**实验报告**

2020年06月22日 成绩：---------

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 刘爱兵 | 学号 | 19114801 | 班级 | 19计科2班 |
| 专业 | 计算机科学与技术 | 课程名称 | 计算机组成原理 实验 | 任课教师 | 吴云志 |
| 实验序号 | 08 | 实验名称 | R-I型指令的CPU设计 | | |
| 实验时间 | 2021.06.22 | 机位号 | 18 | 实验设备号 | 21 |

1. **实验目的与要求**

1、实验目的

掌握MIPS R型和I型指令的综合数据通路设计

掌握数据流的多路选通控制方法

掌握取数指令lw和存数指令sw指令的寻址方式及其有效地址产生方法

实现MIPS的部分 I型和R型指令的功能

2、实验内容与原理

实验七的基础上，再行实现MIPS的6条I型指令：

4条立即数寻址的运算和传送指令

2条相对寄存器寻址的存数和取数指令。

（1）MIPS的I型立即数寻址指令及数据通路

MIPS I型立即数寻址指令格式及编码

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段** | **OP** | **rs** | **rt** | **imm** | **功能描述** |
| **位数** | **6** | **5** | **5** | **16** |
| **汇编助记符** | **编码** | | | |
| **addi rt, rs, imm** | **001000** | **rs** | **rt** | **imm** | **算术加：rs + imm→rt** |
| **andi rt, rs, imm** | **001100** | **rs** | **rt** | **imm** | **逻辑与：rs & imm→rt** |
| **xori rt, rs, imm** | **001110** | **rs** | **rt** | **imm** | **逻辑异或：rs⊕imm→rt** |
| **sltiu rt, rs, imm** | **001011** | **rs** | **rt** | **imm** | **无符号数小于则置位：**  **if (rs < imm) rt=1 else rt=0** |

I型与R型指令有明显不同：

没有rd寄存器，使用rt作为目的寄存器；

源操作数有一个为立即数，位于指令的低16位。

解决目的寄存器的可选性：

设置一个二选一数据选择器，控制信号为rd\_rt\_s：

当rd\_rt\_s=0，将指令的rd字段送写地址W\_Addr；

当rd\_rt\_s=1，将指令的rt字段送写地址W\_Addr。

Verilog语句如下：

assign W\_Addr = (rd\_rt\_s) ? rt : rd;

扩展16位的立即数imm

设置一位imm\_s来控制这两种扩展：

imm\_s=1，符号扩展；

imm\_s=0，0扩展。

Verilog语句如下：

assign imm\_data=(imm\_s) ?{{16{imm[15]}},imm} :{{16{1’b0}},imm};

ALU的输入数据B端的数据选择

方法：设置二选一数据选择器（控制信号为rt\_imm\_s）

当rt\_imm\_s=0，将寄存器堆的B端口读出数据R\_Data\_B送ALU的B端

当rt\_imm\_s=1，将扩展好的立即数imm\_data送ALU的B输入端

Verilog语句如下:

assign ALU\_B = (rt\_imm\_s) ? imm\_data :R\_Data\_B;

改造的数据通路



2.I型取数/存数指令及其数据通路

MIPS I型存储器访问指令格式及编码

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段** | **OP** | **rs** | **rt** | **offset** | **功能描述** |
| **位数** | **6** | **5** | **5** | **16** |
| **汇编助记符** | **编码** | | | |
| **lw rt, offset(rs)** | **100011** | **rs** | **rt** | **offset** | **取数:(rs+offset)→rt** |
| **sw rt, offset(rs)** | **101011** | **rs** | **rt** | **offset** | **存数:rt→(rs+offset)** |

改进数据通路，实现两条访存指令

添加一个数据存储器RAM，存放指令访问的数据必须添加吗？

有效地址EA的计算：ALU来实现，置rt\_imm\_s=1，imm\_s=1。

为何是带符号扩展？

将ALU的输出直接送存储器地址端口

Verilog描述： assign Mem\_Addr = ALU\_F

改进数据通路，实现两条访存指令

存储器读出的数据：

alu\_mem\_s=0，则将ALU的输出送寄存器堆的写数据端口

alu\_mem\_s=1，则将存储器的读出数据送寄存器堆的写数据端口。

Verilog描述如下：assign W\_Data=alu\_mem\_s ?M\_R\_Data :ALU\_F;

存储器的写入数据

将寄存器堆的B端口数据直接送至存储器的写数据端口

Verilog描述：assign M\_W\_Data = R\_Data\_B;

新的完整的R-I型指令数据通路：



R-I型指令的控制流

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **指令** | **rd\_rt\_s** | **imm\_s** | **rt\_imm\_s** | **alu\_mem\_s** | **ALU\_OP** | **Write\_Reg** | **Mem\_Write** |
| **add rd,rs,rt** | **0** | **——** | **0** | **0** | **100** | **1** | **0** |
| **sub rd,rs,rt** | **0** | **——** | **0** | **0** | **101** | **1** | **0** |
| **and rd,rs,rt** | **0** | **——** | **0** | **0** | **000** | **1** | **0** |
| **or rd,rs,rt** | **0** | **——** | **0** | **0** | **001** | **1** | **0** |
| **xor rd,rs,rt** | **0** | **——** | **0** | **0** | **010** | **1** | **0** |
| **nor rd,rs,rt** | **0** | **——** | **0** | **0** | **011** | **1** | **0** |
| **sltu rd,rs,rt** | **0** | **——** | **0** | **0** | **110** | **1** | **0** |
| **sllv rd,rs,rt** | **0** | **——** | **0** | **0** | **111** | **1** | **0** |
| **addi rt,rs,imm** | **1** | **1** | **1** | **0** | **100** | **1** | **0** |
| **andi rt, rs, imm** | **1** | **0** | **1** | **0** | **000** | **1** | **0** |
| **xori rt, rs, imm** | **1** | **0** | **1** | **0** | **010** | **1** | **0** |
| **sltiu rt, rs, imm** | **1** | **0** | **1** | **0** | **110** | **1** | **0** |
| **lw rt, offset(rs)** | **1** | **1** | **1** | **1** | **100** | **1** | **0** |
| **sw rt, offset(rs)** | **——** | **1** | **1** | **——** | **100** | **0** | **1** |

3. I型指令的时序

立即数寻址的I型指令，执行的时序同R型指令：

在clk的上跳沿，指令存储器执行读操作

在clk正脉冲内，读出的指令经过译码、执行运算

在clk的下跳沿，将运算结果打入目的寄存器rd或者rt

对于取数/存数指令，执行的时序：

在clk的上跳沿，**指令存储器**执行读操作

在clk正脉冲内，指令译码、计算有效地址EA、**读写数据存储器**

在clk的下跳沿，lw指令将读出的存储器数据打入目的寄存器rt

所以：数据存储器的读和写访问所使用的clka上跳沿必须迟于CPU的单周期脉冲clk的上跳沿；

**解决办法：**数据存储器的clka使用10MHz的时钟源——意味着在CPU的clk正脉冲期间内，只要读写信号有效，立即可以执行读写。

4.指令测试

测试代码

汇编后机器码

将上述机器指令码填入到和指令存储器模块ROM\_B相关联的\*.coe文件中，也可以调用\*.coe的生成软件来完成。

在和数据存储器模块RAM\_B相关联的\*.coe文件中，可以随意填入一些数据

#baseAddr 0000

xori $1, $0, 0x1234; #$1=0000\_1234

addi $2, $0, 0x6789; #$2=0000\_6789

addi $3, $0, -0x7000; #$3=FFFF\_9000

xori $4, $0, 0x0010; #$4=0000\_0010

sllv $5, $2, $4; #$5=6789\_0000

or $6, $1, $5; #$6=6789\_1234

sllv $7, $3, $4; #$7=9000\_0000

add $8, $2, $6; #$8=6789\_79BD

sub $9, $2, $1; #$9=0000\_5555

sub $10, $1, $2; #$10=FFFF\_AAAB

addi $11, $3, 0x7FFF; #$11=0000\_0FFF

addi $12, $3, -0x8000; #$12=FFFF\_1000

andi $13, $10, 0xFFFF; #$13=0000\_AAAB

sltiu $14, $2, 0x6788; #$14=0000\_0000

sltiu $15, $2, 0x678A; #$15=0000\_0001

sw $11, 0($4); #mem(0000\_0010)=0000\_0FFF

sw $12, 20($0); #mem(0000\_0014)=FFFF\_1000

sw $13, 16($4); #mem(0000\_0020)=0000\_AAAB

sw $14, 20($4); #mem(0000\_0024)=0000\_0000

lw $16, 16($0); #$16=mem(0000\_0010)=0000\_0FFF

lw $17, 4($4); #$17=mem(0000\_0014)=FFFF\_1000

or $18, $16, $17; #$18=FFFF\_1FFF

lw $19, 16($4); #$19=mem(0000\_0020)=0000\_AAAB

lw $20, 20($4);#$20=mem(0000\_0024)=0000\_0000

nor $21, $19, $20; #$21=FFFF\_5554

lw $22, -0x10($4); #$22=mem(0000\_0000)，譬如8888\_8888

lw $23, -0x0C($4); #$23=mem(0000\_0004)，譬如9999\_9999

sltu $24, $22, $23 #$24=? ，譬如=0000\_0001

38011234, 20026789,

20039000, 38040010,

00822804, 00253025,

00833804, 00464020,

00414822, 00225022,

206b7fff, 206c8000,

314dffff, 2c4e6788,

2c4f678a, ac8b0000,

ac0c0014, ac8d0010,

ac8e0014, 8c100010,

8c910004, 02119025,

8c930010, 8c940014,

0274a827, 8c96fff0,

8c97fff4, 02d7c02b

5.实验验证

验证R-I型指令集CPU模块的顶层模块



实验要求

在实验七的基础上，编写一个CPU模块

实现实验七的8条R型指令

实现新的6条I型指令

编写一个实验验证的顶层模块

实验室任务：

配置管脚：见下表

生成\*.bit文件，下载到HDU-XL-01实验板中。

完成板级调试。

撰写实验报告。

|  |  |  |  |
| --- | --- | --- | --- |
|  | **信号** | **配置设备管脚** | **功能说明** |
| **输入信号** | **rst** | **1个按钮** | **清零** |
| **clk** | **1个按钮** | **模拟时钟输入** |
| **ALU\_SW** | **1个按钮** | **按住时，数码管显示 ALU 运算结果；** |
| **Data\_SW** | **1个按钮** | **按住时，数码管显示存储器读出数据** |
| **输出信号** | **LED[31:0]** | **32个LED灯** | **显示字数据或标志位** |
|  | **数码管** | **8段数码管** | **显示的内容由ALU\_SU和Data\_SW控制** |

4、实验步骤

在Xilinx ISE/Vivado中创建工程，编源码，然后编译、综合

编写激励代码，观察仿真波形，直至验证正确

在PC机上打开工程文件，进行管脚配置。

生成编程文件\*.bit，下载到板卡中。

板级实验。

1. **实验设计与程序代码**
2. 模块设计说明

（描述这个实验的设计方案，分几个模块，各模块的功能，各模块之间的连接关系，可附图）

1. 实验程序源代码及注释等

（实验各个模块的代码，包含功能注释）

|  |
| --- |
|  |

1. **实验仿真**
2. 仿真代码（仿真源代码）

|  |
| --- |
| Fetch\_Test.v |

1. 仿真波形（运行仿真时波形截图）
2. 仿真结果分析（对仿真波形进行分析）

这是一个单周期MIPS CPU，所有指令在一个周期内全部完成。

在clk的上跳沿，执行取指令操作；在clk的下跳沿，更新PC值。

1. **电路图**

（开发工具中显示的电路模块图）

1. **引脚配置**

（引脚约束文件的内容，描述主要配置情况）

# 开启比特流压缩，优化 .bit 文件大小

|  |
| --- |
|  |

5、思考与探索：必做（1）

（1）将各条指令执行的结果和标志记录到表中，分析结果正确与否？如果不正确，请分析原因。

（2）I型指令lui rt, imm将立即数imm装入rt寄存器的高16位，低16位清零。它的OP编码为6’b001111，rs字段为5’b00000，试着实现该指令

（3）说说你在实验中碰到了哪些问题，你是如何解决的？