

华中科技大学

2017

系统能力综合训练 课程设计报告

题 目: X86 模拟器设计

专 业: 计算机科学与技术

班 级: CS1506

学 号: U201514639

姓 名: 刘科翰

电 话: 13018052550

邮 件: 524792973@qq.com

完成日期: 2018-12-19 周三上午



计算机科学与技术学院

华中科技大学课程设计报告

目 录

1	课程设计概述.....	2
1.1	课设背景	2
1.2	设计任务	2
2	PA1 简易调试器.....	3
2.1	功能实现的要点.....	3
2.2	必答题	3
2.3	主要故障与调试.....	5
3	PA2 冯诺依曼计算机系统.....	6

1 课程设计概述

1.1 课设背景

1.2 设计任务

2 PA1 简易调试器

2.1 功能实现的要点

task PA1.1: 实现单步执行, 打印寄存器状态, 扫描内存等基本指令。

task PA1.2: 实现简单的算术表达式求值。

task PA1.3: 实现扩展的表达式求值和监视点功能, 完善指令系统。

2.2 必答题

1.理解基础设施。我们通过一些简单的计算来体会简易调试器的作用。首先作以下假设: 假设你需要编译 500 次 NEMU 才能完成 PA。假设这 500 次编译当中, 有 90% 的次数是用于调试。假设你没有实现简易调试器, 只能通过 GDB 对运行在 NEMU 上的客户程序进行调试。在每一次调试中, 由于 GDB 不能直接观测客户程序, 你需要花费 30 秒的时间来从 GDB 中获取并分析一个信息。假设你需要获取并分析 20 个信息才能排除一个 bug。那么这个学期下来, 你将会在调试上花费多少时间? 由于简易调试器可以直接观测客户程序, 假设通过简易调试器只需要花费 10 秒的时间 从中获取并分析相同的信息。那么这个学期下来, 简易调试器可以帮助你节省多少调试的时间?

答: (1) 没有简易调试器下的情况:

一共要进行 450 次调试, 600s 排除一个 bug, 假设一次调试解决一个 bug, 那么在完成 nemu 的过程中, 需要花费 4500 分钟来调试, 也即 75 个小时用于调试。

(2) 有简易调试器下的情况:

解决一个 bug 需要 200s, 则完成 nemu 过程中需要 25 个小时用于调试。

2.查阅 i386 手册。

(1) EFLAGS 寄存器中的 CF 位是什么意思?

(2) ModR/M 字节是什么?

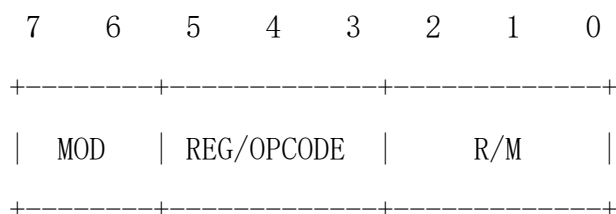
(3) mov 指令的具体格式是怎么样的?

答: (1) 阅读第二章 Basic Programming Model 中的 2.3 章节 Register 其中的 2.3.4 小节 Flag Register, 在 2.3.4.1 中有一个给 Appendix C 的跳转链接, 里面详细介绍了每一个标志物的定义。CF, 全称 Carry Flag, 是一个 Status Flags——状态标志位, 状态标

华中科技大学课程设计报告

志位使得当前指令的执行能够受到之前指令的结果的影响。CF 是标志高位进位或是借位信息的一个标志位。

(2) i386 手册 17.2.1 ModR/M and SIB Bytes 小节中介绍了 ModR/M and SIB 字节的含义和作用, 其中 ModR/M 字节一般紧跟着指令操作码, 这个字节一般包含着三个方面的信息, 一共分为三个区域: 分别为 mod 区、reg 区, r/m 区。mod 区占 3 个 bits, 与 r/m 区的信息结合起来可以指示 8 个寄存器和 24 个索引模式, 一个 32 种可能的情况。reg 区占 3 个 bits, 可以用来表示一个寄存器也可以作为操作码 opcode 的补充, 而这个由这条指令的第一个字节 opcode 决定。而 r/m 区占最后的 3 个 bits, 可以指向一个作为操作数的寄存器, 或者可以成为寻址模式中的一部分编码与 mod 区一起发挥作用。具体的字节信息如下所示:



(3) i386 手册 17.2.2.11 Instruction Set Details 里面详细地讲解了每一种指令, 具体到 MOV 中, MOV 指令的格式为: MOV DES, SRC, MOV 指令将第二个操作数 SRC 的值拷贝到第一个操作数 DES 处。

3.shell 命令。

- (1) 完成 PA1 的内容之后, nemu/ 目录下的所有 .c 和 .h 文件总共有多少行代码?
- (2) 你是使用什么命令得到这个结果的?
- (3) 和框架代码相比, 你在 PA1 中编写了多少行代码?
- (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?

答: (1) 完成 PA1 后, nemu/目录下总共有 4535 行代码 (包括空行、注释)。

(2) 使用的命令: `find . -name "*.c" | xargs cat | wc -l`

(3) checkout 到 pa0 之后执行 (2) 中的命令, 得到的行数为 3833 行, 故在 pa1 中总共编写了 702 行代码。

(4) 除去空行之后为 3804/3114 行代码。使用的命令为: `find . -name "*.c" | xargs cat | grep -v "^$" | wc -l`。可见编写代码中又多加了几行无用的空行。

4.请解释 gcc 中的 -Wall 和 -Werror 有什么作用？为什么要使用 -Wall 和 -Werror？

答：（1）-Wall 选项是一系列警告编译选项的集合，本次实验中比较常见的警告有[-Wunused]，[-Wunused]是也是一系列选项的集合，用来警告存在一个定义了却未使用的局部变量或者一个函数的参数在函数的实现中并未被用到或者一个显式计算表达式的结果未被使用等等；还有[-Wimplicit]，它也是一个选项的集合，用于警告在声明函数却未指明函数返回类型时给出警告或者是在函数声明前调用该函数时给出警告等等。在-Wall 选项集合中还有很多其他的选项集合，在编译调试时是很有用的工具。

（2）-Werror 是将所有的 Warning 都当成 Error 来处理，视警告为错误，只要有警告存在程序都会终止执行，避免潜藏的 Fault。

2.3 主要故障与调试

2.3.1 寻找主运算符出错

出错原因：忽略了运算符之间的优先级。

解决方案：在编写寻找主运算符的过程中，除了排除括号内的运算符和非运算符，剩下的就需要根据优先级进行选择，具体优先级参考 c 语言各个运算符的优先级。

2.3.2 解引用在简单表达式可以用，复杂则错误

出错原因：未将解引用当成一个最高优先级的运算符来看待，而是把它放在 eval 函数的一个 if 分支中进行分流，实现思想有误。

解决方案：把解引用当成一个具有最高优先级的运算符，在寻找主运算符的过程中将它纳入。在进行计算时，由于它是单目运算符，需要与双目运算符进行区分。

2.3.3 计算表达式偶尔出错

出错原因：没有清空 tokens 数组中每一个元素的缓冲区，以至于还保留着上一次分析的 token 的信息。没有对有效 token 之间的空格 token 进行处理。

解决方案：每次 make_tokens 的时候，注意清空缓冲区。时刻注意空格的处理方式。

3 PA2 冯诺依曼计算机系统

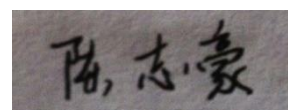
• 指导教师评定意见 •

一、原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字：



二、对课程设计的学术评语（教师填写）

三、对课程设计的评分（教师填写）

评分项目 (分值)	报告撰写 (30 分)	课设过程 (70 分)	最终评定 (100 分)
得分			

指导教师签字：_____