop3

## sw $s5,64($t0)

## ori $a1,$a1,4

## jal loop4

## loop3:sw $a1,56($t0)

## sw $ra,60($t0)

## ori $s5,$s5,5

## jr $ra

## loop4: sw $a1,68($t0)

## sw $ra,72($t0)

这组数据应该考虑了绝大多数的情况。

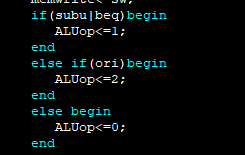
## 三、思考题

1.根据你的理解，在下面给出的DM的输入示例中，地址信号addr位数为什么是[11:2]而不是[9:0]？这个addr信号又是从哪里来的？

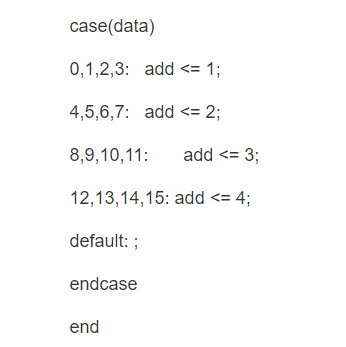
Addr信号从ALU的计算结果得来，结果是以字节为单位的，而dm的取值是以字为单位的，因此要从2位开始取，相当于乘4。

2.思考Verilog语言设计控制器的译码方式，给出代码示例，并尝试对比各方式的优劣。

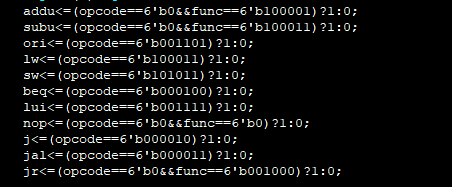
第一种 if-else



第二种 case



第三种 assign



第一种第二种逻辑都比较清晰，取值少时用前者，多时用后者。

第三种写起来更方便。

3. 在相应的部件中，**reset的优先级**比其他控制信号（不包括clk信号）都要**高**，且相应的设计都是**同步复位**。清零信号reset所驱动的部件具有什么共同特点？

首先，它针对IFU,GRF,DM

这些部件中都包含寄存器，存在当复位时需要把寄存器归位的操作

4. C语言是一种弱类型程序设计语言。C语言中不对计算结果溢出进行处理，这意味着C语言要求程序员必须很清楚计算结果是否会导致溢出。因此，如果仅仅支持C语言，MIPS指令的所有计算指令均可以忽略溢出。 请说明为什么在忽略溢出的前提下，addi与addiu是等价的，add与addu是等价的。提示：阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》中相关指令的Operation部分 。

因为在指令说明中，无符号加（ADDIU）的意义就是不考虑溢出，所以没有溢出时它和ADDI完全等价，后者同理。

5. 根据自己的设计说明单周期处理器的优缺点。

优点：优点是模块化很强，可以方便的增添指令，各部分各司其职，不存在冲突，正确性高，逻辑比较清晰，好操作。

缺点：大部分指令并不能用到所有模块，所有指令的执行时间一样，和流水线相比，浪费空间，时间。