实验报告-PA1

甘晨 181240014

2019 年 9 月 29 日

1 实验进程

1.1 PA1.1

ISA 选择了 x86，利用 Union 和 Struct 重新组织了寄存器的结构。实现了单步执行、打印寄存器和扫描内存功能，艰难地尝试着理解框架代码。

1.2 PA1.2

实现了算数表达式的词法分析、匹配括号、寻找主操作浮符和递归求值功能，但没有是实现负数的运算。最后，实现了生成表达式工具，并借助此工具测试了表达式求值的功能。

1.3 PA1.3

实现了拓展表达式求值的功能，能够实现指针解引用，取寄存器值和“&&”、“==”、“!=”运算功能，其余一些运算符在后面有需要再实现。

2 必答题

2.1 PA1.1

实现寄存器结构调整:

首先，通过 RTFSC 和阅读讲义中的 x86 寄存器组织结构，可以发现 32 位、16 位和 8 位寄存器需要共用地址空间，在提示的帮助下，这里可以使用 union 结构来重新组织这些寄存器，而使用匿名 union 的好处是方便直接访问联合类型的成员。然而，运行 make run 还会 abort，报错提示为（似乎是，记不太清了）:

Assertion ”sample[R_EAX] == cpu.eax” failed”

通过 RTFSC，我发现在程序运行过程中没有给 cpu.eax 赋值的过程，所以说这个 assert（）会被触发。那么如何给这些寄存器赋值呢？这里卡了我很久，以至于想换成 riscv32，在做了各种尝试，甚至尝试在结构体定义的过程中直接赋值的操作. 后来在，在大佬的提示下，了解到了可以再用一个联合类型，让上面的 _32、_16 和 _8 寄存器和下面的 eax、edx 等寄存器共用空间，这样就可以实现对寄存器的赋值了，但是这边还需要注意的时，需要把 eax，edx 等通用寄存器整合为一个匿名 struct 结构，防止这些通用寄存器共用空间。

实现单步执行，打印寄存器，扫描内存:

```
nector@debian:~/ics2019/nemu$ make ISA=x86 run
Building x86-nemu
+ CC src/monitor/debug/watchpoint.c
+ LD build/x86-nemu
make -C /home/nector/ics2019/nemu/tools/qemu-diff
make[1]: Entering directory '/home/nector/ics2019/nemu/tools/qemu-diff'
make[1]: Nothing to be done for 'app'.
make[1]: Leaving directory '/home/nector/ics2019/nemu/tools/qemu-diff'
./build/x86-nemu -l ./build/nemu-log.txt -d /home/nector/ics2019/nemu/tools/qemu-diff/build/x86-qemu-so
[src/monitor/monitor.c,36,load_img] No image is given. Use the default build-in image.
[src/memory/memory.c,16,register_pmem] Add 'pmem' at [0x00000000, 0x07ffffff]
[src/device/io/mmio.c,14,add_mmio_map] Add mmio map 'argsrom' at [0xa2000000, 0xa2000fff]
[src/monitor/monitor.c,20,welcome] Debug: ON
[src/monitor/monitor.c,23,welcome] If debug mode is on, A log file will be generated to record every instruction
 NEMU executes. This may lead to a large log file. If it is not necessary, you can turn it off in include/common
.h.
[src/monitor/monitor.c,28,welcome] Build time: 21:09:23, Sep 28 2019
Welcome to x86-NEMU!
For help, type "help"
(nemu) si
  100000:   b8 34 12 00 00                      movl $0x1234,%eax
(nemu) si 5
  100005:   b9 27 00 10 00                      movl $0x100027,%ecx
  10000a:   89 01                               movl %eax,(%ecx)
  10000c:   66 c7 41 04 01 00                   movw $0x1,0x4(%ecx)
  100012:   bb 02 00 00 00                      movl $0x2,%ebx
  100017:   66 c7 84 99 00 e0 ff ff 01 00       movw $0x1,-0x2000(%ecx,%ebx,4)
(nemu) si 5
  100021:   b8 00 00 00 00                      movl $0x0,%eax
  100026:   d6                                  nemu trap
nemu: HIT GOOD TRAP at pc = 0x00100026

[src/monitor/cpu-exec.c,28,monitor_statistic] total guest instructions = 8
(nemu)
```

图 1: 单步执行

```
(nemu) info r
Register_id    Hexadecimal       Decimal
    eax:       0x00000000      000000000000D
    ecx:       0x00100027      000001048615D
    edx:       0x2db590b7      000766873783D
    ebx:       0x00000002      000000000002D
    esp:       0x7b984bd6      002073578454D
    ebp:       0x6d8c6756      001837918038D
    esi:       0x200cfd4e      000537722190D
    edi:       0x43ba55ca      001136285130D
(nemu)
```

图 2: 打印寄存器

```
(nemu) x 10 0x100000
0x00100000:      184           0x000000b8
0x00100001:      52            0x00000034
0x00100002:      18            0x00000012
0x00100003:      0             0x00000000
0x00100004:      0             0x00000000
0x00100005:      185           0x000000b9
0x00100006:      39            0x00000027
0x00100007:      0             0x00000000
0x00100008:      16            0x00000010
0x00100009:      0             0x00000000
(nemu)
```

图 3: 扫描内存

2.2 PA1.2

2.3 PA1.3

3 选做思考题

3.1 PA1.1

3.2 PA1.2

3.3 PA1.3

4 实验心得

4.1 PA1.1

4.2 PA1.2

4.3 PA1.3