PA2 前半阶段实验报告

14307130078 张博洋

一、实验进度

完成了 PA2 所有内容，包括后半阶段的内容。

二、部分选答题

1. 立即数背后的故事

当 NEMU 模拟的机器和运行 NEMU 本身的机器的字节序不同时，在模拟访问内存时必须注意字节序的问题。NEMU 在读写模拟出的内存时，由于模拟出的内存中内容是以相异的字节序存储的，直接读写会导致字节序错乱。因此在读写内存时需要专门函数转换读取到的数值，才能保证模拟出的内存与被模拟机器的字节序一致。

2. 不能返回的 main 函数

main 函数不能返回的根本原因是在调用 main 函数的指令之后没有其他指令了。浏览代码可以发现，在 start.S 文件中包含了程序初始化相关的代码。该初始化代码在设置好栈指针后，便使用 call 指令调用了 main 函数。然而该 call 指令之后再无其他指令。当 main 函数返回后，会从 call 指令之后的指令处继续执行。事实上，一般来说这条 call 指令之后会紧接着 C 语言代码中的第一个函数。因此，在 main 函数返回后，程序会以混乱的状态继续执行，最终以访问越界或非法指令终止执行。

3. 消失的符号

局部变量不能算作符号，因为它们会在栈上分配空间，地址并不固定，不能也无需为它们分配符号。只有全局变量、静态变量以及函数才能作为符号。

4. 寻找"Hello World!"

该字符串字面值不是符号，而是程序当中的数据。字符串表是保存符号名字字符串的地方，因此它不存在于字符串表中。事实上它应该存在于.rodata 节中。

5. 丢失的信息

符号表中只存有符号的名字和地址，还有符号的大小、对齐等信息，唯独没有符号的类型信息。因此 NEMU 并不能知道这个符号是的数值是什么类型，因此只能假定它是 uint32_t 而进行输出。

6. 冗余的符号表

可重定位目标文件之所以可重定位是因为有重定位表，一旦删去了符号表，重定位信息就丢失了，自然就无法链接了。可执行文件可执行是因为有程序头表，它存储的是程序如何装入内存的数据，删除符号表并不会影响它，因此删去可执行文件的符号表一般没有什么影响。

7. %ebp 是必须的吗？

当没有 EBP 后，程序必须通过 ESP 来访问局部变量。编译器会维护函数使用栈的情况，并准确地计算出局部变量相对于 ESP 的偏移。我认为调试器在缺少 EBP 的情况下，打印栈帧链是通过编译器附加的调试信息做到的。打印栈帧链的核心是找到函数的返回地址，编译器只要将使用栈的信息或者返回地址的位置保存好，调试器就可以找到返回地址，进而分析出整个栈帧链。

三、实验心得

1. 边实现边测试，重要的地方一定不能偷懒

边实现边测试可以提高自己实现的功能的正确性，从而减少之后遇到莫名其妙问题的可能性。重要的函数一定要仔细测试，不能偷懒。例如计算 EFLAGS 是通过一

个模拟 ALU 的函数来进行的。由于 EFLAGS 寄存器十分重要，各种判断、跳转都依赖于它，因此我认真仔细地测试了我的实现。由于一开始我的测试数据大部分依赖于随机生成，某些 FLAG 并不能被很好地测试，因此我曾经一度以为我的实现是正确的，直到后来才发现问题。之后我加大了测试数据的测试面，采用随机与手工编写相结合的方式，尽量将更多的 FLAG 和情况容纳进来，确保函数的实现万无一失。

2．测试驱动开发

利用测试驱动开发的编程模式可以迅速地完成要求的任务。在 NEMU 指令的实现中，我往往是遇到一个不能执行的测试用例，就编写相应缺少的指令，或者除掉指令实现中的错误。好的测试用例可以帮助测试实现的正确性，添加自己的测试用例也很有必要，测试用例总是多多益善。我利用我的测试用例发现了 NEMU 中的一个小问题，并且改正了它。我还把数据结构课的作业转化为了测试用例。在测试的帮助下，很多细小的问题都无处遁形，提高了编程效率。