实验报告

PA2-指令系统

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
选做任务 1	未完成
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：main 函数返回到哪里？ **查看 testcase 的相关代码,你知道用户程序从 main 函数返回之后会跳转到哪里吗？如果用户程序在 GNU/Linux 中运行，问题的答案又是什么？** 对于 NEMU，main 函数返回至 start.S（位于 testcase/src/start.S）中“HIT_GOOD_TRAP”。 对于 GNU/Linux, main 函数返回至一个称为“_libc_start_main”的函数，位于 glibc 库中。 **思考题 2：比较 FLOAT 和 float** **FLOAT 和 float 类型的数据都是 32 位，它们都可以表示 2^32 个不同的数，但由于表示方法不一样, FLOAT 和 float 能表示的数集是不一样的。思考一下，我们用 FLOAT 来模拟表示 float，这其中隐含着哪些取舍?** 使用 FLOAT 表示 float 后，其表数范围和表数精度都会变小。 **思考题 3：消失的符号** **我们在 add.c 中定义了宏 NR_DATA，同时也在 add()函数中定义了局部变量 c 和形参 a, b，但你会发现在符号表中找不到和它们对应的表项，为什么会这样？思考一下，什么才算是一个符号(symbol)?** 宏和局部变量都不是符号。其中宏在编译的时候就进行了替换，因此不会保存在 elf 文件中；而局部变量保存在栈中，栈是运行时动

态变化的，故局部变量也不会出现在 elf 文件中。

一个符号是指函数、全局变量或静态变量。

思考题 4：堆和栈在哪里?

我们提到了代码和数据都在可执行文件里面，但却没有提到堆(heap)和栈(stack)。为什么堆和栈的内容没有放入可执行文件里面？那程序运行时刻用到的堆和栈又是怎么来的?

堆和栈不是静态的，是动态的，它们是在程序运行过程中动态变化的。其中，栈是由编译系统自动进行分配的，主要用于在过程调用中保存局部变量、传递的参数、现场和返回地址，每进行一次函数调用就会生成一个栈帧，函数返回后该函数的栈帧就被删除。堆是需要程序员资金申请和释放的，申请时需要指明大小。

例如可通过 malloc, realloc, calloc 函数在堆中申请空间, 通过 free 函数释放已申请的堆中空间。

思考题 5：如何识别不同格式的可执行文件?

如果你在 GNU/Linux 下执行一个从 Windows 拷过来的可执行文件，将会报告"格式错误"。思考一下, GNU/Linux 是如何知道"格式错误"的?

GNU/Linux 解析文件头以判断 magic number，如果是“7f 45 4c”说明该文件是 ELF 文件；否则报告“格式错误”。

思考题 6：冗余的属性?

使用 readelf 查看一个 ELF 文件的信息，你会看到一个 segment 包含两个大小的属性，分别是 FileSiz 和 MemSiz, 这是为什么?

再仔细观察一下，你会发现 FileSiz 通常不会大于相应的 MemSiz，这又是为什么?

FileSiz 属性指该段在 ELF 文件中占据的字节数，而 MemSiz 属性指该段加载到主存后所占据的字节数。

FileSiz 通常不大于相应的 MemSiz 的原因是对于.bss 段它不占据 ELF 文件的空间，但加载到内存后需要给该段分配空间。

实验遇到的问题、思考、解决办法（可以不填写）

问题一：

实验过程中经常出现类似下图的问题。

```
(nemu) c
invalid opcode(eip = 0x001001c4): d3 e8 85 d2 78 0e 5b 5d ...

There are two cases which will trigger this unexpected exception:
1. The instruction at eip = 0x001001c4 is not implemented.
2. Something is implemented incorrectly.
Find this eip value(0x001001c4) in the disassembling result to distinguish which case it is.

If it is the first case, see

for more details.

If it is the second case, remember:
* The machine is always right!
* Every line of untested code is always wrong!

nemu: nemu/src/cpu/exec/special/special.c:24: inv: Assertion `0' failed.
Makefile:64: recipe for target 'run' failed
make: *** [run] Aborted (core dumped)
```

思考与解决办法：

我在 txt 文件中对照相关地址，找到出现问题的指令，修改 opcode 等，逐步调试。

```
1001b0: c1 eb 17             shr    $0x17,%ebx
1001b3: 25 ff ff 7f 00       and    $0x7fffff,%eax
1001b8: 0f b6 db             movzbl %bl,%ebx
1001bb: 0d 00 00 80 00       or     $0x800000,%eax
1001c0: 29 d9                sub    %ebx,%ecx
1001c2: 78 0c                js     1001d0 <f2F+0x30>
1001c4: d3 e8                shr    %cl,%eax
1001c6: 85 d2                test   %edx,%edx
1001c8: 78 0e                js     1001d8 <f2F+0x38>
1001ca: 5b                   pop    %ebx
1001cb: 5d                   pop    %ebp
1001cc: c3                   ret
```

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，面对大量代码时，首先应该梳理框架，再抓细节。