华中科技大学

2024

系统能力培养　课程实验报告

题　　目：	指令模拟器
专　　业：	计算机科学与技术
班　　级：	CS2103 班
学　　号：	U202115395
姓　　名：	杜启铭
电　　话：	13526877089
邮　　件：	2454235539@qq.com
完成日期：	2024-09-24

计算机科学与技术学院

目 录

1 课程实验概述 1

1.1 课设目的 1
1.2 课设任务 1
1.3 实验环境 1

2 PA1 - 开天辟地的篇章：最简单的计算机 2

2.1 简单调试器 2
2.1.1 单步执行 si 2
2.1.2 打印程序状态 info 2
2.1.3 扫描内存 x 2
2.2 表达式求值 2
2.2.1 表达式解析 2
2.2.2 表达式求值 2
2.3 设置与删除监视点 3
2.4 运行结果 3
2.5 必答题 3

3 设计 5

4 实验结果与结果分析 6

参考文献 7

1　课程实验概述

1.1 课设目的

本次课程设计通过实现一个经过简化但功能完备的 riscv32 模拟器 NEMU，最终在 NEMU 上运行游戏“仙剑奇侠传”，来探究“程序在计算机上运行”的基本原理。

1.2 课设任务

本次课程设计主要包含下列实验内容。

1. 实现简易调试器、表达式求值、监视点与断点等功能。
2. 运行一个 C 程序、丰富指令集并测试所有程序、实现 I/O 指令并测试打字游戏。
3. 实现系统调用、实现文件系统、运行仙剑奇侠传。
4. 实现分页机制、实现进程上下文切换、时钟中断驱动的上下文切换。

1.3 实验环境

使用教师提供的 virtual box 镜像。

2 PA1 - 开天辟地的篇章: 最简单的计算机

2.1 简单调试器

我们需要为nemu实现一些简单的调试功能,包括单步执行、打印程序状态、扫描内存、表达式求值、扫描内存、设置监视点、删除监视点。

2.1.1 单步执行 si

cpu_exec 函数实现了执行指定次数的 CPU 循,因此单步执行的功能通过直接调用该函数即可实现。

2.1.2 打印程序状态 info

程序状态包括两种状态：寄存器状态和监视点状态。

定义一个名为 cmd_info 的函数,用于处理 info 命令。根据传入的参数 args，如果参数是 'r'，则调用 isa_reg_display() 显示寄存器信息；如果参数是 'w'，则调用 display_watchpoints() 显示监视点信息并返回；如果参数无效，则打印 "Wrong argument!" 错误信息。如果没有提供参数，则打印 "Lack argument!" 错误信息。

2.1.3 扫描内存 x

定义一个名为 cmd_x 的函数，用于处理 x 命令。该函数从参数 args 中解析出要读取的内存单元数量 number 和起始地址 index，如果解析失败或参数无效，则打印错误信息 "Wrong argument!" 并返回。否则，它会循环读取指定数量的内存单元，每读取四个单元打印一行，调用 isa_vaddr_read 函数读取内存地址的值，并以十六进制格式打印地址和值。

2.2 表达式求值

2.2.1 表达式解析

定义一个名为 make_token 的函数，用于将输入字符串 e 分解成一系列的标记（tokens）。函数通过正则表达式逐个匹配输入字符串中的子串，并根据匹配的规则将子串记录为相应类型的标记。对于不同类型的标记，函数会执行不同的操作。

2.2.2 表达式求值

定义一个名为 calculate 的递归函数，用于计算表达式的值。函数根据传入的标记数组 tokens 和索引范围 [i, j] 解析并计算表达式的值。如果 i 和 j 相等且标记是数字或寄存器，则返回其值；如果表达式被括号包围，则去掉括号递归计算；否则，找到主操作符并递归计算其左右操作数的值，并根据操作符类型执行相应的运算。函数通过 success 标志指示计算是否成功，并处理各种运算符和错误情况，如除零错误和无效表达式。

2.3 设置与删除监视点

我们需要实现：监视点的创建、删除、打印、检查。

监视点的创建。定义一个名为 new_wp 的函数，用于分配一个新的监视点（WP）。如果没有可用的空闲监视点，则打印错误信息并触发断言失败。否则，从空闲链表中取出一个监视点，将其插入到活动监视点链表的头部，并返回该监视点的指针。

监视点的删除。定义一个名为 free_wp 的函数，用于根据监视点编号 NO 从活动监视点链表中删除相应的监视点；如果找到该监视点，则将其从链表中移除并添加到空闲监视点链表中，以便后续复用

监视点的打印。定义一个名为 display_watchpoints 的函数，用于显示当前所有活动的监视点。函数首先打印表头，然后遍历活动监视点链表 head，对于每个监视点，打印其编号 (NO)、表达式 (wp_expr) 和上次计算的结果 (last_value)。

监视点的检查。定义一个名为 check_watchpoints 的函数，用于检查所有活动的监视点是否发生变化。函数遍历监视点链表 head，计算每个监视点表达式的当前值 res，如果计算失败则打印错误信息并触发断言失败；如果当前值与上次记录的值 last_value）不同，则打印监视点信息和新旧值，并更新 last_value。函数返回一个布尔值 result，指示是否有监视点发生了变化。

2.4 运行结果

简单调试器的功能已基本实现。

2.5 必答题

这里我们来对实验文档中的必答题进行逐一的解答。

1.我选择的 ISA 是__riscv32__。

2.用于调试的时间为 10 小时，实现简单调试器后的调试时间将降低为 2 小时，由此可见，通过实现一定的基础设施，可以有效的减少我们在后续工作中 debug 的工作量。

3. riscv32 的指令格式有：R 型（Register）、I 型（Immediate）、S 型（Store）、B 型（Branch）、U 型（Upper Immediate）、J 型（Jump）

LUI（Load Upper Immediate）指令将一个 20 位的立即数加载到寄存器的高 20 位，低 12 位填 0。

mstatus 包含字段：MIE 机器模式全局中断使能、MPI 机器模式中断使能前值、MPP 机器模式前模式、FS 浮点状态、XS 扩展状态、SD 状态脏位。

4.

（1）nemu/目录下的所有.c 和.h 文件总共有多少行代码?

使用以下命令统计代码行数：

find nemu/ -name '*.c' -o -name '*.h' | xargs wc -l

（2）和框架代码相比, 你在 PA1 中编写了多少行代码?

两次使用（1）中的命令并计算结果。

（3）将统计代码行数的命令写入 Makefile 中。

在 Makefile 添加以下内容：

count:

```
find nemu/ -name '*.c' -o -name '*.h' | xargs wc -l
```

运行 make count 即可统计代码行数。

（4）除去空行之外, nemu/目录下的所有.c 和.h 文件总共有多少行代码?

使用以下命令统计除去空行的代码行数：

```
find nemu/ -name '*.c' -o -name '*.h' | xargs grep -v '^\s*$$' | wc -l
```

5. -Wall 和-Werror 是 GCC 编译器中的两个常用选项：

-Wall：

（1）启用所有常见的警告选项，帮助开发者发现潜在的代码问题。

（2）包括未使用的变量、未初始化的变量、隐式函数声明等警告。

-Werror：

（1）将所有警告视为错误，编译器在遇到警告时会停止编译。

（2）强制开发者修复所有警告，确保代码质量。

为什么要使用-Wall 和-Werror：

（1）提高代码质量：通过启用警告，开发者可以发现并修复潜在的代码问题。

（2）强制修复警告：将警告视为错误，确保所有警告都被修复，避免潜在的运行时错误。

（3）保持代码整洁：减少代码中的潜在问题和不良实践，保持代码库的整洁和可维护性。

3　设计

4　实验结果与结果分析

参考文献