**计算机组成原理**

**课程设计报告**

**学 号\_\_\_\_\_\_\_\_\_\_20090121\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_\_\_陈立标\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_\_\_\_魏坚华\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_\_\_2022年7月4日\_\_\_\_**

**成绩评价表**

|  |  |  |
| --- | --- | --- |
| **报告内容** | **报告结构** | **报告最终成绩** |
| **□丰富正确**  **□基本正确**  **□有一些问题**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** |  |
| **报告与Project功能一致性** | **报告图表** | **总体评价** |
| **□完全一致**  **□基本一致**  **□基本不一致** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** |  |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

[1 总体数据通路结构设计图 1](#_Toc106132553)

[2 模块定义 1](#_Toc106132554)

[2.1 IFU模块定义 1](#_Toc106132555)

[2.1.1 基本描述 1](#_Toc106132556)

[2.1.2 模块接口 1](#_Toc106132557)

[2.1.3 功能定义 2](#_Toc106132558)

[2.2 GetCode模块定义 2](#_Toc106132559)

[2.2.1 基本描述 2](#_Toc106132560)

[2.2.2 模块接口 2](#_Toc106132561)

[2.2.3 功能定义 3](#_Toc106132562)

[2.3 Controller模块定义 3](#_Toc106132563)

[2.3.1 基本描述 3](#_Toc106132564)

[2.3.2 模块接口 3](#_Toc106132565)

[2.3.3 功能定义 4](#_Toc106132566)

[2.4 RegFile模块定义 4](#_Toc106132567)

[2.4.1 基本描述 4](#_Toc106132568)

[2.4.2 模块接口 4](#_Toc106132569)

[2.4.3 功能定义 5](#_Toc106132570)

[2.5 Extender模块定义 5](#_Toc106132571)

[2.5.1 基本描述 5](#_Toc106132572)

[2.5.2 模块接口 5](#_Toc106132573)

[2.5.3 功能定义 5](#_Toc106132574)

[2.6 ALU模块定义 5](#_Toc106132575)

[2.6.1 基本描述 5](#_Toc106132576)

[2.6.2 模块接口 5](#_Toc106132577)

[2.6.3 功能定义 6](#_Toc106132578)

[2.7 DM模块定义 6](#_Toc106132579)

[2.7.1 基本描述 6](#_Toc106132580)

[2.7.2 模块接口 7](#_Toc106132581)

[2.7.3 功能定义 7](#_Toc106132582)

[3 指令测试 7](#_Toc106132583)

[3.1 指令说明 7](#_Toc106132584)

[3.2 测试代码 8](#_Toc106132585)

[3.3 测试结果 10](#_Toc106132586)

[4 拓展指令设计 12](#_Toc106132587)

[4.1 设计思路 12](#_Toc106132588)

[4.2 指令设计 12](#_Toc106132589)

[4.3 测试结果 12](#_Toc106132590)

[5 总结 13](#_Toc106132591)

# 总体数据通路结构设计图

总体数据通路以及控制器设置如图1所示，完成了MIPS-LITE1共13条指令以及拓展指令BGTZ。

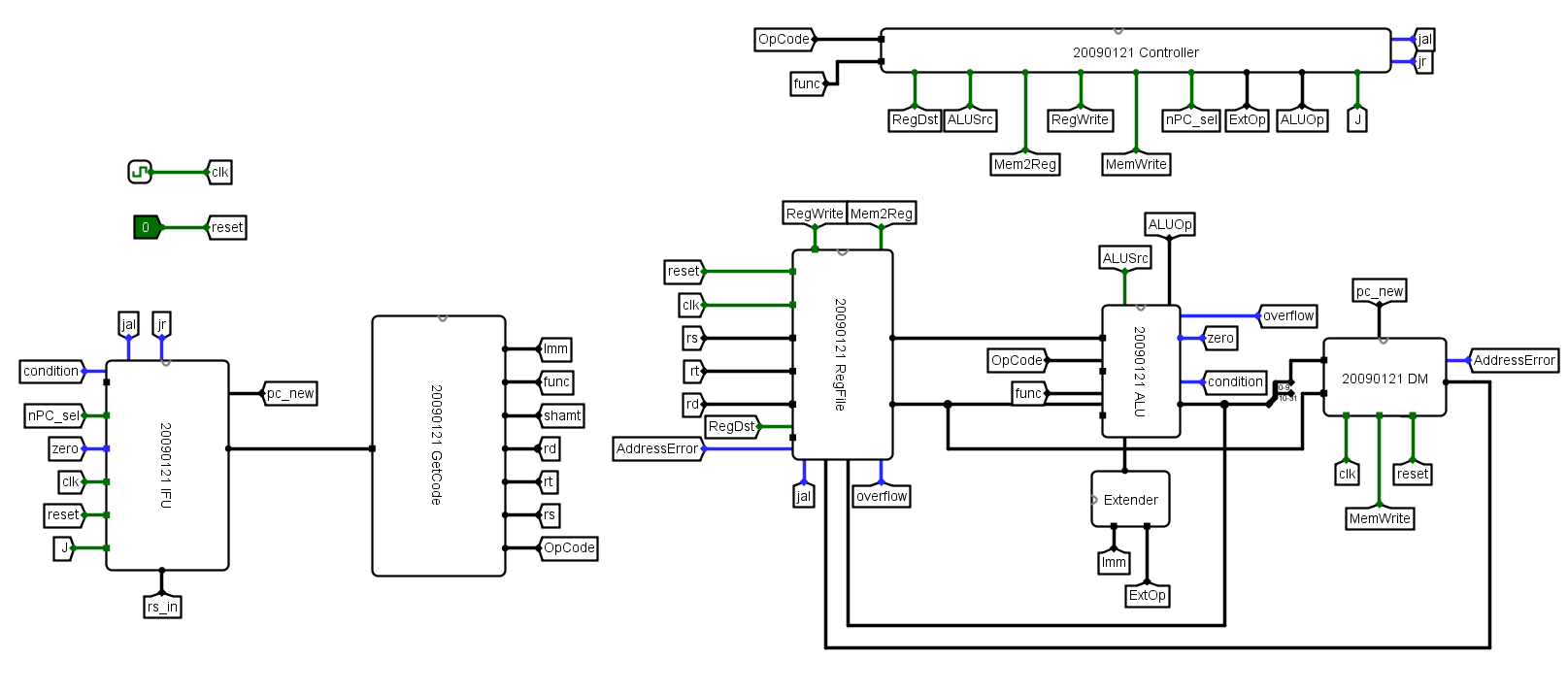


图1 总体数据通路结构

# 模块定义

## IFU模块定义

### 基本描述

IFU主要完成各种情况下的取指令功能：顺序取指令;BEQ、BGTZ相对跳转取指令;J、JR、JAL绝对跳转取指令。IFU内还包含一个1KB指令存储器用来存储指令。

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | 指示当前是否为BEQ指令。  0：当前指令非BEQ指令  1：当前指令为BEQ指令 |
|  |  | 指示ALU的计算结果是否为0。  0：计算结果非0  1：计算结果为0 |
|  |  | 指示RS寄存器内容是否大于0，用于BGTZ指令的跳转判断  0：寄存器内容不大于0  1：寄存器内容大于0 |
|  |  | 指示当前指令是否为J指令。  0：当前指令非J指令  1：当前指令为J指令 |
|  |  | 指示当前是否为JAL指令。  0：当前指令非JAL指令  1：当前指令是JAL指令 |
|  |  | 指示当前是否为JRL指令。  0：当前指令非JR指令  1：当前指令是JR指令 |
|  |  | 时钟信号 |
|  |  | 复位信号。  0：无效  1：复位 |
|  |  | 传输rs寄存器内容，用于jr跳转 |
|  |  | MIPS指令中的低16位立即数 |
|  |  | 32位MIPS指令 |
|  |  | 下一条应执行的指令，   1. 用于判断lw、sw地址是否正确。   2.用于实现JAL所规定的指令存储 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 复位 | 当复位信号有效时，PC被复位 |
| 2 | 取指令 | 根据PC从IM中取出指令 |
| 3 | 计算下一条指令地址 | 1. 若nPC\_sel为1，且zero为1或condition为1，则   PC←PC+4+(sign-extend(Imm[15:0])<<2)   1. 若J为1或JAL为1或，则   PC←PCGPRLEN-1..28 || instr\_index || 02   1. 若JR为1，则   PC←GPR[rs]GPRLEN-1..1 || 0   1. 若为其他情况，则   PC←PC+4 |

## GetCode模块定义

### 基本描述

GetCode模块用于将IFU取出的32位MIPS分割为多个信号方便后续器件使用，本身无特殊功能，为过渡器件。

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | 32位MIPS指令 |
|  |  | I指令中的低16位字段， |
|  |  | 功能码，与操作数码相结合，确定指令功能 |
|  |  | 提供移位数目，非移位指令时总是0 |
|  |  | 提供接收结果的目的寄存器的编号 |
|  |  | 提供第一源操作数所使用寄存器的编码 |
|  |  | 提供第二源操作数所使用寄存器的编码 |
|  |  | 操作码 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 取字段 | 将32位MIPS指令分解为7个字段 |

## Controller模块定义

### 基本描述

Controller为系统的控制模块，通过分析MIPS指令的操作码与功能码，辨别该指令属于哪一条具体的指令，随后根据指令功能产生控制信号用于控制其他模块。

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | 操作码 |
|  |  | 功能码，与操作数码相结合，确定指令功能 |
|  |  | 写寄存器的目标寄存器号来源。  0：来源于rt  1：来源于rd |
|  |  | 第二个ALU操作数来源。  0：来源于寄存器  1：来源于立即数拓展 |
|  |  | 写入寄存器的数据来源。  0：来源于ALU计算结果  1：来源于数据存储器 |
|  |  | 寄存器写使能。  0：不能写入  1：能够写入 |
|  |  | 数据存储器写使能。  0：不能写入  1：能够写入 |
|  |  | 标志是否为BEQ指令  0：非BEQ指令  1：是BEQ指令 |
|  |  | 控制立即数拓展方式。  00：0拓展  01：符号拓展  10：低16位补零 |
|  |  | 控制ALU运算种类。  00：加法运算  01：减法运算  10：按位或运算 |
|  |  | 标志是否为J指令。  0：非J指令  1：是J指令 |
|  |  | 标志是否为JAL指令。  0：非JAL指令  1：是JAL指令 |
|  |  | 标志是否为JR指令。  0：非JR指令  1：是JR指令 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 译码操作 | 将MIPS指令译码为对应操作。 |
| 2 | 产生控制信号 | 针对不同指令产生相对应的控制信号 |

## RegFile模块定义

### 基本描述

RegFile模块存放32个32位具有写使能的寄存器，使用5位译码器可以在使能端有效时向指定寄存器写入数据，另外具有两个多路选择器支持同时取出两个寄存器内的内容。当指令为JAL时，将当前指令顺序执行的下一条指令PC地址写入32号寄存器，当指令为ADDI时，若运算溢出则30号寄存器写入1.

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | 时钟信号 |
|  |  | 重置所有寄存器的内容 |
|  |  | 寄存器写使能。  0：不能写入  1：能够写入 |
|  |  | 指示写入rt寄存器还是rd寄存器  0：写入rt寄存器  1：写入rd寄存器 |
|  |  | 指示写入的数据来自ALU还是DM  0：数据来自ALU  1：数据来自DM |
|  |  | 指示立即数运算是否溢出  0：未溢出  1：溢出 |
|  |  | 指示lw、sw地址是否正确（后两位是否为00）  0：地址错误  1：地址正确 |
|  |  | 指示写入的寄存器编号 |
|  |  | 指示第一个取出数据的寄存器编号 |
|  |  | 指示第二个取出数据的寄存器编号  或指示写入的寄存器编号 |
|  |  | 若为JAL指令，pc\_new为PC+4对应地址，用于写入32号寄存器 |
|  |  | ALU运算结果 |
|  |  | DM存储器内容 |
|  |  | 传输从rs对应寄存器取出的数据 |
|  |  | 传输从rt对应寄存器取出的数据 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 写入数据 | 写使能有效时向指定寄存器中写入数据  1．写入ALU运算结果  2．写入DM指定存储器内容  3. 写入PC+4地址于32号寄存器  4. 写入溢出标志于30号寄存器 |
| 2 | 读取数据 | 同时从最多两个指定寄存器中取出数据 |

## Extender模块定义

### 基本描述

不同的指令要求不同形式的立即数拓展，Extender设计了三种立即数拓展方式，支持本题要求的所有指令的拓展需求。

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | 16位待拓展立即数 |
|  |  | 控制立即数拓展方式。  00：0拓展  01：符号拓展  10：低16位补零 |
|  |  | 输出拓展后的数据 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 0拓展 | 将16位立即数0拓展至32位 |
| 2 | 符号拓展 | 将16位立即数符号拓展至32位 |
| 3 | 低16位补零 | 将16位立即数置高16位同时低16位补0 |

## ALU模块定义

### 基本描述

ALU作为运算器，支持无符号数加法、与立即数的有符号支持溢出加法、与立即数的无符号加法、无符号数减法、与立即数按位取或五种运算。ALU还能够为BEQ判断结果是否大于0、为BGTZ判断RS寄存器内容是否大于0、同时输出结果为0信号、溢出信号、RS大于0信号.

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | rs对应寄存器内容 |
|  |  | rt对应寄存器内容 |
|  |  | 拓展后的立即数 |
|  |  | 控制ALU运算种类。  00：加法运算  01：减法运算  10：按位或运算 |
|  |  | 指示运算第二操作数来源  0：来源于rt寄存器内容  1：来源于拓展后立即数内容 |
|  |  | 用于判断具体指令，以完成对应运算 |
|  |  | 用于判断具体指令，以完成对应运算 |
|  |  | 运算结果，采用双符号位以判断溢出 |
|  |  | 指示运算结果是否为0。  0：结果非0  1：结果为0 |
|  |  | 指示运算是否溢出  0：不溢出  1：溢出 |
|  |  | 指示rs寄存器内容是否大于0  0：不大于0  1：大于0 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 无符号加法 | 完成ADDU关于两个寄存器的无符号加法 |
| 2 | 无符号减法 | 完成SUBU关于两个寄存器的无符号减法 |
| 3 | 按位或 | 完成ORI关于寄存器和0拓展后立即数按位或 |
| 4 | 有符号加法 | 完成ADDI关于寄存器和符号拓展立即数的加法，支持溢出 |
| 5 | 无符号立即数加法 | 完成ADDIU关于寄存器和符号拓展立即数的无符号加法。 |
| 6 | 指示结果0 | zero指示结果是否为0，可以与nPC\_Sel一同确定是否为BEQ指令 |
| 7 | 指示溢出 | overflow指示ADDI运算结果溢出 |
| 8 | 指示rs寄存器内容大于0 | condition指示rs寄存器内容大于0，用于BGTZ指令的跳转判断 |

## DM模块定义

### 基本描述

此部分存储器容量为1KB，用RAM实现,是与数据存储器的交互部分，能够实现存储器内数据向寄存器的传递，当使能有效时，支持数据向存储器的写入。

### 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
|  |  | 确定写入的存储器地址 |
|  |  | rt寄存器对应内容 |
|  |  | 用于为lw、sw判断操作地址是否合法 |
|  |  | 数据存储器写使能。  0：不能写入  1：能够写入 |
|  |  | 时钟信号 |
|  |  | 复位信号  0：无效  1：复位 |
|  |  | 输出指定存储器的数据 |
|  |  | 指示lw、sw操作地址错误  0：地址正确  1：地址错误 |

### 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| **1** | 写入 | 当MemWrite为1时以小端序顺序向指定存储器地址处写入数据 |
| **2** | 读取 | 从指定存储器地址处读取数据 |

# 指令测试

## 指令说明

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **操作码** | **助记符** | **指令功能** |
| **1** |  |  | 两个寄存器的无符号数加 |
| **2** |  |  | 两个寄存器的无符号减 |
| **3** |  |  | 判断rs寄存器是否小于rt寄存器并将结果写入rd寄存器 |
| **4** |  |  | 跳转至rs寄存器所存储的地址 |
| **5** |  |  | 从存储器中加载字 |
| **6** |  |  | 向存储器中存储字 |
| **7** |  |  | 若运算结果为0则相对跳转 |
| **8** |  |  | 低16位立即数左移至高位 |
| **9** |  |  | 绝对跳转 |
| **10** |  |  | 寄存器内容与立即数按位或 |
| **11** |  |  | 寄存器内容与立即数有符号加 |
| **12** |  |  | 寄存器内容与立即数无符号加 |
| **13** |  |  | 将pc+4地址写入32号寄存器  后绝对跳转 |
| **14** |  |  | 若rs寄存器内容大于0则相对跳转 |

## 测试代码

|  |  |  |
| --- | --- | --- |
| **编号** | **代码** | **注释** |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  | ) |
| **5** |  |  |
| **6** |  |  |
| **7** |  |  |
| **8** |  |  |
| **9** |  |  |
| **10** |  |  |
| **11** |  |  |
| **12** |  |  |
| **13** |  |  |
| **14** |  |  |
| **15** |  |  |
| **16** |  |  |
| **17** |  |  |
| **18** |  |  |
| **19** |  |  |
| **20** |  |  |
| **21** |  |  |
| **22** |  |  |
| **23** |  |  |
| **24** |  |  |
| **25** |  |  |
| **26** |  |  |
| **27** |  |  |
| **28** |  |  |
| **29** |  |  |
| **30** |  |  |
| **31** |  |  |
| **32** |  |  |
| **33** |  |  |
| **34** |  |  |
| **35** |  |  |
| **36** |  |  |
| **37** |  |  |
| **38** |  |  |
| **39** |  |  |
| **40** |  |  |
| **41** |  |  |
| **42** |  |  |
| **43** |  |  |
| **44** |  |  |
| **45** |  |  |
| **46** |  |  |
| **47** |  |  |
| **48** |  |  |
| **49** |  |  |
| **50** |  |  |
| **51** |  |  |

## 测试结果

完整波形截图如图3.3.1所示，其中第一行为clk信号，第三行为overflow信号，共执行69行指令。最终寄存器结果如图3.3.2所示，存储器结果如图3.3.3所示，mars寄存器、存储器结果如图3.3.4、3.3.5所示. 日程表

描述已自动生成

图3.3.1 完整波形截图

**图形用户界面, 文本, 应用程序, Word

描述已自动生成**

图3.3.2 Modelsim寄存器结果截图

**背景图案

描述已自动生成**

图3.3.3Modelsim存储器结果

表格

描述已自动生成

图3.3.4 MARS寄存器结果

图形用户界面

描述已自动生成

图3.3.5 MARS存储器结果

# 拓展指令设计

## 设计思路

拓展指令BGTZ功能为：若rs寄存器内容大于0则以相对跳转方式进行跳转，如果不大于0则无效果。

对于此功能的设计，首先要清楚如何判断数据是否大于0。显然直接令input>0的方式是无法实现的，我采用的方法是MIPS手册上提供的判断方式：若符号位为0且数值位大于0则说明数据大于0，此时置符号位condition为1，传入IFU模块。在IFU模块中，BGTZ的相对跳转方式和先前已经设计的BEQ指令相同，因此直接在判断BEQ指令的地方或上condition信号，使用BEQ跳转指令下的指令即可实现跳转。

## 指令设计

思路如下：初始化1号寄存器为1，此时1号寄存器大于0，跳转至CON1，将2号寄存器赋值为0，此步若BGTZ不正常工作2号寄存器不会发生变化。随后3号寄存器赋值为-1，此时小于0，程序结束，此步若BGTZ不正常工作4号寄存器会写入1。

## 测试结果

仿真后寄存器内容如图3.4.3.1所示，MARS运行结果如图3.4.3.2所示，二者完全符合，说明BGTZ设计无错误。

文本

中度可信度描述已自动生成

3.4.3.1仿真结果

表格

中度可信度描述已自动生成

3.4.3.2 MARS运行结果

# 总结

得益于先前大作业的基础和魏老师详细的课上讲解，本次期末作业对于单周期cpu的设计整体比较顺利。这次的实践工作让我对cpu有了更进一步的认识，感受到了计算机前辈的精妙设计，也为后续的继续深入探究打下了良好的基础。

实践作业总是麻烦与欣喜共存的。每次实践作业总会在很多意想不到的地方给我带来困扰，比如此次对于vscode的配置、对于modelsim的功能调试等，但每次解决后的成就感便是我长久以来学习的动力。本次设计令我感触最深的是在设计过程中应当给予设计手册十足的重视，所有的设计都应当在完全理解手册要求后进行，不可想当然地自行设计。这不仅是我个人遇到的问题，也是我在为他人解答问题时突出受到感触的地方。另外，我也愈发认为独立解决问题的能力是一个人的核心能力，遇到问题如何通过一步步地调试、通过准确查找相关资料是在学习过程中无论如何也避不开的问题，也是我们需要十分重视的问题。

最后，感谢在设计过程中帮助到我的同学，感谢魏老师辛勤的教学工作！