实验报告

PA1-简易调试器

班级 大类八班

学号 3022244290

姓名 陈秋澄

实验进度（任务自查表）

序号	完成情况
必做任务 1	已完成
必做任务 2	已完成
必做任务 3	已完成
必做任务 4	已完成
必做任务 5	已完成
必做任务 6	已完成
必做任务 7	已完成
选做任务 1	已完成
选做任务 2	未完成

思考题（请注明题号，如思考题 1，思考题 2，...）

思考题 1：opcode_table 到底是一个什么类型的数组？

opcode_table 是一个函数指针类型的数组。

```
typedef int (*helper_fun)(swaddr_t);
helper_fun opcode_table [256]
```

在 exec.c 文件中第 96 行找到 opcode_table 数组，发现其为 helper_fun 类型；该文件中第 6 行找到了 helper_fun 的定义，老师上课讲述过，helper_fun 是一个返回类型为 int 型，参数类型为 swaddr_t 类型的一个函数指针。因此，opcode_table 数组是一个函数指针数组，返回值类型为 int 型，参数类型为 swaddr_t 类型。

思考题 2：

一、在 cmd_c()函数中，调用 cpu_exec()的时候传入了参数-1，你知道为什么吗？

函数 cpu_exec()中，

```
void cpu_exec(volatile uint32_t n) {
	if(nemu_state == END) {
		printf("Program execution has ended. To restart the program, exit NEMU and run again.\n");
		return;
	}
```

n 为无符号整数，-1 转变成无符号，即无符号中最大的数字，那么函数里的 for 循环可以执行最大次数的循环，从而让 cpu 处理之后的指令。

二、框架代码中定义 wp_pool 等变量的时候使用了关键字 static，static 在此处的含义是什么？为什么要在此处使用它？

框架代码中定义 wp_pool 等变量时使用了关键字 static，为只能被

本文件中的函数调用静态全局变量且是全局变量，不能被同一程序其他文件中的函数调用。在此处使用 static 是为了避免它被误修改。

思考题 3：

一、查阅 i386 手册

1、EFLAGS 寄存器中的 CF 位是什么意思?

P34 CF 是进位标志，文中提到还可以看附录 c

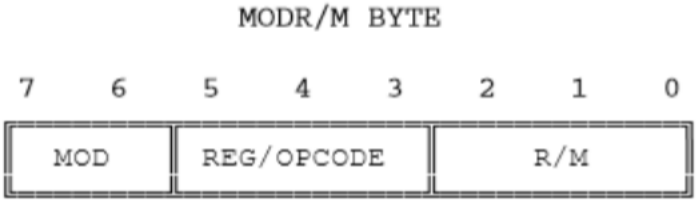
2.3.4.1 Status Flags

The status flags of the EFLAGS register allow the results of one instruction to influence later instructions. The arithmetic instructions use OF, SF, ZF, AF, PF, and CF. The SCAS (Scan String), CMPS (Compare String), and LOOP instructions use ZF to signal that their operations are complete. There are instructions to set, clear, and complement CF before execution of an arithmetic instruction. Refer to Appendix C for definition of each status flag.

P419 中写到 CF: carry flag-set on high-order bit carry or borrow; cleared otherwise 即 CF 位：在最高位发生进位或者借位的时候将其置 1，否则为 0。

2、ModR/M 字节是什么?

P241-243：ModR/M 由 Mod，Reg/Opcode，R/M 三部分组成。Mod 是前两位，提供寄存器寻址和内存寻址，而 Reg/Opcode 为 3-5 位，若 Reg 表示使用哪个寄存器，Opcode 表示对 group 属性的 Opcode 进行补充；R/M 为 6-8 位，与 mod 结合起来由图得 8 个寄存器和 24 个内存寻址。



3、mov 指令的具体格式是怎么样的?

P347，格式为 DEST <-SRC。

二、shell 命令

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

```
  4691 total
find ./nemu/ -name "*.[ch]" | xargs grep "^." | wc -l
3778
trouverecc@ubuntu:~/NEMU2021$
```

nemu 目录下的所有.c 和.h 和文件总共有 4195 行代码。通过 find . -name "*[.h/.c]" | xargs wc -l 命令得到了这个结果。和框架代码相比，我在 PA1 中编写了 496 行代码。在 Makefile 中添加相应的代码，实现 make count 命令。

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?

除去空行之外, nemu 目录下的所有.c 和.h 文件总共有 3778 代码。通过 find . -name "*[.h/.c]" | xargs grep "^." | wc -l 命令得到了这个结果。

三、Make 文件

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

-Wall 使 GCC 尽可能产生多的警告信息，取消编译操作，打印出编译时所有错误或警告信息。

-Werror 要求 GCC 将所有的警告当成错误进行处理，取消编译操作。

使用 -Wall 和 -Werror 可以找出存在的错误，尽可能地避免程序运行出现错误，优化程序。

思考题 4：请简述 NEMU 从启动到用户程序加载，再到运行用户程序的全过程。

NEMU 开始执行时，会进行一些和 monitor 相关的初始化工作，包括打开日志文件，读入 ELF 文件的符号表和字符串表，编译正则表达式，初始化监视点结构池。之后代码会对寄存器结构的实现进行测试（使用 reg_test()），测试通过后会 restart() 函数（在 nemu/src/monitor/monitor.c 中定义），它模拟了"计算机启动"的功能，主要是进行一些和"计算机启动"相关的初始化工作，包括初始化 ramdisk、读入入口代码 entry、设置%eip 的初值、初始化 DRAM 的模拟 init_ddr3()。

而 restart() 函数执行完毕后，NEMU 会进入用户界面主循环 ui_mainloop()(在 nemu/src/monitor/debug/ui.c 中定义)。键入 c 之后，NEMU 开始进入指令执行的主循环 cpu_exec()（在 nemu/src/monitor/cpu-exec.c 中定义）。

NEMU 不断执行指令，直到遇到以下情况之一，才会退出指令执行的循环：第一，达到要求的循环次数；第二，用户程序执行了 nemu_trap 指令。这是一条特殊的指令，机器码为 0xd6。x86-32 中并没有这条指令，它是为了指示程序的结束而加入的。退出

cpu_exec()之后，NEMU 将返回到 ui_mainloop()，等待用户输入命令。

思考题 5：请回答 NEMU 中主存是使用哪种数据类型模拟的？为什么使用该类型？

NEMU 中主存是使用 uint8_t 数组的数据类型模拟的。

原因有三：第一，内存中能够存储的最小单元是字节（8 位），使用 uint8_t 可以精准地读出每一个字节，方便模拟出一个计算机系统。

第二，数组在内存中的布局是连续的，这符合实际计算机内存的物理结构。这种连续的布局使得数据的访问更加高效，可以利用计算机的缓存机制提高访问速度。

第三，数组是一种简单的数据结构，易于实现和理解。它提供了直接的访问和操作内存数据的方式，无需复杂的数据结构或算法，加快了 NEMU 运行时的速度。

实验遇到的问题、思考、解决办法（可以不填写）

问题一:

在实现表达式求值的功能的时候，我自己调试程序的时候发现，在第一次执行指令时 p (1 + 3) * 4 程序能够执行并返回正确的结果16，接着执行 p -55 + 2 指令，程序仍然能够执行并返回正确的结果-53，但是接下来在执行 p (1 + 3) * 4 的指令的时候，程序返回错误的值 72。

思考与解决办法:

开始时我认为是计算负数的功能出现问题，于是检查 expr.c 文件中与计算负数相关的代码，没有看出漏洞。后来，我发现其实是输入两位数以及两位以上的数的问题。找到错误之后我发现我在 make_token 函数中复制字符串的时候使用的是 strncpy 函数，我自己在本地试验后发现，strncpy 函数在复制字符串的时候没有将目标字符串原有的值清空。

这就造成了如果已知串比目的串的长度小的情况时，目的串长出的部分会保留，这就导致了我输入两位及两位以上的数并计算之后，在下一次输入一个一位数计算的时候，那个两位数的第二位仍然被保留，从而造成了计算错误，出现了这次奇怪的 bug，然后通过询问同学之后，使用强大的 sprintf 函数进行字符串复制时可以首先清空目的串的内容，问题得以解决。

问题二:

我在做 PA1-3 时，始终编译不通过，出现这样的提示 error: identifier "" is undefined。

思考与解决办法：

我在 CSDN 上搜索错误提示相关内容的解决办法，得知头文件相互引用可能会造成错误，于是我在 nemu/src/monitor/cpu-exec.c 里添加头文件#include "monitor/watchpoint.h"，错误得以解决。

实验心得（可以不填写）

第一，我要加强查找文献、整理文件、阅读文献的能力，尤其是加强阅读英文文献的能力。

第二，学会自己解决问题，可以通过网络搜索解决大部分的问题。

第三，遇到不会的问题，可以请教老师、与同学交流，增进对知识的理解。

第四，学会使用 gcc 调试器，利用打断点来自行测试程序从中发现错误的代码，减少因为找不到问题在哪而浪费的时间。

第五，面对大量代码时，首先应该梳理框架，再抓细节。