山东大学计算机科学与技术学院

计算机组成与设计课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202300130183 | 姓名： 宋浩宇 | | 班级： 23级人工智能 |
| 实验题目： 创新实验 | | | |
| 实验学时： 2 | | 实验日期： 2024/12/24 | |
| 实验目的：  题目 | | | |
| 硬件环境：  13th Gen Intel(R) Core(TM) i9-13980HX 2.20 GHz  32.0 GB (31.6 GB 可用)  康芯KX-CDS FPGA平台  芯片Cyclong IV E EP4CE6E22C8 | | | |
| 软件环境：  Windows 11 家庭中文版23H2 22631.4317  Intel Quartus II 13.0sp1(64 bit) | | | |
| 实验内容与设计：  1、实验内容  MIPS指令系统的分析:查找资料，针对典型的MIPS指令集，分析其指令格式、指令功能、分类及其寻址方式等，形成研发报告。  2、实验参考资料  参考资料1  参考资料2  参考资料3  参考资料4  参考资料5  参考资料6   1. 实验结果   由于原文使用markdown格式编写，因此在此附上markdown源码和markdown导出的pdf  markdown源码：  ### 一、指令格式分析  \*\*R 型指令 (Register Format)\*\*:  - 结构: op(6位) rs(5位) rt(5位) rd(5位) shamt(5位) funct(6位)  - 示例: `add $rd, $rs, $rt`  - 说明: R 型指令主要用于寄存器间操作,例如算术运算、逻辑运算和移位操作。`op` 字段表示操作码,`rs` 和 `rt` 是源寄存器,`rd` 是目标寄存器,`shamt` 用于移位操作的位移量,`funct` 字段进一步细化操作类型。  \*\*I 型指令 (Immediate Format)\*\*:  - 结构: op(6位) rs(5位) rt(5位) immediate(16位)  - 示例: `addi $rt, $rs, immediate`  - 说明: I 型指令用于处理立即数,常见于条件分支、加载和存储操作。`immediate` 字段可以表示一个常数值或偏移量。  \*\*J 型指令 (Jump Format)\*\*:  - 结构: op(6位) address(26位)  - 示例: `j address`  - 说明: J 型指令用于跳转,`address` 字段提供了一个跳转地址,需要注意的是,这个地址需要经过处理以形成完整的32位地址。  ### 二、指令功能及分类  \*\*1. 空操作指令\*\*  - `nop`: 无操作指令,用于填充指令流水线。  - `ssnop`: 超级标量无操作指令,用于特定处理器架构的流水线控制。  \*\*2. 寄存器/寄存器传送指令\*\*  - `move`: 将一个寄存器的值传送到另一个寄存器。  - `movf`, `movt`, `movn`, `movz`: 条件传送指令,基于条件码或寄存器内容进行传送。  \*\*3. 常数加载指令\*\*  - `dla`, `la`: 加载地址到寄存器,`dla` 用于 64 位地址。  - `dli`, `li`, `lui`: 加载立即数到寄存器,`lui` 加载立即数到寄存器的高16位。  \*\*4. 算术/逻辑运算指令\*\*  - 加法、减法、乘法、除法、位运算等指令,有符号和无符号操作的变体。  - `sll`, `srl`, `sra`: 移位操作,逻辑左移、逻辑右移、算术右移。  \*\*5. 条件设置指令\*\*  - `slt`, `sltiu`, `sltu` 等: 比较两个值并设置目标寄存器为 0 或 1。  \*\*6. 整数乘法/除法/求余指令\*\*  - `div`, `mul`, `rem`: 进行整数的乘法、除法和求余运算。  \*\*7. 加载/存储指令\*\*  - `lb`, `lh`, `lw` 等: 从内存加载数据到寄存器,不同字节宽度。  - `sb`, `sh`, `sw` 等: 将寄存器中的数据存储到内存。  \*\*8. 跳转/子程序调用/分支指令\*\*  - `jr`, `j`, `jal`: 跳转指令,包括无条件跳转和调用子程序。  - `beq`, `bne`: 条件分支,根据两个寄存器是否相等或不等进行跳转。  \*\*9. 其他指令\*\*  - `rfe`: 从异常返回。  - `syscall`, `break`: 系统调用和断点指令。  ### 三、寻址方式  - \*\*寄存器寻址\*\*: 直接从寄存器中获取操作数。  - \*\*立即数寻址\*\*: 使用指令中的立即数作为操作数或偏移量。  - \*\*绝对地址寻址\*\*: 直接跳转到指定的内存地址。  - \*\*基址寻址\*\*: 使用寄存器的值加上立即数作为内存地址,常用于加载和存储操作。  ### 四、其他特点  - \*\*有符号与无符号操作\*\*: 许多指令有有符号和无符号版本,以处理不同类型的整数数据。  - \*\*双精度版本\*\*: 一些指令有 64 位操作的版本,如 `dadd` 等。  - \*\*字节/半字/字寻址\*\*: 加载和存储指令支持不同大小的数据传输。  - \*\*条件传送\*\*: 根据条件码或寄存器的值进行条件传送。  - \*\*逻辑移位和算术移位\*\*: 移位指令有逻辑和算术两种方式,以处理不同的移位需求。  导出的pdf:  MIPS指令集分析-图片-0  MIPS指令集分析-图片-1 | | | |
| 结论分析与体会：  MIPS指令集的设计巧妙地平衡了硬件复杂度与性能需求，体现了计算机体系结构领域的深刻洞察和创新。它通过采用精简指令集、固定长度指令、流水线处理、寄存器-寄存器操作和加载/存储体系结构等策略，实现了高效的计算能力与简化的硬件设计，同时保持了指令操作的可预测性和直观性。这种设计不仅优化了CPU的性能和功耗，还为计算机科学教育提供了直观的学习工具，展现了人类在追求计算效率、简化设计、以及教育普及方面的智慧和创新精神。 | | | |