南京航空航天大学《计算机组成原理Ⅱ课程设计》报告

- 姓名：邵震哲
- 班级：1620204
- 学号：162020130
- 报告阶段：PA1.2&1.3
- 完成日期：2022.4.10
- 本次实验，操作题第14题（加分项）未完成，其他均完成

思考题

1.有什么办法？（5分）

用数据结构的知识，用两个栈，一个存运算符，一个存操作数。根据优先级压栈出栈进行计算。

2.一些简单的正则表达式（10分）

`0x\d{8}`

`[a-z0-9A-Z]+`

`[a-zA-Z_][a-zA-Z0-9_]*` （以字母或下划线开始）

`[0-9]{9} \-[\u4e00-\u9fa5]+ \- PA1.1.pdf`

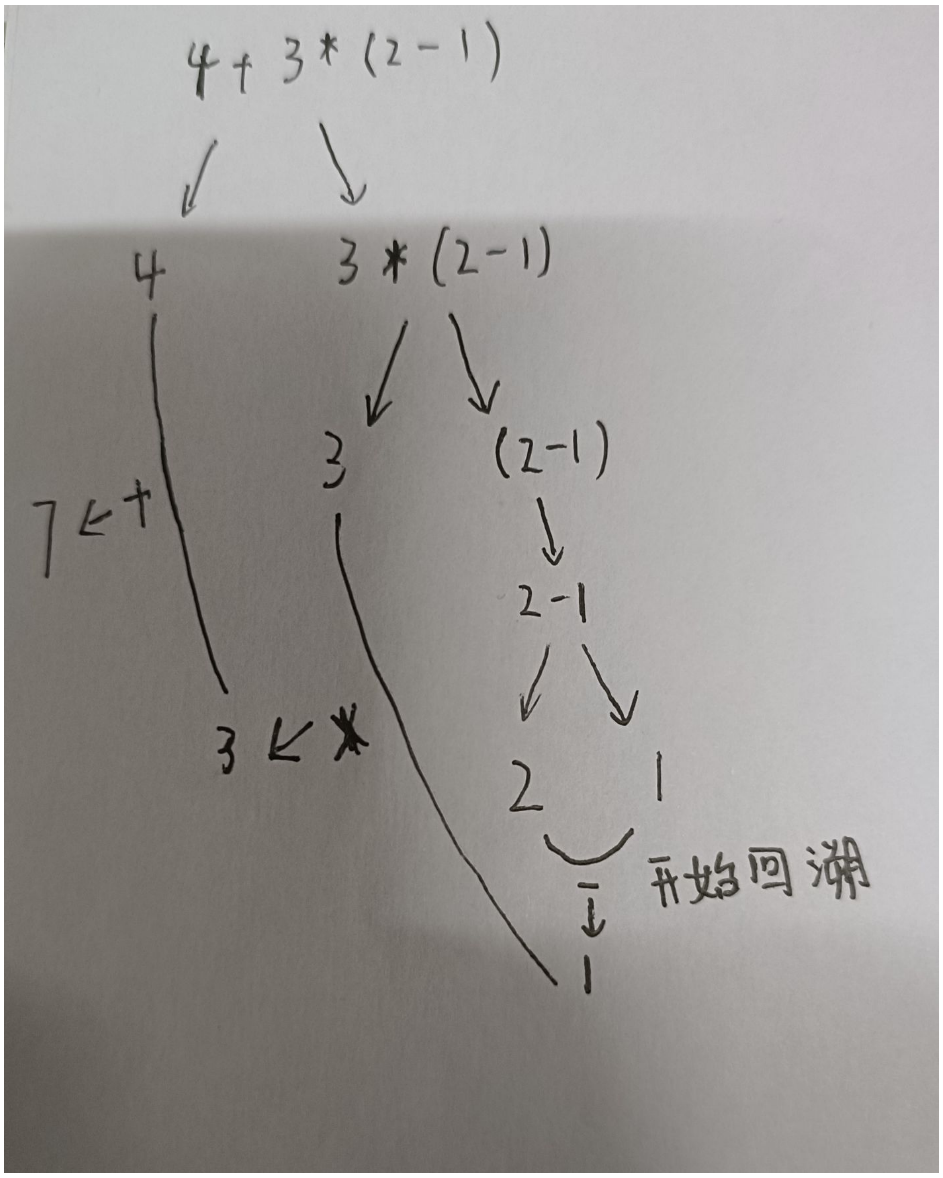
3.这是为什么？（5分）

c语言中如果字符串里想要表达 \ ，需要对其进行转义，\ \ 才能输出一个 \

4.如何处理以上的问题（5分）

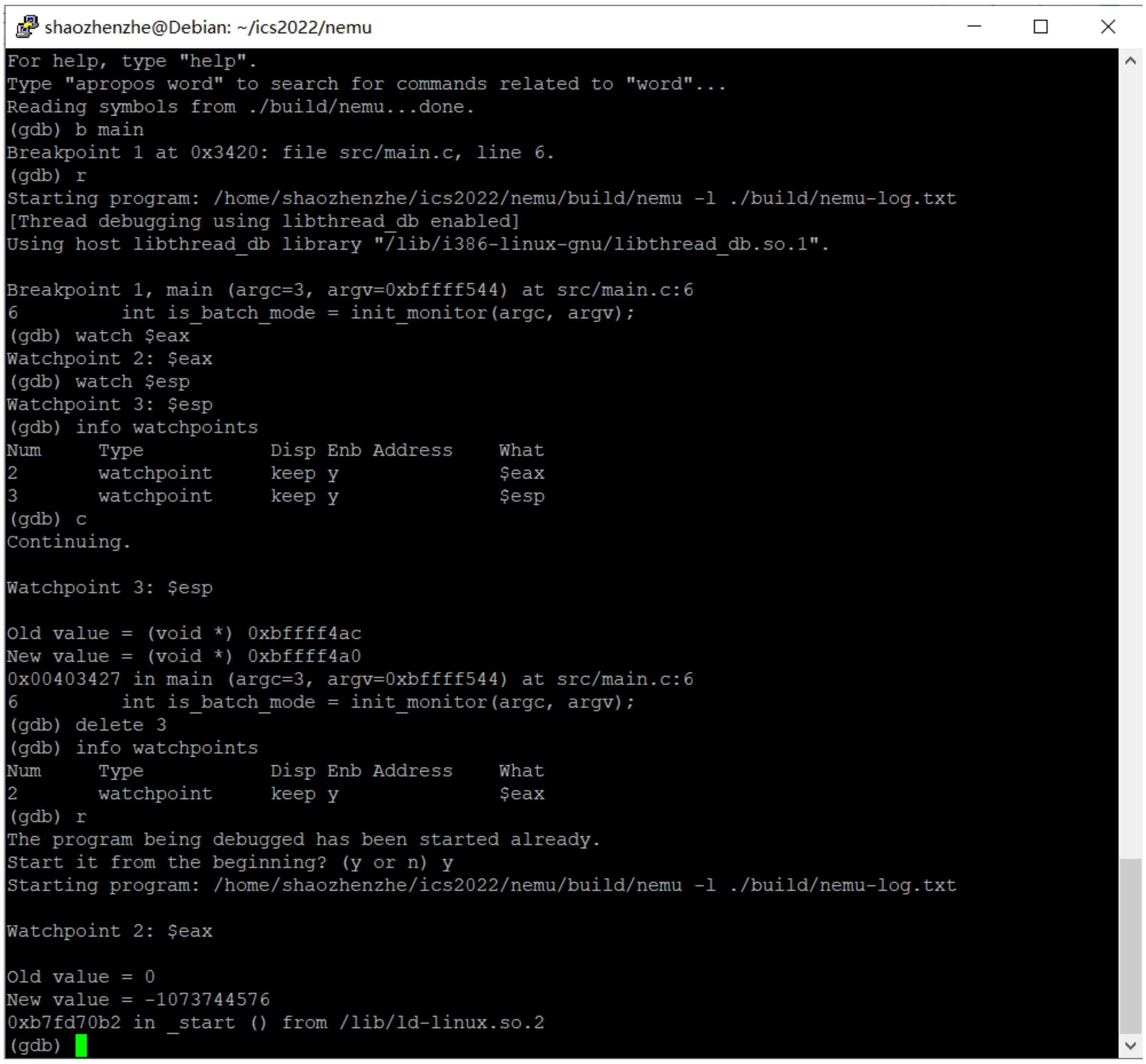
token再添加一个溢出标志成员。存放时计算长度，如果超过str成员长度，设置溢出标志成员，再申请一块空间，把申请出来的地址存放在str数组内。读取时先判断溢出成员标志，如果没溢出就正常读取，如果溢出就读取地址，再根据地址读取内容。

5.递归求值的过程？（5分）



6.体验监视点（5分）

```
b main              //在main函数出处打断点
watch $eax
watch $esp          //添加监视点
info watchpoints    //显示当前所有监视点
c                   //继续运行，使程序命中监视点
delete 3            //删除3号监视点
r                   //重新运行程序，使程序不能在被删除的监视点上命中
```


shaozhenzhe@Debian: ~/ics2022/nemu

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./build/nemu...done.
(gdb) b main
Breakpoint 1 at 0x3420: file src/main.c, line 6.
(gdb) r
Starting program: /home/shaozhenzhe/ics2022/nemu/build/nemu -l ./build/nemu-log.txt
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=3, argv=0xbffff544) at src/main.c:6
6           int is_batch_mode = init_monitor(argc, argv);
(gdb) watch $eax
Watchpoint 2: $eax
(gdb) watch $esp
Watchpoint 3: $esp
(gdb) info watchpoints
Num     Type           Disp Enb Address    What
2       watchpoint     keep y              $eax
3       watchpoint     keep y              $esp
(gdb) c
Continuing.

Watchpoint 3: $esp

Old value = (void *) 0xbffff4ac
New value = (void *) 0xbffff4a0
0x00403427 in main (argc=3, argv=0xbffff544) at src/main.c:6
6           int is_batch_mode = init_monitor(argc, argv);
(gdb) delete 3
(gdb) info watchpoints
Num     Type           Disp Enb Address    What
2       watchpoint     keep y              $eax
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/shaozhenzhe/ics2022/nemu/build/nemu -l ./build/nemu-log.txt

Watchpoint 2: $eax

Old value = 0
New value = -1073744576
0xb7fd70b2 in _start () from /lib/ld-linux.so.2
(gdb)
```

7.科学起名（5分）

和free()函数同名，编译时会报错

8.温故而知新（5分）

在此处的含义是静态全局变量，该变量只能只能在本文件中访问，不能在其它文件中访问。使用static是为了避免它被误修改。

9.一点也不能长？（10分）

不可以正常工作。因为当`int3`指令的长度变成2个字节，设置指令时，原本列表中的地址上的指令码均替换为`int3`指令，但原本的指令都是1字节，`int3`是2字节，会产生错误。将原有指令替换回所有被`int3`所占据的位置，1字节替换2字节，也会有错误。

10.“随心所欲”的断点（10分）

会发生无法检测到断点。因为需要检测读取指令的第一个字节，判断是否等于`0xcc`，如果不在首字节，就无法检测读取。

11.NEMU的前世今生（5分）

`dubugger`是一个命令行调试工具，设置断点，调试程序，测试bug，`emulator`和虚拟机类似，是模拟出一个独立的系统。

nemu是gdb的简单版，gdb可以直接在函数某一行或是函数入口处设置断点，可以随便查看变量当前的值

12.尝试通过目录定位关注的问题（5分）

5.1.3

13.理解基础设施（5分）

75小时，节省50小时

14.查阅i386手册（5分）

- EFLAGS寄存器中的CF位是什么意思?

  P34页中提到参阅附录c，CF是进位标志。P419页：在最高位发生进位或者借位的时候将其置1，否则清零。

- ModR/M字节是什么?

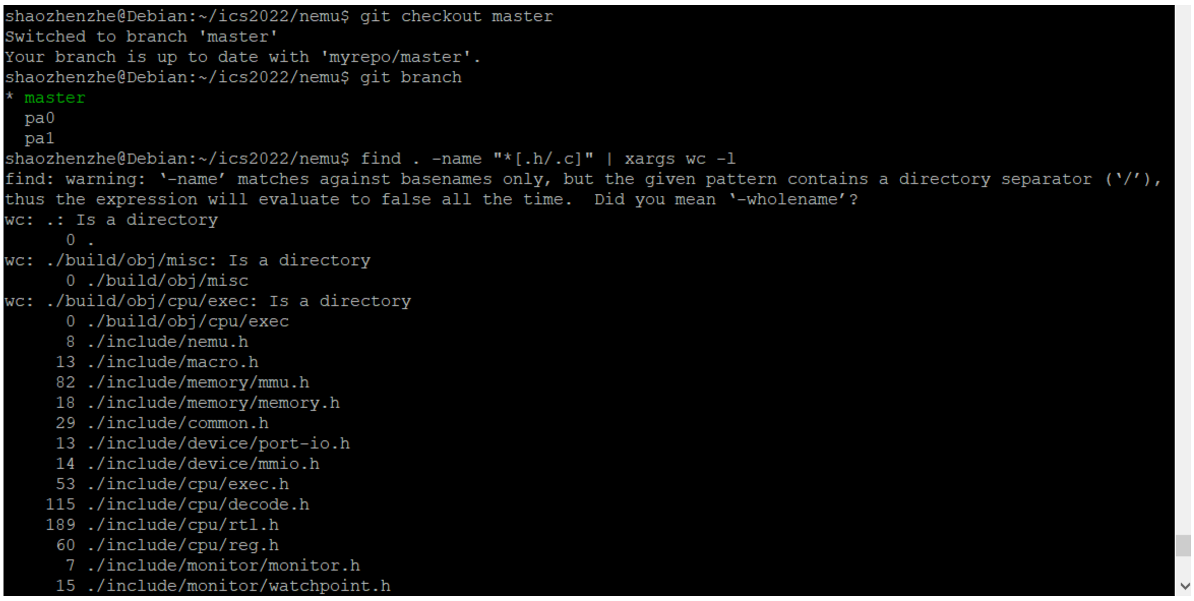
  P241-243页。ModR/M 由 Mod，Reg/Opcode，R/M 三个部分组成。 Mod 是前两位，提供寄存器寻址和内存寻址， Reg/Opcode为3-5位，如果是Reg表示使用哪个寄存器，Opcode表示对group属性的Opcode进行补充； R/M为6-8位，与mod结合起来会得到8个寄存器和24个内存寻址。

- mov指令的具体格式是怎么样的?

  P345页，格式是DEST ← SRC。

15.shell 命令（5分）

```
find . -name "*[.h/.c]" | xargs wc -l    #pa1分支下nemu/目录下的所有.c和.h和文件行数
git checkout master                      #切换到master分支
find . -name "*[.h/.c]" | xargs wc -l    #master分支下行数
```



```
shaozhenzhe@Debian:~/ics2022/nemu$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'myrepo/master'.
shaozhenzhe@Debian:~/ics2022/nemu$ git branch
* master
  pa0
  pa1
shaozhenzhe@Debian:~/ics2022/nemu$ find . -name "*[.h/.c]" | xargs wc -l
find: warning: ‘-name’ matches against basenames only, but the given pattern contains a directory separator (‘/’),
thus the expression will evaluate to false all the time.  Did you mean ‘-wholename’?
wc: .: Is a directory
      0 .
wc: ./build/obj/misc: Is a directory
      0 ./build/obj/misc
wc: ./build/obj/cpu/exec: Is a directory
      0 ./build/obj/cpu/exec
      8 ./include/nemu.h
     13 ./include/macro.h
     82 ./include/memory/mmu.h
     18 ./include/memory/memory.h
     29 ./include/common.h
     13 ./include/device/port-io.h
     14 ./include/device/mmio.h
     53 ./include/cpu/exec.h
    115 ./include/cpu/decode.h
    189 ./include/cpu/rtl.h
     60 ./include/cpu/reg.h
      7 ./include/monitor/monitor.h
     15 ./include/monitor/watchpoint.h
```

得到4069行，和框架代码3536行相比，我在PA1中编写了533行代码

Makefile中写入：

```
.PHONY: …… count

count:
    find . -name "*.c" -o -name "*.h" | xargs cat | grep -v ^$$ | wc -l
```

去除空行的所有.c .h文件行数

```
find . -name "*[.h/.c]" | xargs grep "^." | wc -l
```

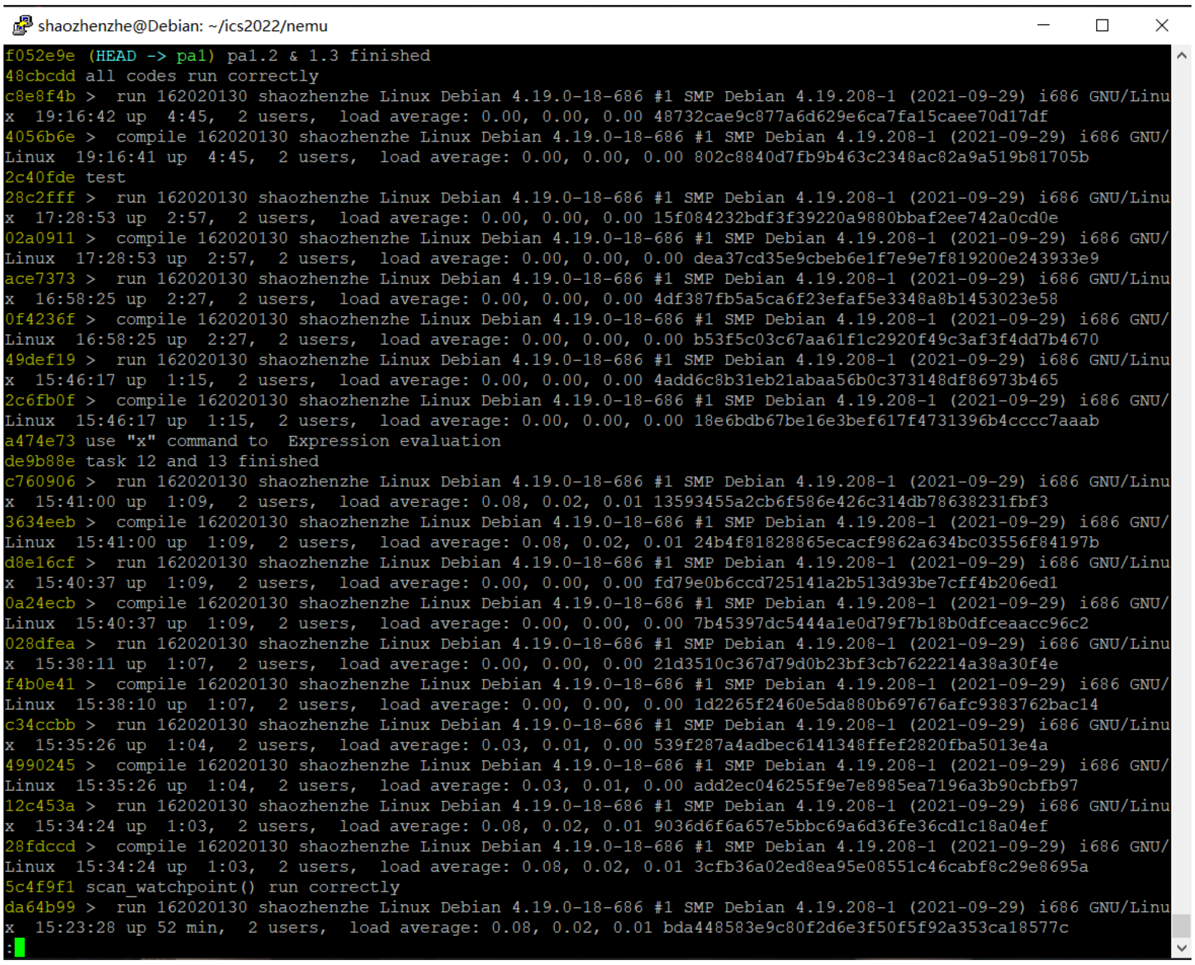
得到3382行

16.使用`man`（5分）

-Wall 使gcc编译后显示所有的警告信息。 -Werror 会将将所有的警告当成错误进行处理，并且取消编译操作。为了找出所有可能造成的错误，尽可能地避免程序运行出错，优化程序。

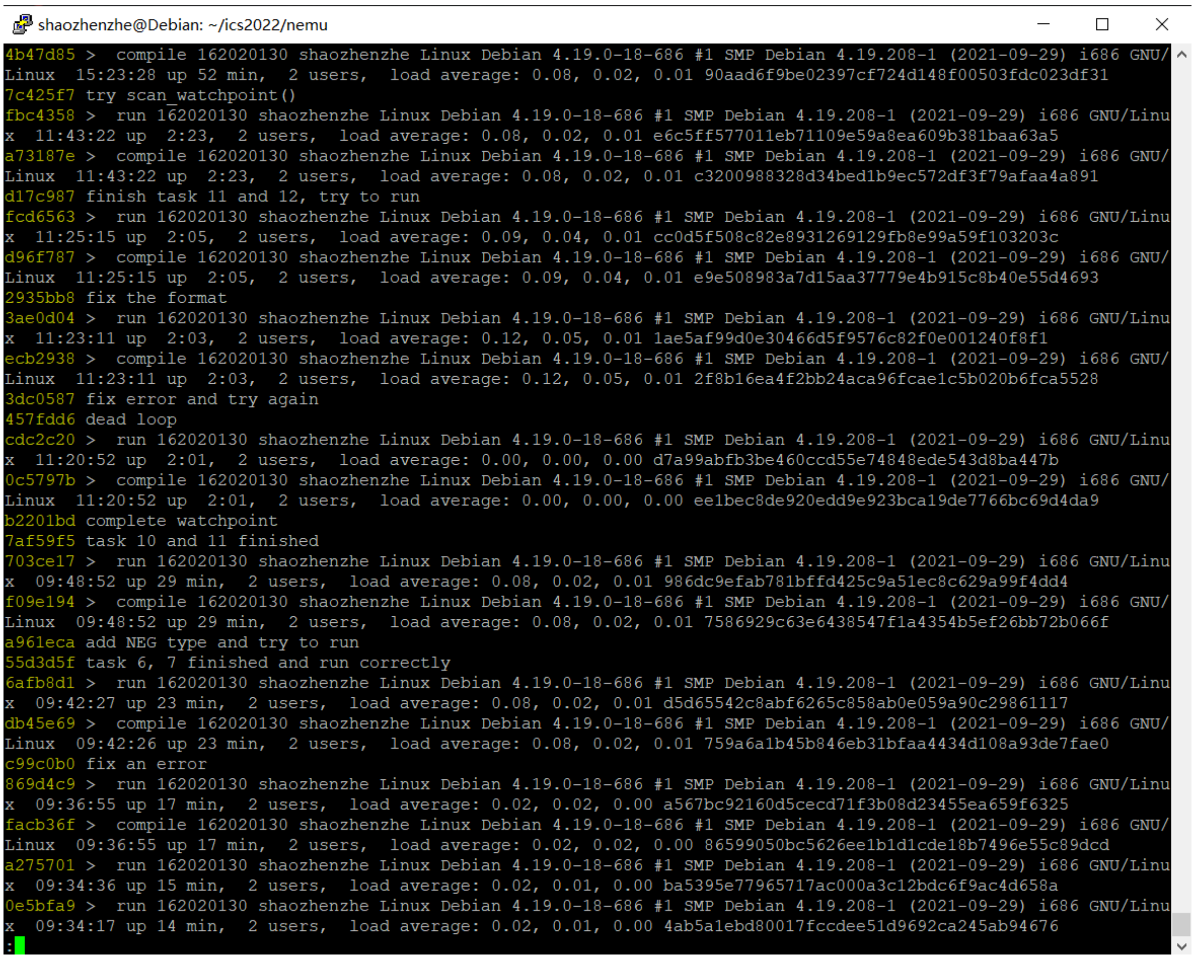
17.`git log`和远程git仓库提交截图（5分）

`git log --oneline`截图



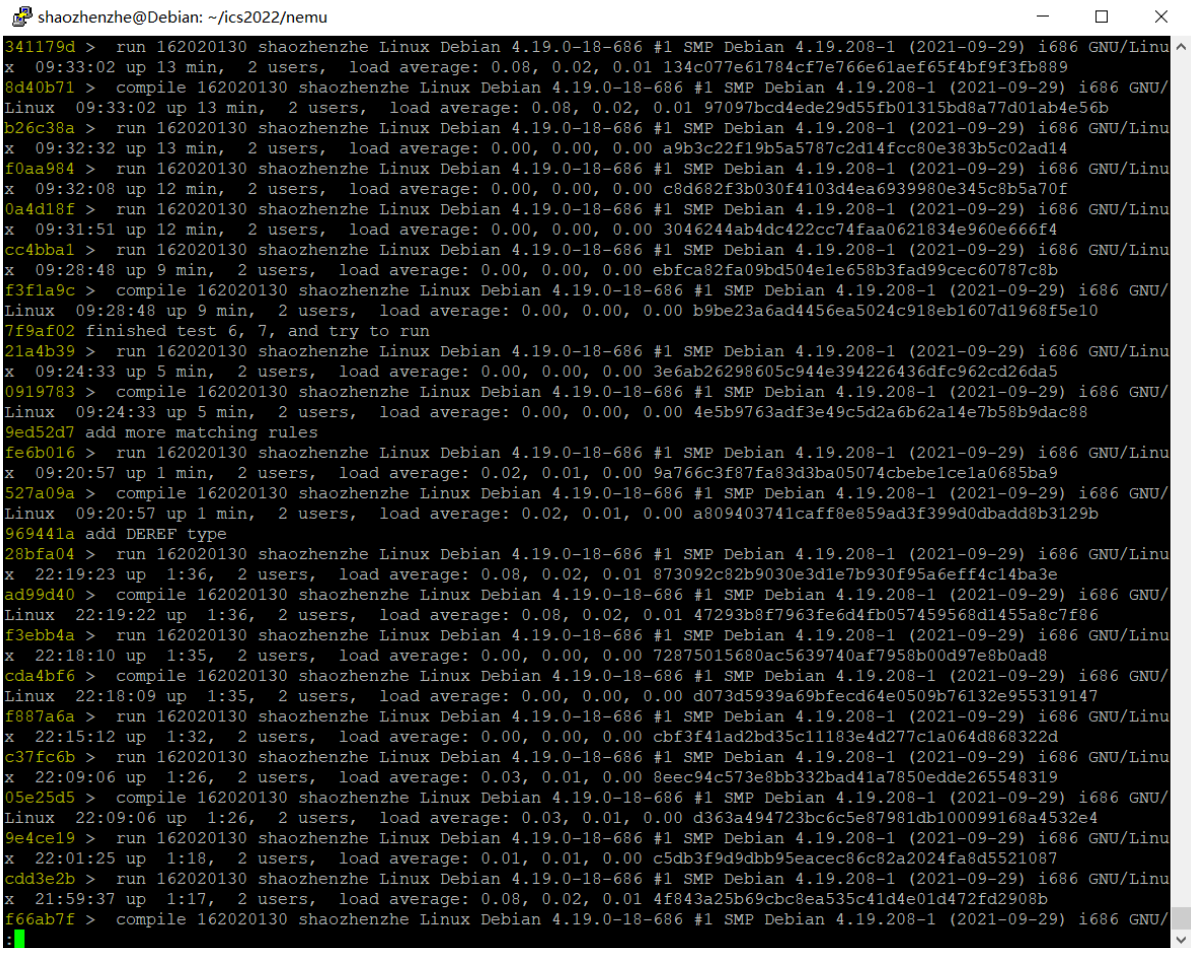
shaozhenzhe@Debian: ~/ics2022/nemu

```
f052e9e (HEAD -> pa1) pa1.2 & 1.3 finished
48cbcdd all codes run correctly
c8e8f4b >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  19:16:42 up  4:45,  2 users,  load average: 0.00, 0.00, 0.00 48732cae9c877a6d629e6ca7fa15caee70d17df
4056b6e >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  19:16:41 up  4:45,  2 users,  load average: 0.00, 0.00, 0.00 802c8840d7fb9b463c2348ac82a9a519b81705b
2c40fde test
28c2fff >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  17:28:53 up  2:57,  2 users,  load average: 0.00, 0.00, 0.00 15f084232bdf3f39220a9880bbaf2ee742a0cd0e
02a0911 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  17:28:53 up  2:57,  2 users,  load average: 0.00, 0.00, 0.00 dea37cd35e9cbeb6e1f7e9e7f819200e243933e9
ace7373 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  16:58:25 up  2:27,  2 users,  load average: 0.00, 0.00, 0.00 4df387fb5a5ca6f23efaf5e3348a8b1453023e58
0f4236f >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  16:58:25 up  2:27,  2 users,  load average: 0.00, 0.00, 0.00 b53f5c03c67aa61f1c2920f49c3af3f4dd7b4670
49def19 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:46:17 up  1:15,  2 users,  load average: 0.00, 0.00, 0.00 4add6c8b31eb21abaa56b0c373148df86973b465
2c6fb0f >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:46:17 up  1:15,  2 users,  load average: 0.00, 0.00, 0.00 18e6bdb67be16e3bef617f4731396b4cccc7aaab
a474e73 use "x" command to  Expression evaluation
de9b88e task 12 and 13 finished
c760906 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:41:00 up  1:09,  2 users,  load average: 0.08, 0.02, 0.01 13593455a2cb6f586e426c314db78638231fbf3
3634eeb >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:41:00 up  1:09,  2 users,  load average: 0.08, 0.02, 0.01 24b4f81828865ecacf9862a634bc03556f84197b
d8e16cf >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:40:37 up  1:09,  2 users,  load average: 0.00, 0.00, 0.00 fd79e0b6ccd725141a2b513d93be7cff4b206ed1
0a24ecb >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:40:37 up  1:09,  2 users,  load average: 0.00, 0.00, 0.00 7b45397dc5444a1e0d79f7b18b0dfceaacc96c2
028dfea >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:38:11 up  1:07,  2 users,  load average: 0.00, 0.00, 0.00 21d3510c367d79d0b23bf3cb7622214a38a30f4e
f4b0e41 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:38:10 up  1:07,  2 users,  load average: 0.00, 0.00, 0.00 1d2265f2460e5da880b697676afc9383762bac14
c34ccbb >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:35:26 up  1:04,  2 users,  load average: 0.03, 0.01, 0.00 539f287a4adbec6141348ffef2820fba5013e4a
4990245 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:35:26 up  1:04,  2 users,  load average: 0.03, 0.01, 0.00 add2ec046255f9e7e8985ea7196a3b90cbfb97
12c453a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:34:24 up  1:03,  2 users,  load average: 0.08, 0.02, 0.01 9036d6f6a657e5bbc69a6d36fe36cd1c18a04ef
28fdccd >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:34:24 up  1:03,  2 users,  load average: 0.08, 0.02, 0.01 3cfb36a02ed8ea95e08551c46cabf8c29e8695a
5c4f9f1 scan_watchpoint() run correctly
da64b99 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:23:28 up 52 min,  2 users,  load average: 0.08, 0.02, 0.01 bda448583e9c80f2d6e3f50f5f92a353ca18577c
:
```



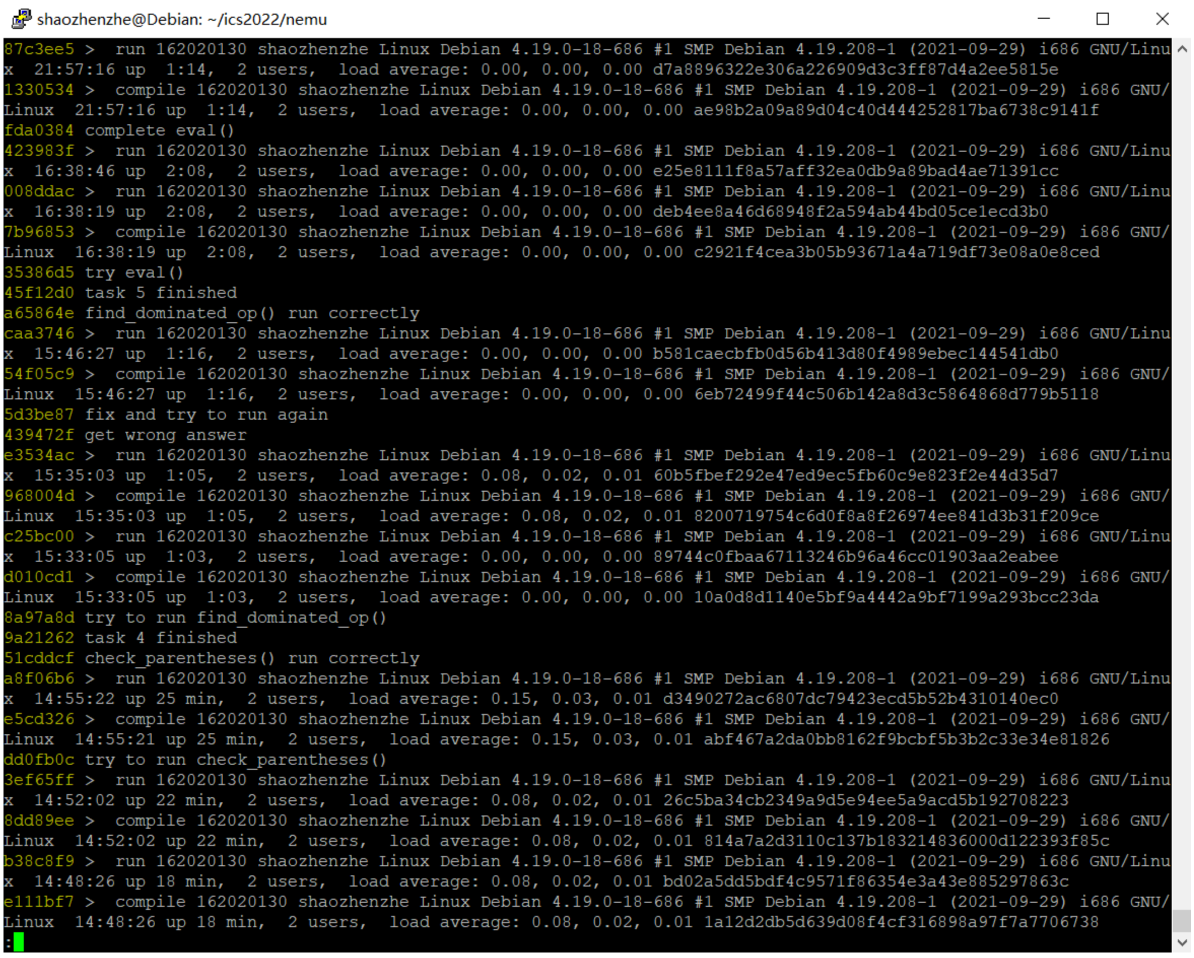
shaozhenzhe@Debian: ~/ics2022/nemu

```
4b47d85 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:23:28 up 52 min,  2 users,  load average: 0.08, 0.02, 0.01 90aad6f9be02397cf724d148f00503fdc023df31
7c425f7 try scan_watchpoint()
fbc4358 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:43:22 up  2:23,  2 users,  load average: 0.08, 0.02, 0.01 e6c5ff577011eb71109e59a8ea609b381baa63a5
a73187e >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:43:22 up  2:23,  2 users,  load average: 0.08, 0.02, 0.01 c3200988328d34bed1b9ec572df3f79afaa4a891
d17c987 finish task 11 and 12, try to run
fcd6563 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:25:15 up  2:05,  2 users,  load average: 0.09, 0.04, 0.01 cc0d5f508c82e8931269129fb8e99a59f103203c
d96f787 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:25:15 up  2:05,  2 users,  load average: 0.09, 0.04, 0.01 e9e508983a7d15aa37779e4b915c8b40e55d4693
2935bb8 fix the format
3ae0d04 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:23:11 up  2:03,  2 users,  load average: 0.12, 0.05, 0.01 1ae5af99d0e30466d5f9576c82f0e001240f8f1
ecb2938 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:23:11 up  2:03,  2 users,  load average: 0.12, 0.05, 0.01 2f8b16ea4f2bb24aca96fcae1c5b020b6fca5528
3dc0587 fix error and try again
457fdd6 dead loop
cdc2c20 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:20:52 up  2:01,  2 users,  load average: 0.00, 0.00, 0.00 d7a99abfb3be460ccd55e74848ede543d8ba447b
0c5797b >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:20:52 up  2:01,  2 users,  load average: 0.00, 0.00, 0.00 ee1bec8de920edd9e923bca19de7766bc69d4da9
b2201bd complete watchpoint
7af59f5 task 10 and 11 finished
703ce17 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:48:52 up 29 min,  2 users,  load average: 0.08, 0.02, 0.01 986dc9efab781bffd425c9a51ec8c629a99f4dd4
f09e194 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:48:52 up 29 min,  2 users,  load average: 0.08, 0.02, 0.01 7586929c63e6438547f1a4354b5ef26bb72b066f
a961eca add NEG type and try to run
55d3d5f task 6, 7 finished and run correctly
6afb8d1 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:42:27 up 23 min,  2 users,  load average: 0.08, 0.02, 0.01 d5d65542c8abf6265c858ab0e059a90c29861117
db45e69 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:42:26 up 23 min,  2 users,  load average: 0.08, 0.02, 0.01 759a6a1b45b846eb31bfaa4434d108a93de7fae0
c99c0b0 fix an error
869d4c9 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:36:55 up 17 min,  2 users,  load average: 0.02, 0.02, 0.00 a567bc92160d5cecd71f3b08d23455ea659f6325
facb36f >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:36:55 up 17 min,  2 users,  load average: 0.02, 0.02, 0.00 86599050bc5626ee1b1d1cde18b7496e55c89dcd
a275701 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:34:36 up 15 min,  2 users,  load average: 0.02, 0.01, 0.00 ba5395e77965717ac000a3c12bdc6f9ac4d658a
0e5bfa9 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:34:17 up 14 min,  2 users,  load average: 0.02, 0.01, 0.00 4ab5a1ebd80017fccdee51d9692ca245ab94676
:
```



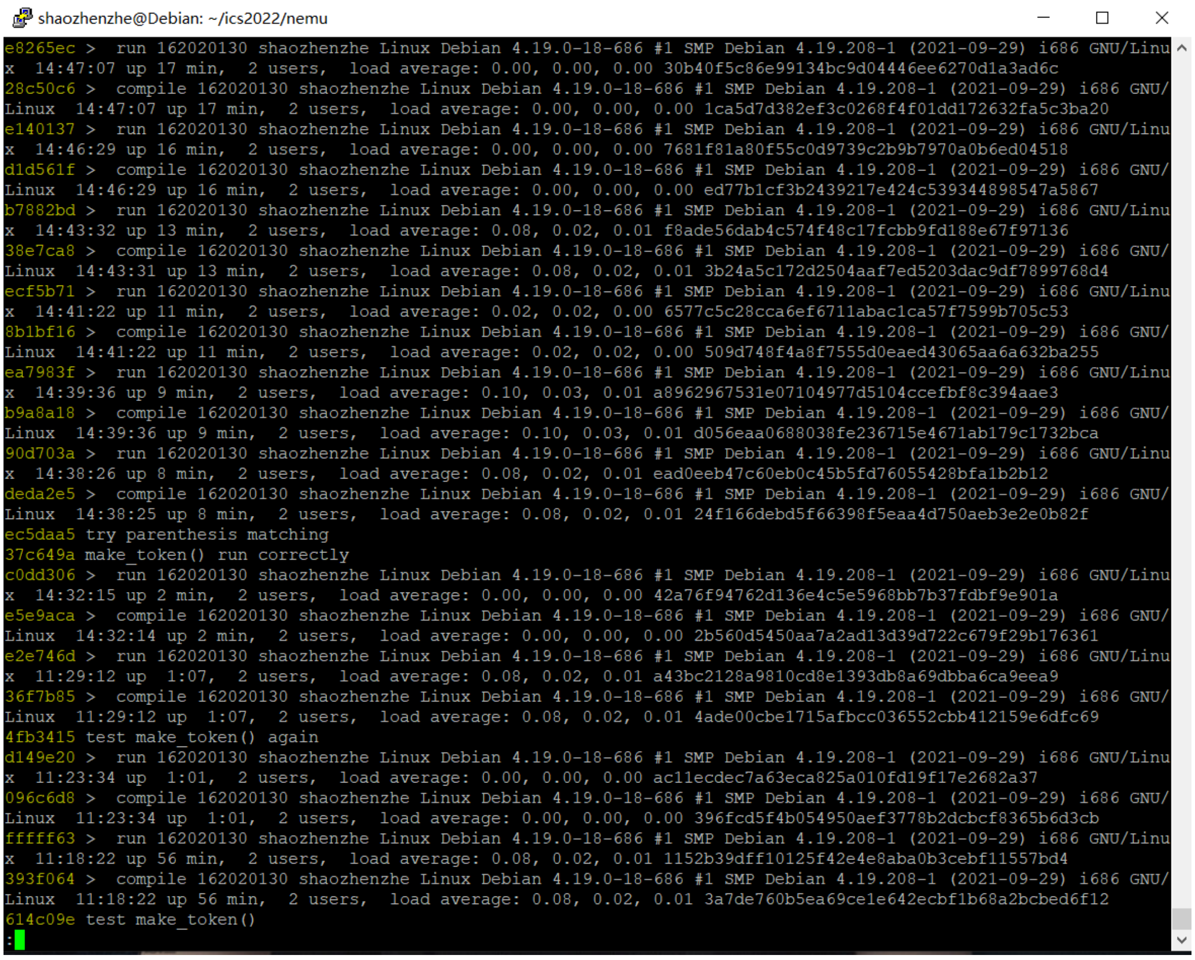
shaozhenzhe@Debian: ~/ics2022/nemu

```
341179d >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:33:02 up 13 min,  2 users,  load average: 0.08, 0.02, 0.01 134c077e61784cf7e766e61aef65f4bf9f3fb889
8d40b71 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:33:02 up 13 min,  2 users,  load average: 0.08, 0.02, 0.01 97097bcd4ede29d55fb01315bd8a77d01ab4e56b
b26c38a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:32:32 up 13 min,  2 users,  load average: 0.00, 0.00, 0.00 a9b3c22f19b5a5787c2d14fcc80e383b5c02ad14
f0aa984 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:32:08 up 12 min,  2 users,  load average: 0.00, 0.00, 0.00 c8d682f3b030f4103d4ea6939980e345c8b5a70f
0a4d18f >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:31:51 up 12 min,  2 users,  load average: 0.00, 0.00, 0.00 3046244ab4dc422cc74faa0621834e960e666f4
cc4bba1 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:28:48 up 9 min,  2 users,  load average: 0.00, 0.00, 0.00 ebfca82fa09bd504e1e658b3fad99cec60787c8b
f3f1a9c >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:28:48 up 9 min,  2 users,  load average: 0.00, 0.00, 0.00 b9be23a6ad4456ea5024c918eb1607d1968f5e10
7f9af02 finished test 6, 7, and try to run
21a4b39 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:24:33 up 5 min,  2 users,  load average: 0.00, 0.00, 0.00 3e6ab26298605c944e394226436dfc962cd26da5
0919783 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:24:33 up 5 min,  2 users,  load average: 0.00, 0.00, 0.00 4e5b9763adf3e49c5d2a6b62a14e7b58b9dac88
9ed52d7 add more matching rules
fe6b016 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:20:57 up 1 min,  2 users,  load average: 0.02, 0.01, 0.00 9a766c3f87fa83d3ba05074cbebe1ce1a0685ba9
527a09a >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:20:57 up 1 min,  2 users,  load average: 0.02, 0.01, 0.00 a809403741caff8e859ad3f399d0dbadd8b3129b
969441a add DEREF type
28bfa04 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:19:23 up  1:36,  2 users,  load average: 0.08, 0.02, 0.01 873092c82b9030e3d1e7b930f95a6eff4c14ba3e
ad99d40 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  22:19:22 up  1:36,  2 users,  load average: 0.08, 0.02, 0.01 47293b8f7963fe6d4fb057459568d1455a8c7f86
f3ebb4a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:18:10 up  1:35,  2 users,  load average: 0.00, 0.00, 0.00 72875015680ac5639740af7958b00d97e8b0ad8
cda4bf6 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  22:18:09 up  1:35,  2 users,  load average: 0.00, 0.00, 0.00 d073d5939a69bfecd64e0509b76132e955319147
f887a6a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:15:12 up  1:32,  2 users,  load average: 0.00, 0.00, 0.00 cbf3f41ad2bd35c11183e4d277c1a064d868322d
c37fc6b >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:09:06 up  1:26,  2 users,  load average: 0.03, 0.01, 0.00 8eec94c573e8bb332bad41a7850edde265548319
05e25d5 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  22:09:06 up  1:26,  2 users,  load average: 0.03, 0.01, 0.00 d363a494723bc6c5e87981db100099168a4532e4
9e4ce19 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:01:25 up  1:18,  2 users,  load average: 0.01, 0.01, 0.00 c5db3f9d9dbb95eacec86c82a2024fa8d5521087
cdd3e2b >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  21:59:37 up  1:17,  2 users,  load average: 0.08, 0.02, 0.01 4f843a25b69cbc8ea535c41d4e01d472fd2908b
f66ab7f >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
:
```



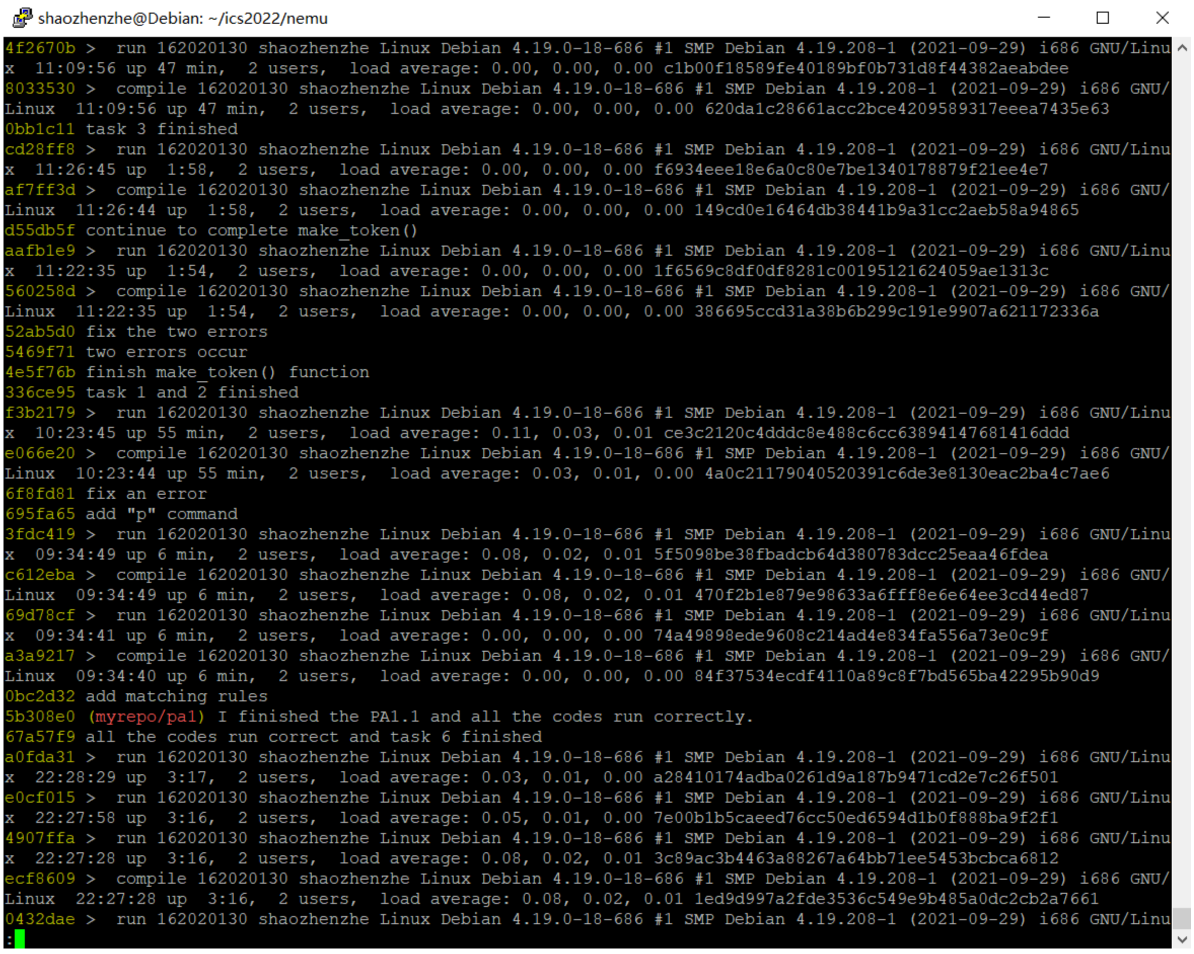
shaozhenzhe@Debian: ~/ics2022/nemu

```
87c3ee5 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  21:57:16 up  1:14,  2 users,  load average: 0.00, 0.00, 0.00 d7a8896322e306a226909d3c3ff87d4a2ee5815e
1330534 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  21:57:16 up  1:14,  2 users,  load average: 0.00, 0.00, 0.00 ae98b2a09a89d04c40d444252817ba6738c9141f
fda0384 complete eval()
423983f >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  16:38:46 up  2:08,  2 users,  load average: 0.00, 0.00, 0.00 e25e8111f8a57aff32ea0db9a89bad4ae71391cc
008ddac >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  16:38:19 up  2:08,  2 users,  load average: 0.00, 0.00, 0.00 deb4ee8a46d68948f2a594ab44bd05ce1ecd3b0
7b96853 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  16:38:19 up  2:08,  2 users,  load average: 0.00, 0.00, 0.00 c2921f4cea3b05b93671a4a719df73e08a0e8ced
35386d5 try eval()
45f12d0 task 5 finished
a65864e find_dominated_op() run correctly
caa3746 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:46:27 up  1:16,  2 users,  load average: 0.00, 0.00, 0.00 b581caecbfb0d56b413d80f4989ebec144541db0
54f05c9 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:46:27 up  1:16,  2 users,  load average: 0.00, 0.00, 0.00 6eb72499f44c506b142a8d3c5864868d779b5118
5d3be87 fix and try to run again
439472f get wrong answer
e3534ac >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:35:03 up  1:05,  2 users,  load average: 0.08, 0.02, 0.01 60b5fbef292e47ed9ec5fb60c9e823f2e44d35d7
968004d >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:35:03 up  1:05,  2 users,  load average: 0.08, 0.02, 0.01 8200719754c6d0f8a8f26974ee841d3b31f209ce
c25bc00 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  15:33:05 up  1:03,  2 users,  load average: 0.00, 0.00, 0.00 89744c0fbaa67113246b96a46cc01903aa2eabee
d010cd1 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  15:33:05 up  1:03,  2 users,  load average: 0.00, 0.00, 0.00 10a0d8d1140e5bf9a4442a9bf7199a293bcc23da
8a97a8d try to run find_dominated_op()
9a21262 task 4 finished
51cddcf check_parentheses() run correctly
a8f06b6 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:55:22 up 25 min,  2 users,  load average: 0.15, 0.03, 0.01 d3490272ac6807dc79423ecd5b52b4310140ec0
e5cd326 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:55:21 up 25 min,  2 users,  load average: 0.15, 0.03, 0.01 abf467a2da0bb8162f9bcbf5b3b2c33e34e81826
dd0fb0c try to run check_parentheses()
3ef65ff >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:52:02 up 22 min,  2 users,  load average: 0.08, 0.02, 0.01 26c5ba34cb2349a9d5e94ee5a9acd5b192708223
8dd89ee >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:52:02 up 22 min,  2 users,  load average: 0.08, 0.02, 0.01 814a7a2d3110c137b183214836000d122393f85c
b38c8f9 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:48:26 up 18 min,  2 users,  load average: 0.08, 0.02, 0.01 bd02a5dd5bdf4c9571f86354e3a43e885297863c
e111bf7 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:48:26 up 18 min,  2 users,  load average: 0.08, 0.02, 0.01 1a12d2db5d639d08f4cf316898a97f7a7706738
:
```



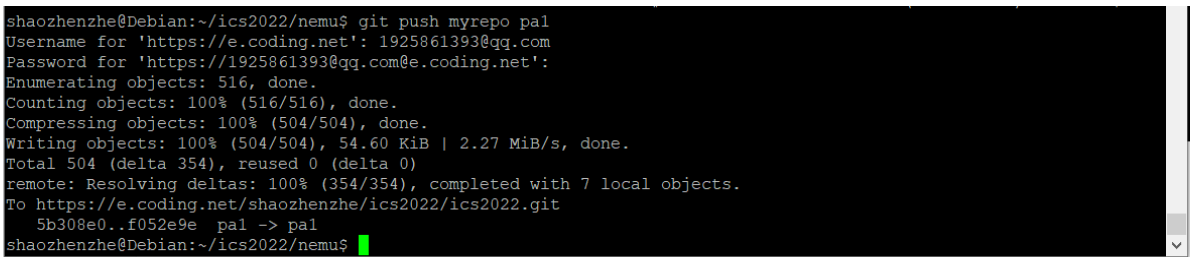
shaozhenzhe@Debian: ~/ics2022/nemu

```
e8265ec >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:47:07 up 17 min,  2 users,  load average: 0.00, 0.00, 0.00 30b40f5c86e99134bc9d04446ee6270d1a3ad6c
28c50c6 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:47:07 up 17 min,  2 users,  load average: 0.00, 0.00, 0.00 1ca5d7d382ef3c0268f4f01dd172632fa5c3ba20
e140137 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:46:29 up 16 min,  2 users,  load average: 0.00, 0.00, 0.00 7681f81a80f55c0d9739c2b9b7970a0b6ed04518
d1d561f >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:46:29 up 16 min,  2 users,  load average: 0.00, 0.00, 0.00 ed77b1cf3b2439217e424c539344898547a5867
b7882bd >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:43:32 up 13 min,  2 users,  load average: 0.08, 0.02, 0.01 f8ade56dab4c574f48c17fcbb9fd188e67f97136
38e7ca8 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:43:31 up 13 min,  2 users,  load average: 0.08, 0.02, 0.01 3b24a5c172d2504aaf7ed5203dac9df7899768d4
ecf5b71 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:41:22 up 11 min,  2 users,  load average: 0.02, 0.02, 0.00 6577c5c28cca6ef6711abac1ca57f7599b705c53
8b1bf16 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:41:22 up 11 min,  2 users,  load average: 0.02, 0.02, 0.00 509d748f4a8f7555d0eaed43065aa6a632ba255
ea7983f >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:39:36 up 9 min,  2 users,  load average: 0.10, 0.03, 0.01 a8962967531e07104977d5104ccefbf8c394aae3
b9a8a18 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:39:36 up 9 min,  2 users,  load average: 0.10, 0.03, 0.01 d056eaa0688038fe236715e4671ab179c1732bca
90d703a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:38:26 up 8 min,  2 users,  load average: 0.08, 0.02, 0.01 ead0eeb47c60eb0c45b5fd76055428bfa1b2b12
deda2e5 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:38:25 up 8 min,  2 users,  load average: 0.08, 0.02, 0.01 24f166debd5f66398f5eaa4d750aeb3e2e0b82f
ec5daa5 try parenthesis matching
37c649a make_token() run correctly
c0dd306 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  14:32:15 up 2 min,  2 users,  load average: 0.00, 0.00, 0.00 42a76f94762d136e4c5e5968bb7b37fdbf9e901a
e5e9aca >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  14:32:14 up 2 min,  2 users,  load average: 0.00, 0.00, 0.00 2b560d5450aa7a2ad13d39d722c679f29b176361
e2e746d >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:29:12 up  1:07,  2 users,  load average: 0.08, 0.02, 0.01 a43bc2128a9810cd8e1393db8a69dbba6ca9eea9
36f7b85 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:29:12 up  1:07,  2 users,  load average: 0.08, 0.02, 0.01 4ade00cbe1715afbcc036552cbb412159e6dfc69
4fb3415 test make_token() again
d149e20 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:23:34 up  1:01,  2 users,  load average: 0.00, 0.00, 0.00 ac11ecdec7a63eca825a010fd19f17e2682a37
096c6d8 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:23:34 up  1:01,  2 users,  load average: 0.00, 0.00, 0.00 396fcd5f4b054950aef3778b2dcbcf8365b6d3cb
fffff63 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:18:22 up 56 min,  2 users,  load average: 0.08, 0.02, 0.01 1152b39dff10125f42e4e8aba0b3cebf11557bd4
393f064 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:18:22 up 56 min,  2 users,  load average: 0.08, 0.02, 0.01 3a7de760b5ea69ce1e642ecbf1b68a2bcbed6f12
614c09e test make_token()
:
```

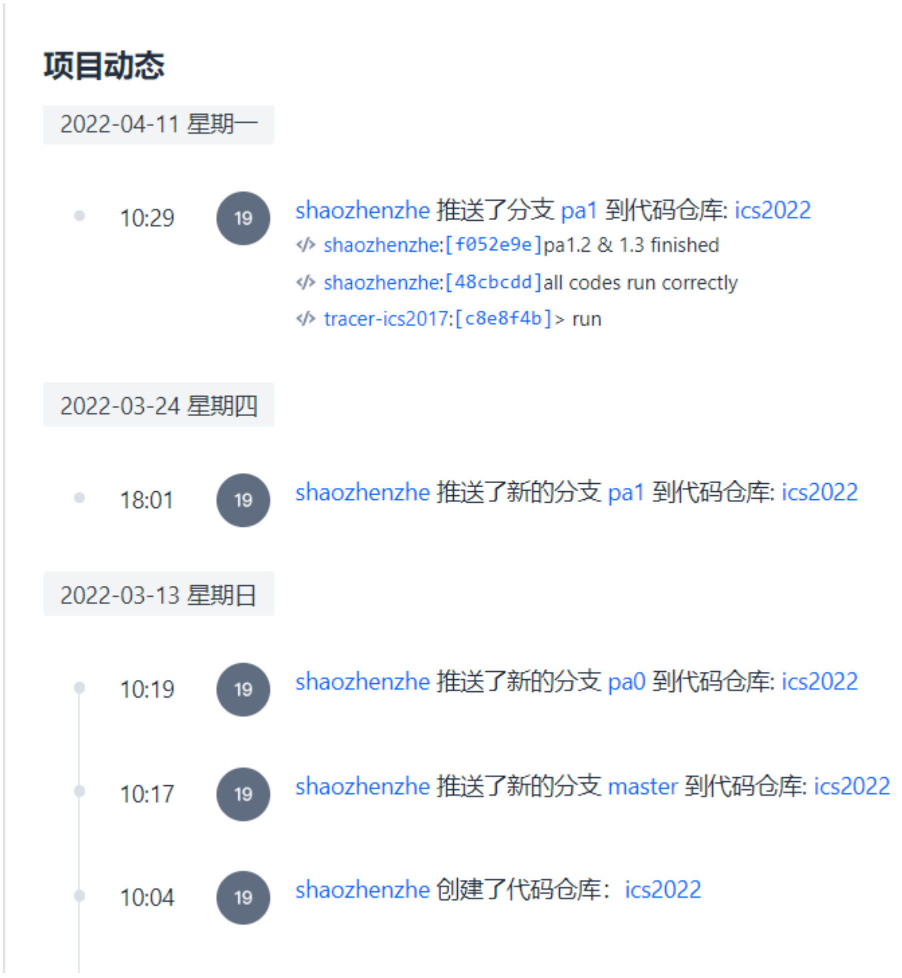


```
shaozhenzhe@Debian: ~/ics2022/nemu
4f2670b >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:09:56 up 47 min,  2 users,  load average: 0.00, 0.00, 0.00 c1b00f18589fe40189bf0b731d8f44382aeabdee
8033530 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:09:56 up 47 min,  2 users,  load average: 0.00, 0.00, 0.00 620da1c28661acc2bce4209589317eeea7435e63
0bb1c11 task 3 finished
cd28ff8 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:26:45 up  1:58,  2 users,  load average: 0.00, 0.00, 0.00 f6934eee18e6a0c80e7be1340178879f21ee4e7
af7ff3d >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:26:44 up  1:58,  2 users,  load average: 0.00, 0.00, 0.00 149cd0e16464db38441b9a31cc2aeb58a94865
d55db5f continue to complete make_token()
aafb1e9 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  11:22:35 up  1:54,  2 users,  load average: 0.00, 0.00, 0.00 1f6569c8df0df8281c00195121624059ae1313c
560258d >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  11:22:35 up  1:54,  2 users,  load average: 0.00, 0.00, 0.00 386695ccd31a38b6b299c191e9907a621172336a
52ab5d0 fix the two errors
5469f71 two errors occur
4e5f76b finish make_token() function
336ce95 task 1 and 2 finished
f3b2179 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  10:23:45 up 55 min,  2 users,  load average: 0.11, 0.03, 0.01 ce3c2120c4dddc8e488c6cc63894147681416ddd
e066e20 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  10:23:44 up 55 min,  2 users,  load average: 0.03, 0.01, 0.00 4a0c21179040520391c6de3e8130eac2ba4c7ae6
6f8fd81 fix an error
695fa65 add "p" command
3fdc419 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:34:49 up 6 min,  2 users,  load average: 0.08, 0.02, 0.01 5f5098be38fbadcb64d380783dcc25eaa46fdea
c612eba >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:34:49 up 6 min,  2 users,  load average: 0.08, 0.02, 0.01 470f2b1e879e98633a6fff8e6e64ee3cd44ed87
69d78cf >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  09:34:41 up 6 min,  2 users,  load average: 0.00, 0.00, 0.00 74a49898ede9608c214ad4e834fa556a73e0c9f
a3a9217 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  09:34:40 up 6 min,  2 users,  load average: 0.00, 0.00, 0.00 84f37534ecdf4110a89c8f7bd565ba42295b90d9
0bc2d32 add matching rules
5b308e0 (myrepo/pa1) I finished the PA1.1 and all the codes run correctly.
67a57f9 all the codes run correct and task 6 finished
a0fda31 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:28:29 up  3:17,  2 users,  load average: 0.03, 0.01, 0.00 a28410174adba0261d9a187b9471cd2e7c26f501
e0cf015 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:27:58 up  3:16,  2 users,  load average: 0.05, 0.01, 0.00 7e00b1b5caeed76cc50ed6594d1b0f888ba9f2f1
4907ffa >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
x  22:27:28 up  3:16,  2 users,  load average: 0.08, 0.02, 0.01 3c89ac3b4463a88267a64bb71ee5453bcbca6812
ecf8609 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/
Linux  22:27:28 up  3:16,  2 users,  load average: 0.08, 0.02, 0.01 1ed9d997a2fde3536c549e9b485a0dc2cb2a7661
0432dae >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i686 GNU/Linu
:
```

远程git仓库提交截图



```
shaozhenzhe@Debian:~/ics2022/nemu$ git push myrepo pa1
Username for 'https://e.coding.net': 1925861393@qq.com
Password for 'https://1925861393@qq.com@e.coding.net':
Enumerating objects: 516, done.
Counting objects: 100% (516/516), done.
Compressing objects: 100% (504/504), done.
Writing objects: 100% (504/504), 54.60 KiB | 2.27 MiB/s, done.
Total 504 (delta 354), reused 0 (delta 0)
remote: Resolving deltas: 100% (354/354), completed with 7 local objects.
To https://e.coding.net/shaozhenzhe/ics2022/ics2022.git
   5b308e0..f052e9e  pa1 -> pa1
shaozhenzhe@Debian:~/ics2022/nemu$
```


项目动态

2022-04-11 星期一

10:29 shaozhenzhe 推送了分支 pa1 到代码仓库: ics2022
shaozhenzhe:[f052e9e]pa1.2 & 1.3 finished
shaozhenzhe:[48cbcdd]all codes run correctly
tracer-ics2017:[c8e8f4b]> run

2022-03-24 星期四

18:01 shaozhenzhe 推送了新的分支 pa1 到代码仓库: ics2022

2022-03-13 星期日

10:19 shaozhenzhe 推送了新的分支 pa0 到代码仓库: ics2022

10:17 shaozhenzhe 推送了新的分支 master 到代码仓库: ics2022

10:04 shaozhenzhe 创建了代码仓库: ics2022

实验内容

1. 编写匹配规则(1)

`/nemu/src/monitor/debug/expr.c`

在enum添加数据类型，rule添加正则匹配规则（这里注意hex的匹配规则要在十进制的匹配规则之前，否则匹配0x开头的十六进制时会先匹配十进制）

```
enum {
  TK_NOTYPE = 256,
  TK_EQ = 0,

  /* TODO: Add more token types */
  TK_NUM = 1,
  TK_HEX = 2,
  TK_REG = 3,
};

static struct rule {
  char *regex;
  int token_type;
} rules[] = {

  /* TODO: Add more rules.
   * Pay attention to the precedence level of different rules.
   */

  {" +", TK_NOTYPE},    // spaces
```

```
    {"\\+", '+'},         // plus
    {"==", TK_EQ},        // equal
    {"\\-", '-'},         // -
    {"\\*", '*'},         // *
    {"\\/", '/'},         // /

    {"0x[0-9,a-f]+", TK_HEX},     //hex
    {"[0-9]+", TK_NUM},           //num
    {"\\$[a-z]{2,3}", TK_REG},    //register name

    {"\\(", '('},         // (
    {"\\)", ')'},         // )

};
```

2. 添加 p 命令

先往`cmd_table`里添加p命令，再根据`expr.h`头文件只包含了`expr`函数可知，`cmd_p()`里面需要调用`expr()`函数得到结果

`/nemu/src/monitor/debug/ui.c`

```
static struct {
  char *name;
  char *description;
  int (*handler) (char *);
} cmd_table [] = {
  { "help", "Display informations about all supported commands", cmd_help },
  { "c", "Continue the execution of the program", cmd_c },
  { "q", "Exit NEMU", cmd_q },
  { "si", "One Step", cmd_si },
  { "info", "Display informations about all regisiters", cmd_info},
  { "x", "Scan memory", cmd_x},
  /* TODO: Add more commands */
  { "p", "Expression evaluation", cmd_p},

};



static int cmd_p(char* args){
    bool* success=false;
    uint32_t result;
    result = expr(args,success);
    printf("%d\n", result);
    return 0;
}
```

需要把`expr`函数里的`TODO()`去掉，否则会报错

3. 识别并存储 token

在给出的框架中，for循环用`regexec()`函数匹配目标文本串和前面定义的`rules[i]`中的正则表达式比较，成功识别得到对应规则后，存储匹配到的`token`，类型赋给`tokens[nr_token].type`，数据复制到`tokens[nr_token].str`中，会用到`strcpy`或者`strncpy`函数，其中`substr_start`代表匹配开始的位置，`substr_len`表示读取长度。最后`nr_token++`

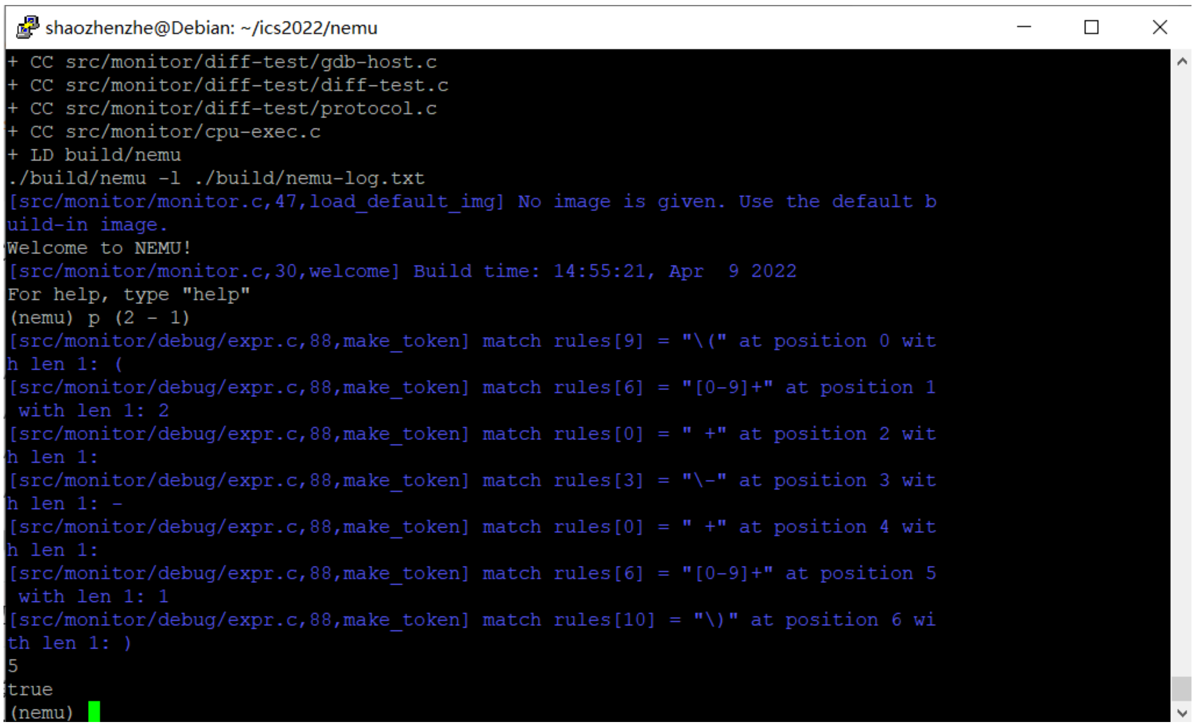
`/nemu/src/monitor/debug/expr.c`

```
static bool make_token(char *e) {
  int position = 0;
  int i;
  regmatch_t pmatch;

  nr_token = 0;

  while (e[position] != '\0') {
    /* Try all rules one by one. */
    for (i = 0; i < NR_REGEX; i ++) {
      if (regexec(&re[i], e + position, 1, &pmatch, 0) == 0 && pmatch.rm_so ==
0) {
        char *substr_start = e + position;
        int substr_len = pmatch.rm_eo;

        Log("match rules[%d] = \"%s\" at position %d with len %d: %.*s",
            i, rules[i].regex, position, substr_len, substr_len, substr_start);
        position += substr_len;

        /* TODO: Now a new token is recognized with rules[i]. Add codes
         * to record the token in the array `tokens'. For certain types
         * of tokens, some extra actions should be performed.
         */

        switch (rules[i].token_type) {  //根据匹配到的type存储
          case '+':
            tokens[nr_token].type = '+';
            nr_token++;
            break;
          case '-':
            tokens[nr_token].type = '-';
            nr_token++;
            break;
          case '*':
            tokens[nr_token].type = '*';
            nr_token++;
            break;
          case '/':
            tokens[nr_token].type = '/';
            nr_token++;
            break;
          case '(':
            tokens[nr_token].type = '(';
            nr_token++;
            break;
          case ')':
            tokens[nr_token].type = ')';
            nr_token++;
            break;

          case 256:
            break;
          case 0:
            tokens[nr_token].type = 0;
            strcpy(tokens[nr_token].str, "==");
            nr_token++;
            break;
          case 1:
            tokens[nr_token].type = 1;
            strncpy(tokens[nr_token].str, substr_start, substr_len);
            nr_token++;
            break;
          case 2:
            tokens[nr_token].type = 2;
            strncpy(tokens[nr_token].str, substr_start, substr_len);
            nr_token++;
            break;
          case 3:
            tokens[nr_token].type = 3;
            strncpy(tokens[nr_token].str, substr_start, substr_len);
            nr_token++;
            break;
          default:
            assert(0);
        }

        break;
      }
    }

    if (i == NR_REGEX) {
      printf("no match at position %d\n%s\n%*.s^\n", position, e, position, "");
      return false;
    }
  }

  return true;
}
```

测试时`expr`函数改写为如下，如果`make_token`错误，就会输出`false`

```
uint32_t expr(char *e, bool *success) {
  if (!make_token(e)) {
    *success = false;
    printf("make_token() false\n"); //make_token错误，输出错误
    return 0;
  }
  /* TODO: Insert codes to evaluate the expression. */
  return 0;
}
```

测试结果：未出现false，说明正确


shaozhenzhe@Debian: ~/ics2022/nemu

```
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/cpu-exec.c
+ LD build/nemu
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 14:55:21, Apr  9 2022
For help, type "help"
(nemu) p (2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 wit
h len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1
 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 wit
h len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 3 wit
h len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 wit
h len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5
 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 6 wi
th len 1: )
5
true
(nemu)
```

4. 实现括号匹配

用count来记录匹配对数，遇到左括号+1，右括号-1

只有最后一个位置是 `)` 且 `count==0` 才是匹配，其他情况都不匹配

`/nemu/src/monitor/debug/expr.c`

```c
bool check_parentheses(int p, int q){
    int count=0;
    if(tokens[p].type == '('){
        for(int i=p; i<=q; i++){
            if(tokens[i].type == '('){
                count++;
            }
            if(tokens[i].type == ')'){
                count--;
            }
            if(count == 0 && i!=q){
                return false;
            }
        }
        if(count == 0)
            return true;
        else
            return false;
    }
    return false;
}
```

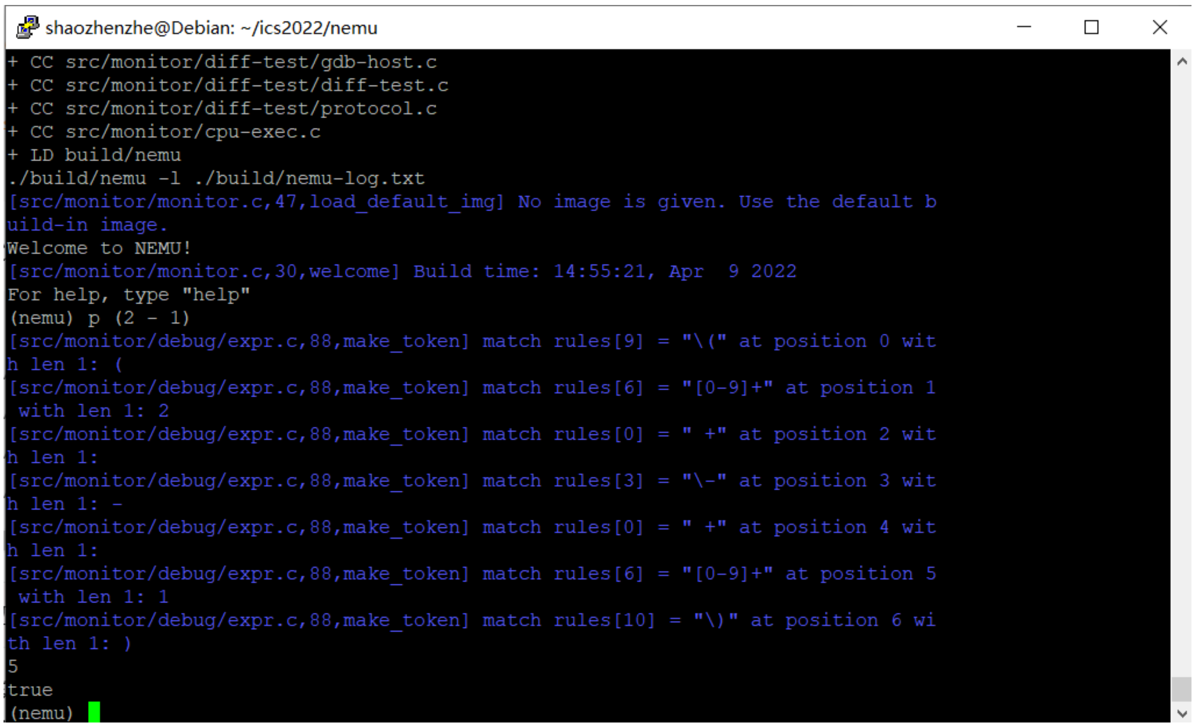
测试时可以将expr()函数写为如下，如果满足括号匹配会输出true，不满足输出false

```c
uint32_t expr(char *e, bool *success) {
  if (!make_token(e)) {
    *success = false;
    return 0;
  }
  printf("%d\n", nr_token);
  /* TODO: Insert codes to evaluate the expression. */
  bool flag = check_parentheses(0, nr_token-1);
  if(flag == false) printf("false\n");
  else printf("true\n");
  return 0;
}
```

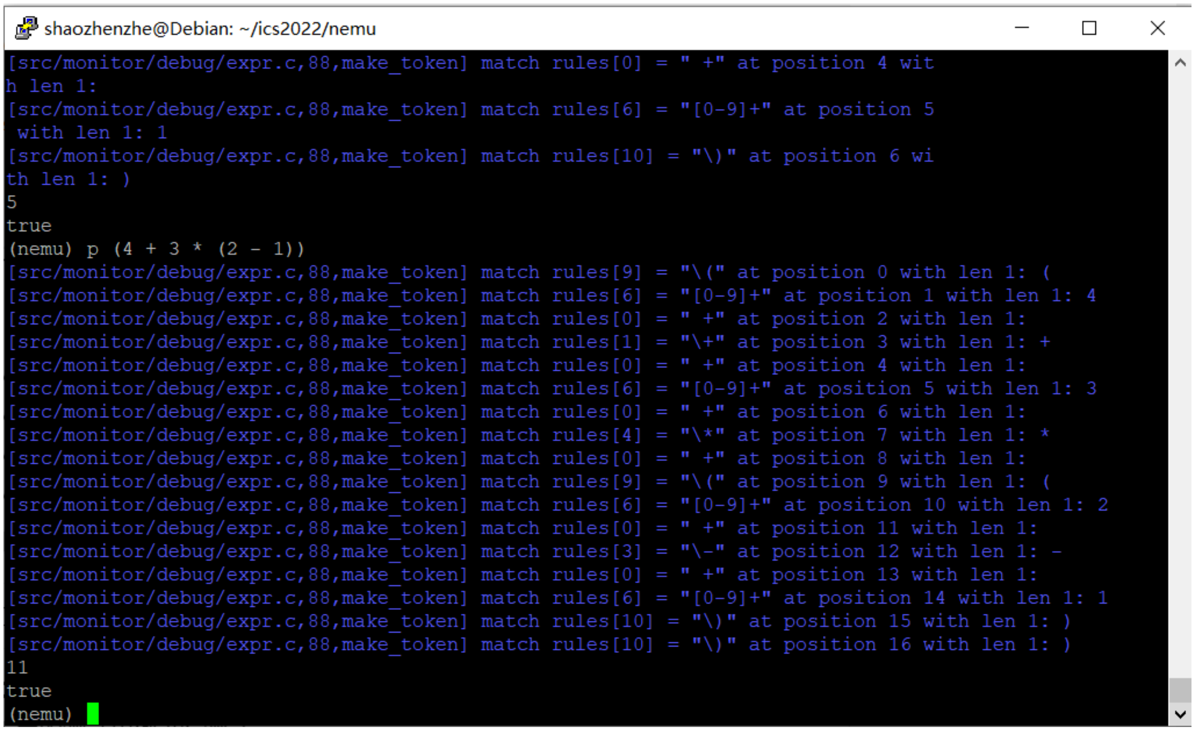
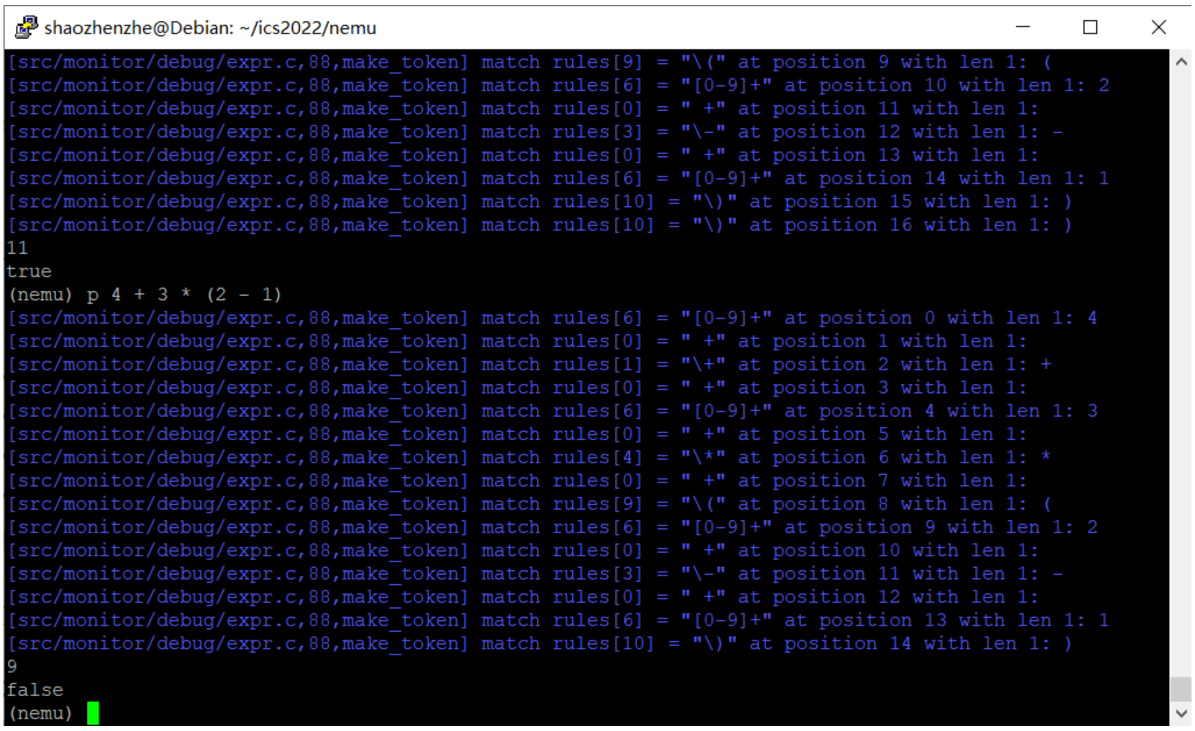
测试用例

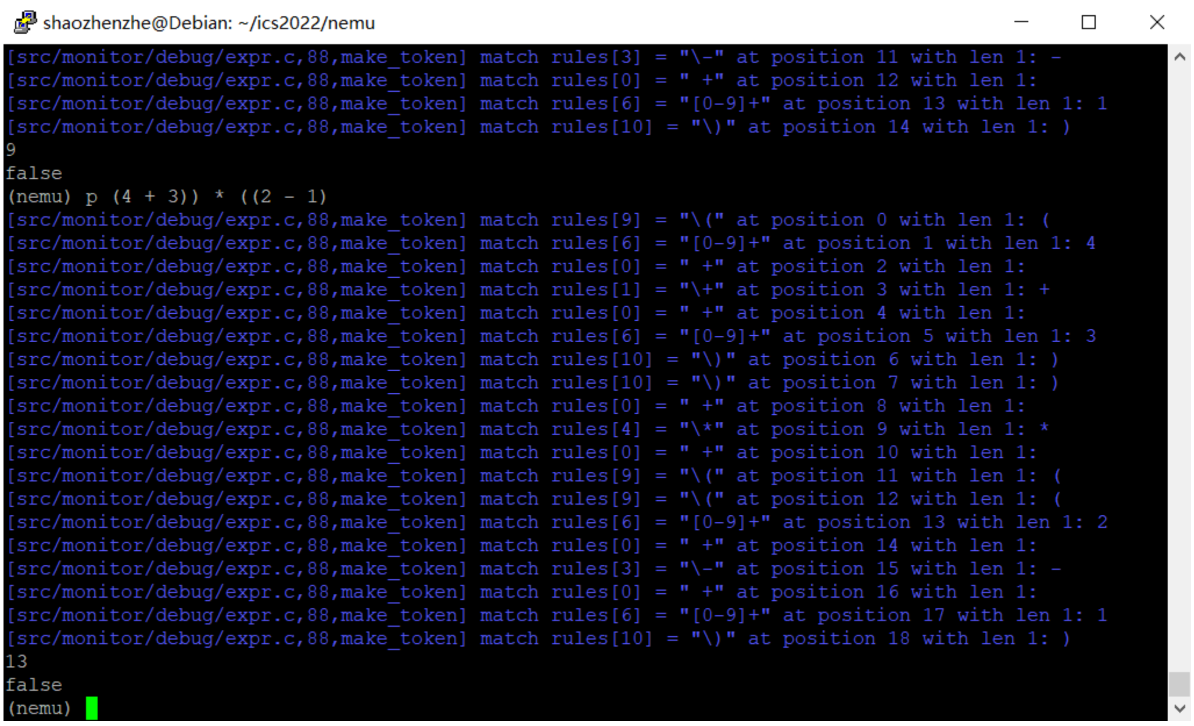
> "(2 - 1)"   // true
> "(4 + 3 * (2 - 1))"   // true
> "4 + 3 * (2 - 1)"   // false, the whole expression is not surrounded by a matched pair of parentheses
> "(4 + 3)) * ((2 - 1)" // false, bad expression
> "(4 + 3) * (2 - 1)"   // false, the leftmost '(' and the rightmost ')' are not matched

测试结果：均正确输出

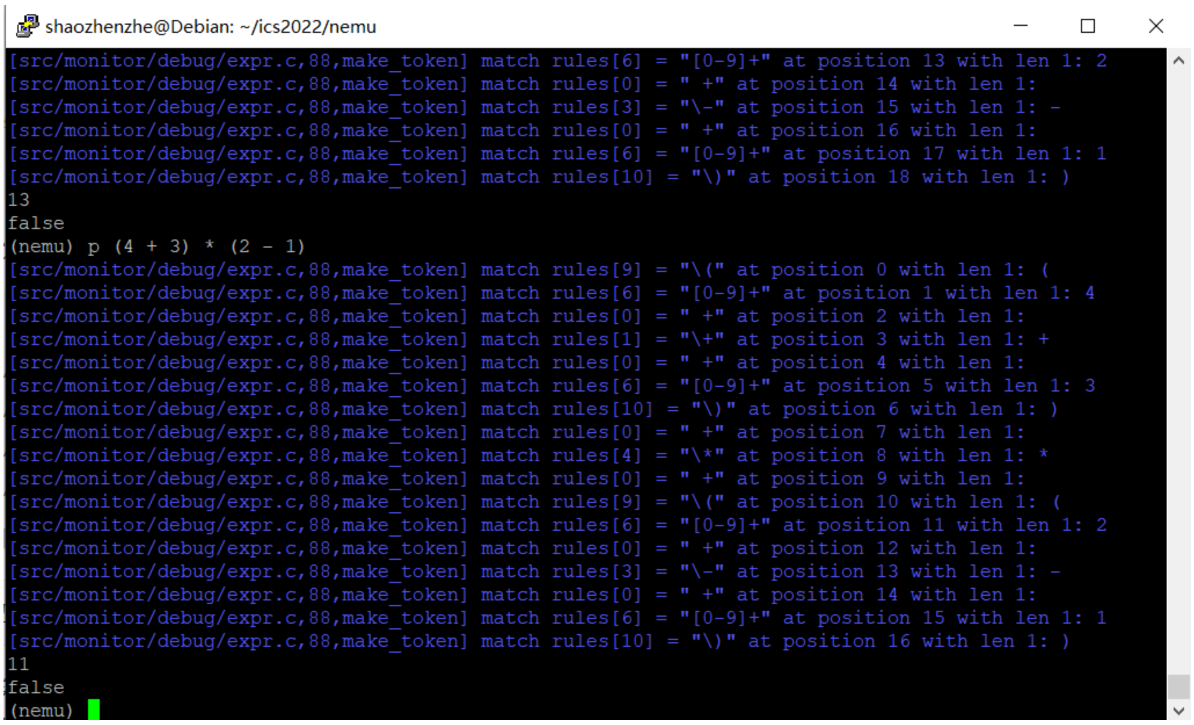


```
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/cpu-exec.c
+ LD build/nemu
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 14:55:21, Apr  9 2022
For help, type "help"
(nemu) p (2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 wit
h len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1
 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 wit
h len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 3 wit
h len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 wit
h len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5
 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 6 wi
th len 1: )
5
true
(nemu)
```


shaozhenzhe@Debian: ~/ics2022/nemu
```
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 wit
h len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5
 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 6 wi
th len 1: )
5
true
(nemu) p (4 + 3 * (2 - 1))
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 6 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[4] = "\*" at position 7 with len 1: *
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 9 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 10 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 12 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 13 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 14 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 15 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 16 with len 1: )
11
true
(nemu)
```

shaozhenzhe@Debian: ~/ics2022/nemu
```
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 9 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 10 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 12 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 13 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 14 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 15 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 16 with len 1: )
11
true
(nemu) p 4 + 3 * (2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 0 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 2 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 3 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 4 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 5 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[4] = "\*" at position 6 with len 1: *
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 7 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 8 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 9 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 10 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 11 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 12 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 13 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 14 with len 1: )
9
false
(nemu)
```



```
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 11 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 12 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 13 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 14 with len 1: )
9
false
(nemu) p (4 + 3)) * ((2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 6 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 7 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[4] = "\*" at position 9 with len 1: *
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 10 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 11 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 12 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 13 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 14 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 15 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 16 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 17 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 18 with len 1: )
13
false
(nemu) 
```



```
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 13 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 14 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 15 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 16 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 17 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 18 with len 1: )
13
false
(nemu) p (4 + 3) * (2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 6 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 7 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[4] = "\*" at position 8 with len 1: *
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 9 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 10 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 11 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 12 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 13 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 14 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 15 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 16 with len 1: )
11
false
(nemu) 
```

5. 实现子表达式拆分

最后一步进行运行的运算符肯定在括号外，所以和第四步类似，先忽略括号内的内容，对在括号外的运算符进行优先级判断

这里进行了==， +- 和 */ 的优先级判断，priority越小，优先级越低

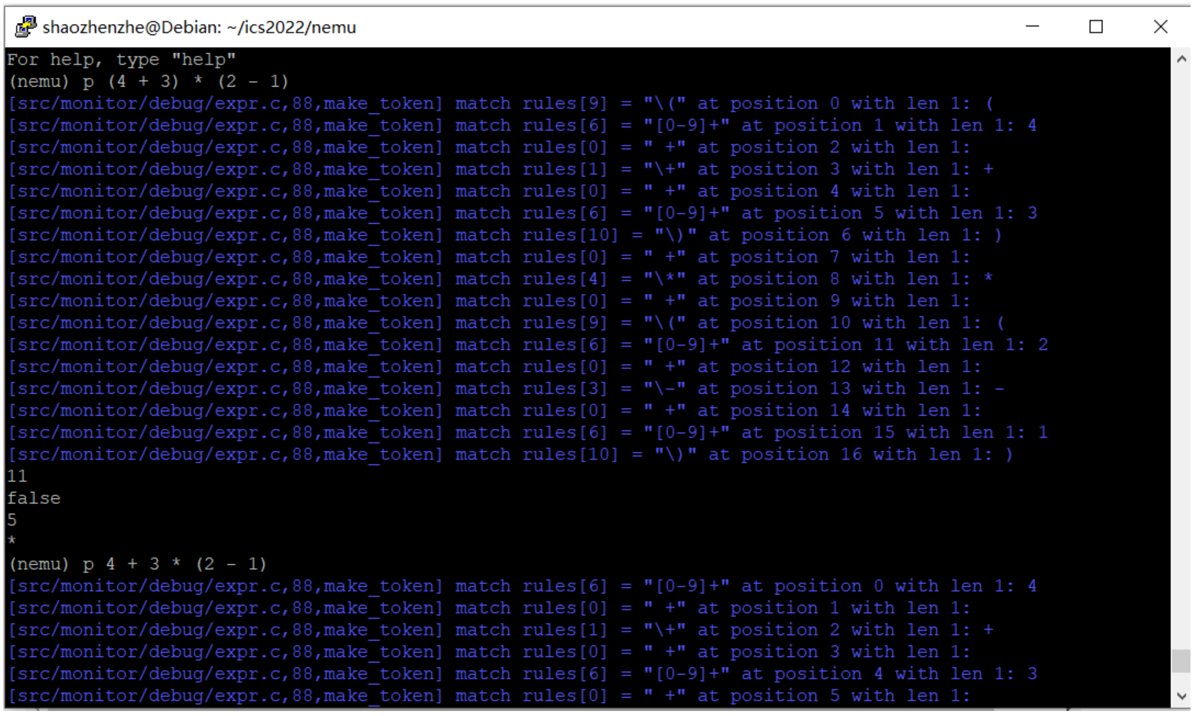
`/nemu/src/monitor/debug/expr.c`

```c
int find_dominated_op(int p, int q){
    int op = -1;
    int count = 0;
    int priority = 999;
    for(int i=p; i<=q; i++){
        if(tokens[i].type == '('){
            count++;
        }
        if(tokens[i].type == ')'){
            count--;
        }

        if(count == 0){
            if(tokens[i].type == 0){    // ==
                if(priority >= 0){
                    priority = 0;
                    op = i;
                }
            }
            else if(tokens[i].type == '+' || tokens[i].type == '-'){
                if(priority >= 1){
                    priority = 1;
                    op = i;
                }
            }
            else if(tokens[i].type == '*' || tokens[i].type == '/'){
                if(priority >= 2){
                    priority = 2;
                    op = i;
                }
            }

        }
    }
    return op;
}
```

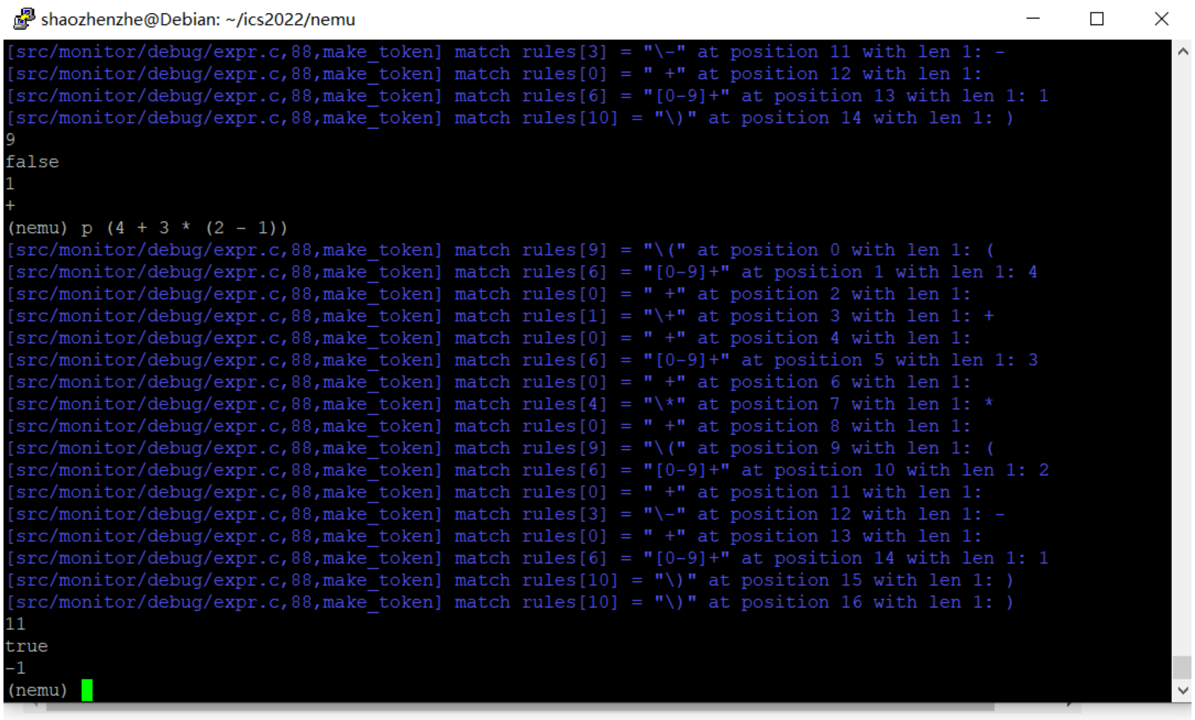
测试时把`expr`函数改为如下，可以输出中心操作符和位置进行验证

```
uint32_t expr(char *e, bool *success) {
  if (!make_token(e)) {
    *success = false;
    return 0;
  }
  printf("%d\n", nr_token);
  /* TODO: Insert codes to evaluate the expression. */
  bool flag = check_parentheses(0, nr_token-1);
  if(flag == false) printf("false\n");
  else printf("true\n");

  int op = find_dominated_op(0, nr_token-1);
  printf("%d\n", op);
  if(op != -1)  printf("%c\n", tokens[op].type);    //如果能拆分，输出中心操作符
  return 0;
}
```

测试结果：正确输出


shaozhenzhe@Debian: ~/ics2022/nemu

```
For help, type "help"
(nemu) p (4 + 3) * (2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 6 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 7 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[4] = "\*" at position 8 with len 1: *
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 9 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 10 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 11 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 12 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 13 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 14 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 15 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 16 with len 1: )
11
false
5
*
(nemu) p 4 + 3 * (2 - 1)
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 0 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 2 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 3 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 4 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 5 with len 1:
```


shaozhenzhe@Debian: ~/ics2022/nemu

```
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 11 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 12 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 13 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 14 with len 1: )
9
false
1
+
(nemu) p (4 + 3 * (2 - 1))
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[1] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 6 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[4] = "\*" at position 7 with len 1: *
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[9] = "\(" at position 9 with len 1: (
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 10 with len 1: 2
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[3] = "\-" at position 12 with len 1: -
[src/monitor/debug/expr.c,88,make_token] match rules[0] = " +" at position 13 with len 1:
[src/monitor/debug/expr.c,88,make_token] match rules[6] = "[0-9]+" at position 14 with len 1: 1
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 15 with len 1: )
[src/monitor/debug/expr.c,88,make_token] match rules[10] = "\)" at position 16 with len 1: )
11
true
-1
(nemu) 
```

6. 实现表达式求值

这题的测试用例还包含指针解引用，因此需要同步完成第七题

`p==q`时，根据`type`读取返回数据即可，寄存器要分类

最后else情况里分情况，如果没有中心操作符，说明是单个操作符，有可能是指针引用（其他情况都在其他`if`分支里解决了）

读取指针引用的后一个内容，分十进制，十六进制和寄存器讨论，用`vaddr_read()`函数读取指向的内存内容

如果有中心操作符，那就调用`find_dominated_op()`进行拆分，递归求解拆分后的两部分，根据操作符进行对应计算

`/nemu/src/monitor/debug/expr.c`

```
uint32_t eval(int p, int q) {
    int num;
    int op = -1;
    int val1,val2;
    vaddr_t address;
    if (p > q) {
        /* Bad expression */
        assert(0);
    }
    else if (p == q) {  //单个token，分情况得到数据
        /* Single token.
        * For now this token should be a number.
        * Return the value of the number.
        */
        if(tokens[p].type == 1){
            sscanf(tokens[p].str, "%d", &num);
            return num;
        }
        else if(tokens[p].type == 2){
            sscanf(tokens[p].str, "%x", &num);
            return num;
        }
        else if(tokens[p].type == 3){
            if(strcmp(tokens[p].str, "$eax") == 0){
                return cpu.eax;
            }
            else if(strcmp(tokens[p].str, "$ecx") == 0){
                return cpu.ecx;
            }
            else if(strcmp(tokens[p].str, "$edx") == 0){
                return cpu.edx;
            }
            else if(strcmp(tokens[p].str, "$ebx") == 0){
                return cpu.ebx;
            }
            else if(strcmp(tokens[p].str, "$esp") == 0){
                return cpu.esp;
            }
            else if(strcmp(tokens[p].str, "$ebp") == 0){
                return cpu.ebp;
            }
            else if(strcmp(tokens[p].str, "$esi") == 0){
                return cpu.esi;
            }
            else if(strcmp(tokens[p].str, "$edi") == 0){
                return cpu.edi;
            }
            else if(strcmp(tokens[p].str, "$eip") == 0){
                return cpu.eip;
            }
            else{
                assert(0);
            }
        }
    }
    else if (check_parentheses(p, q) == true) {
        /* The expression is surrounded by a matched pair of parentheses.
        * If that is the case, just throw away the parentheses.
        */
        return eval(p + 1, q - 1);
    }
    else {
        op = find_dominated_op(p, q);

        if(op == -1){   //没有中心操作符，即一个操作符
            if(tokens[p].type == DEREF){    //指针解引用
                if(tokens[q].type == TK_REG){   //寄存器，例如 *$eax
                    if(strcmp(tokens[q].str, "$eax") == 0){
                        return vaddr_read(cpu.eax, 4);
                    }
                    else if(strcmp(tokens[q].str, "$ecx") == 0){
                        return vaddr_read(cpu.ecx, 4);
                    }
                    else if(strcmp(tokens[q].str, "$edx") == 0){
                        return vaddr_read(cpu.edx, 4);
                    }
                    else if(strcmp(tokens[q].str, "$ebx") == 0){
                        return vaddr_read(cpu.ebx, 4);
                    }
                    else if(strcmp(tokens[q].str, "$esp") == 0){
                        return vaddr_read(cpu.esp, 4);
                    }
                    else if(strcmp(tokens[q].str, "$ebp") == 0){
                        return vaddr_read(cpu.ebp, 4);
                    }
                    else if(strcmp(tokens[q].str, "$esi") == 0){
                        return vaddr_read(cpu.esi, 4);
                    }
                    else if(strcmp(tokens[q].str, "$edi") == 0){
                        return vaddr_read(cpu.edi, 4);
                    }
                    else if(strcmp(tokens[q].str, "$eip") == 0){
                        return vaddr_read(cpu.eip, 4);
                    }
                    else{
                        //printf("%s\n", tokens[q].str);
                        //printf("here\n");
                        assert(0);
                    }
                }
                else if(tokens[q].type == 1){   //十进制地址地址，例如 *1000
                    sscanf(tokens[q].str, "%d", &address);
                    return vaddr_read(address, 4);
                }
                else if(tokens[q].type == 2){  //十六进制地址，例如 *0x100000
                    sscanf(tokens[q].str, "%x", &address);
                    return vaddr_read(address, 4);
                }
            }
        }

        val1 = eval(p, op - 1);
        val2 = eval(op + 1, q);
        switch (tokens[op].type) {
            case '+': return val1 + val2;
            case '-': return val1 - val2;
            case '*': return val1 * val2;
            case '/': return val1 / val2;
            case 0:
                if(val1 == val2) return 1;
                else return 0;
            default: assert(0);
        }
    }
    return 0;
}
```

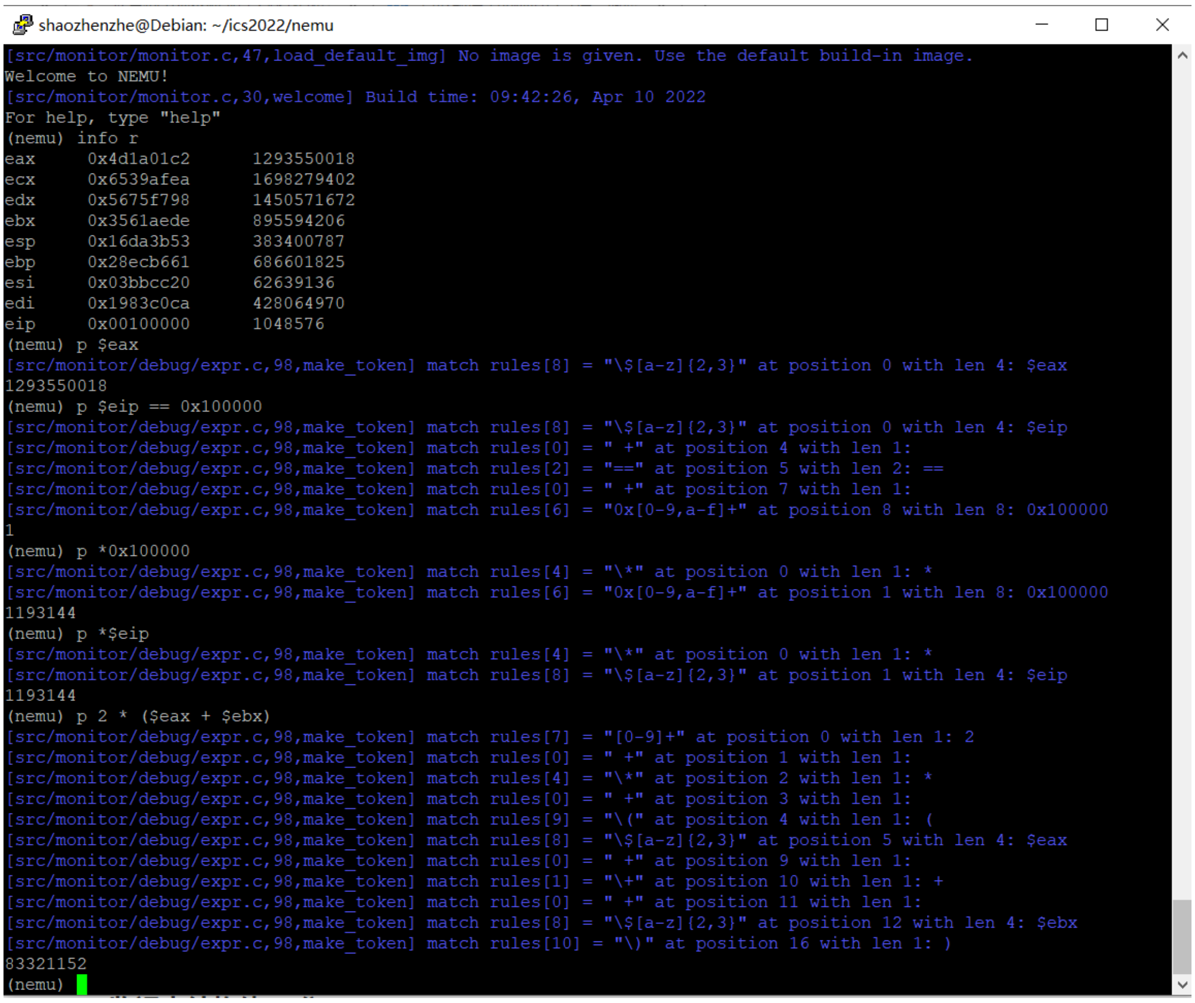
确定 * 为指针还是乘号需要在`expr()`函数里，完成`make_token()`后，进行`eval()`前。对所有token遍历，如果 * 处于第一位或者前一个token为 `(` ，说明这里的 * 是指针解引用

```
uint32_t expr(char *e, bool *success) {
  if (!make_token(e)) {
    *success = false;
    return 0;
  }
  //printf("%d\n", nr_token);
  /* TODO: Insert codes to evaluate the expression. */
  for (int i = 0; i < nr_token; i ++) {
    if (tokens[i].type == '*' && (i == 0 || tokens[i - 1].type == '(') ) {  //判断是否为指针解引用
        tokens[i].type = DEREF;
    }
}
  uint32_t result;
  result = eval(0, nr_token-1);
  //printf("result is %d\n", result);
  return result;
}
```

测试用例：（先进行 info r 命令）

> p $eax
> p $eip == 0x100000
> p *0x100000
> p *$eip
> p 2 * ($eax + $ebx)

测试结果：正确输出



```
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 09:42:26, Apr 10 2022
For help, type "help"
(nemu) info r
eax     0x4d1a01c2      1293550018
ecx     0x6539afea      1698279402
edx     0x5675f798      1450571672
ebx     0x3561aede      895594206
esp     0x16da3b53      383400787
ebp     0x28ecb661      686601825
esi     0x03bbcc20      62639136
edi     0x1983c0ca      428064970
eip     0x00100000      1048576
(nemu) p $eax
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 0 with len 4: $eax
1293550018
(nemu) p $eip == 0x100000
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 0 with len 4: $eip
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[2] = "==" at position 5 with len 2: ==
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 7 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[6] = "0x[0-9,a-f]+" at position 8 with len 8: 0x100000
1
(nemu) p *0x100000
[src/monitor/debug/expr.c,98,make_token] match rules[4] = "\*" at position 0 with len 1: *
[src/monitor/debug/expr.c,98,make_token] match rules[6] = "0x[0-9,a-f]+" at position 1 with len 8: 0x100000
1193144
(nemu) p *$eip
[src/monitor/debug/expr.c,98,make_token] match rules[4] = "\*" at position 0 with len 1: *
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 1 with len 4: $eip
1193144
(nemu) p 2 * ($eax + $ebx)
[src/monitor/debug/expr.c,98,make_token] match rules[7] = "[0-9]+" at position 0 with len 1: 2
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[4] = "\*" at position 2 with len 1: *
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 3 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[9] = "\(" at position 4 with len 1: (
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 5 with len 4: $eax
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 9 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[1] = "\+" at position 10 with len 1: +
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 12 with len 4: $ebx
[src/monitor/debug/expr.c,98,make_token] match rules[10] = "\)" at position 16 with len 1: )
83321152
(nemu)
```

7. 实现指针解引用

在第六题里已经实现

8. 实现负数

和指针解引用一样，在`expr()`里判断 - 是负号还是减号

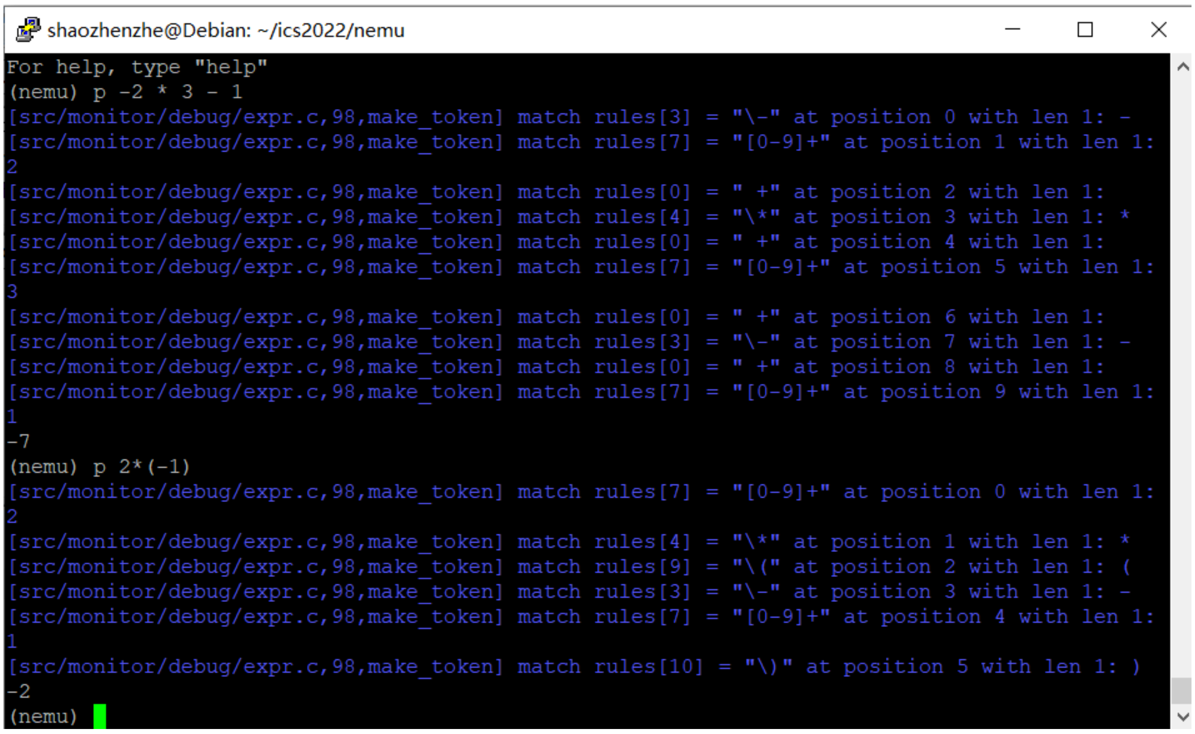
`/nemu/src/monitor/debug/expr.c`

```
uint32_t expr(char *e, bool *success) {
  if (!make_token(e)) {
    *success = false;
    return 0;
  }
  //printf("%d\n", nr_token);
  /* TODO: Insert codes to evaluate the expression. */
  for (int i = 0; i < nr_token; i ++) {
    if (tokens[i].type == '*' && (i == 0 || tokens[i - 1].type == '(') ) {  //判断是否为指针解引用
        tokens[i].type = DEREF;
    }
    if (tokens[i].type == '-' && (i == 0 || tokens[i - 1].type == '(') ) {  //判断是否为负数
        tokens[i].type = NEG;
    }
}
  uint32_t result;
  result = eval(0, nr_token-1);
  //printf("result is %d\n", result);
  return result;
}
```

`eval()`里也要有对应的负号情况，这里就把增加的负数部分的代码贴上来，在 `op == -1`，`tokens[p].type == NEG`的情况下分类即可

```
uint32_t eval(int p, int q) {
    //……
    else{
        //……
        if(op == -1){
            if(tokens[p].type == NEG){      //负数
                if(tokens[q].type == TK_REG){   //寄存器
                    if(strcmp(tokens[q].str, "$eax") == 0){
                        return -cpu.eax;
                    }
                    else if(strcmp(tokens[q].str, "$ecx") == 0){
                        return -cpu.ecx;
                    }
                    else if(strcmp(tokens[q].str, "$edx") == 0){
                        return -cpu.edx;
                    }
                    else if(strcmp(tokens[q].str, "$ebx") == 0){
                        return -cpu.ebx;
                    }
                    else if(strcmp(tokens[q].str, "$esp") == 0){
                        return -cpu.esp;
                    }
                    else if(strcmp(tokens[q].str, "$ebp") == 0){
                        return -cpu.ebp;
                    }
                    else if(strcmp(tokens[q].str, "$esi") == 0){
                        return -cpu.esi;
                    }
                    else if(strcmp(tokens[q].str, "$edi") == 0){
                        return -cpu.edi;
                    }
                    else if(strcmp(tokens[q].str, "$eip") == 0){
                        return -cpu.eip;
                    }
                    else{
                        //printf("%s\n", tokens[q].str);
                        //printf("here\n");
                        assert(0);
                    }
                }
                else if(tokens[q].type == 1){   //十进制
                    sscanf(tokens[q].str, "%d", &num);
                    return -num;
                }
                else if(tokens[q].type == 2){   //十六进制
                    sscanf(tokens[q].str, "%x", &num);
                    return -num;
                }
            }
            //……
        }
        //……
    }
}
```

测试结果：正确输出



```
For help, type "help"
(nemu) p -2 * 3 - 1
[src/monitor/debug/expr.c,98,make_token] match rules[3] = "\-" at position 0 with len 1: -
[src/monitor/debug/expr.c,98,make_token] match rules[7] = "[0-9]+" at position 1 with len 1:
2
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[4] = "\*" at position 3 with len 1: *
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[7] = "[0-9]+" at position 5 with len 1:
3
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 6 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[3] = "\-" at position 7 with len 1: -
[src/monitor/debug/expr.c,98,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,98,make_token] match rules[7] = "[0-9]+" at position 9 with len 1:
1
-7
(nemu) p 2*(-1)
[src/monitor/debug/expr.c,98,make_token] match rules[7] = "[0-9]+" at position 0 with len 1:
2
[src/monitor/debug/expr.c,98,make_token] match rules[4] = "\*" at position 1 with len 1: *
[src/monitor/debug/expr.c,98,make_token] match rules[9] = "\(" at position 2 with len 1: (
[src/monitor/debug/expr.c,98,make_token] match rules[3] = "\-" at position 3 with len 1: -
[src/monitor/debug/expr.c,98,make_token] match rules[7] = "[0-9]+" at position 4 with len 1:
1
[src/monitor/debug/expr.c,98,make_token] match rules[10] = "\)" at position 5 with len 1: )
-2
(nemu)
```

9. 实现x命令使用表达式求值

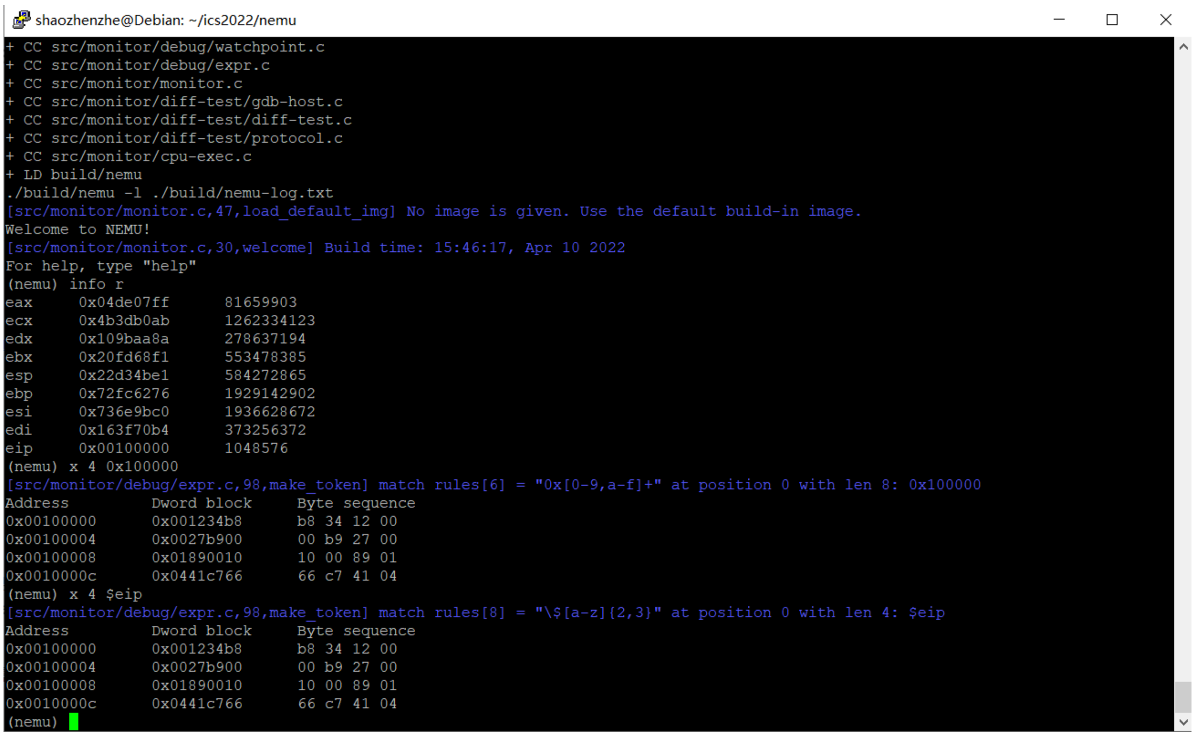
前面的`expr`函数已经可以求值，只需把读取的x命令参数传入`expr`调用即可

`/nemu/src/monitor/debug/ui.c`

```c
static int cmd_x(char *args){
    char *arg = strtok(NULL, " ");
    char *arg_1 = strtok(NULL, " ");
    int count;
    vaddr_t address;
    sscanf(arg, "%d", &count);
    //sscanf(arg_1, "%x", &address);

    bool* success = false;      //变化在这里
    address = expr(arg_1, success);

    printf("Address\t\tDword block\tByte sequence\n");
    for(int i=0; i<count;i++){
        printf("0x%08x\t0x%08x\t", address, vaddr_read(address, 4));
        for(int j=0;j<4;j++) {
            printf("%02x ", vaddr_read(address+j, 1));/*read 1 byte once*/
        }
        printf("\n");
        address += 4; /*address add 4 bytes to the next Dword block*/
    }
    return 0;
}
```

测试用例：`x 4 $eip`、`x 4 0x100000` 两者结果应该相同

测试结果：正确输出


shaozhenzhe@Debian: ~/ics2022/nemu

```
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/monitor.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/cpu-exec.c
+ LD build/nemu
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 15:46:17, Apr 10 2022
For help, type "help"
(nemu) info r
eax     0x04de07ff      81659903
ecx     0x4b3db0ab      1262334123
edx     0x109baa8a      278637194
ebx     0x20fd68f1      553478385
esp     0x22d34be1      584272865
ebp     0x72fc6276      1929142902
esi     0x736e9bc0      1936628672
edi     0x163f70b4      373256372
eip     0x00100000      1048576
(nemu) x 4 0x100000
[src/monitor/debug/expr.c,98,make_token] match rules[6] = "0x[0-9,a-f]+" at position 0 with len 8: 0x100000
Address         Dword block     Byte sequence
0x00100000      0x001234b8      b8 34 12 00
0x00100004      0x0027b900      00 b9 27 00
0x00100008      0x01890010      10 00 89 01
0x0010000c      0x0441c766      66 c7 41 04
(nemu) x 4 $eip
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 0 with len 4: $eip
Address         Dword block     Byte sequence
0x00100000      0x001234b8      b8 34 12 00
0x00100004      0x0027b900      00 b9 27 00
0x00100008      0x01890010      10 00 89 01
0x0010000c      0x0441c766      66 c7 41 04
(nemu) 
```

10. 监视点结构体

`char expr[32]`参考了token结构体的`char str[32]`，`new_val`和`old_val`的类型都是`uint32_t`

`/nemu/include/monitor/watchpoint.h`

```c
typedef struct watchpoint {
  int NO;
  struct watchpoint *next;

  /* TODO: Add more members if necessary */
  char expr[32];
  uint32_t new_val;
  uint32_t old_val;

} WP;
```

11. 监视点池的管理

首先到`watchpoint.h`声明这两个函数

`new_wp`有两种情况，一种是head链表为空时，直接`head = p`，另一种head不为空，需要查找head最后一个节点，把free_链表的第一个节点插上，表达式用`strcpy`赋值，`old_val`用`expr`函数赋值

`/nemu/src/monitor/debug/watchpoint.c`

```c
WP* new_wp(char *args){
    WP* p = free_;
    free_ = free_->next;
    if(free_ == NULL){
        assert(0);
    }
    p->next = NULL;

    strcpy(p->expr, args);
    bool *success = false;
    p->old_val = expr(args, success);

    if(head == NULL){
        head = p;
        WP_NO = 0;
        p->NO = WP_NO;
    }
    else{
        WP_NO++;
        WP* q = head;
        while(q->next != NULL){
            q = q->next;
        }
        q->next = p;
        p->NO = WP_NO;
    }
    return p;
}
```

`free_wp()`也分两种情况，一种是`wp == head`，直接把head后移一位即可，另一种是需要在head中找到wp，把wp内容清空，插到free_开头，head中需要跳过wp保持链表连接

```
void free_wp(WP *wp){
    if(wp == NULL){
        assert(0);
    }
    if(wp == head){
        head = head->next;
    }
    else{
        WP* p = head;
        while(p->next != wp){
            p = p->next;
        }
        p->next = wp->next;
    }
    wp->next = free_;
    free_ = wp;
    wp->new_val = 0;
    wp->expr[0] = '\0';
}
```

12. 监视点加入调试器

`/nemu/src/monitor/debug/ui.c`

`cmd_w()`调用`new_wp()`函数来存储新的监视点

```
static int cmd_w(char* args){
    char *arg = strtok(NULL, " ");
    WP* p = new_wp(arg);
    printf("Set watchpoint #%d\n", p->NO);
    printf("expr\t= %s\n", p->expr);
    printf("old value = %d\n", p->old_val);
    return 0;
}
```

`cmd_()`需要调用第13题的`delete_watchpoint()`函数

```
static int cmd_d(){
    char *arg = strtok(NULL, " ");
    int num;
    sscanf(arg, "%d", &num);
    delete_watchpoint(num);
    return 0;
}
```

`info w`命令需要调用第13题的`list_watchpoint()`函数

```
static int cmd_info(char *args) {
    char *arg = strtok(NULL, " ");
    if(strcmp(arg, "r") == 0) {
        for(int i=0;i<8;i++) {
            printf("%s\t0x%08x\t%d\t\n", regsl[i], cpu.gpr[i]._32,
cpu.gpr[i]._32);
        }   /*print the register informations*/
        printf("eip\t0x%08x\t%d\t\n", cpu.eip, cpu.eip);
    }
    else if (strcmp(arg, "w") == 0) {   //info w
        //printf("Waiting to add!!\n"); /*waiting to add*/
        list_watchpoint();
    }
    return 0;
}
```

13. 监视点主要功能

`/nemu/src/monitor/debug/watchpoint.c`

这些函数都要到`watchpoint.h`里先声明

`set_watchpoint()`函数调用`new_wp()`即可

```
int set_watchpoint(char *e){
    WP* p = new_wp(e);
    return p->NO;
    return 0;
}
```

`delete_watchpoint()`调用`free_wp()`即可

```
bool delete_watchpoint(int NO){
    WP* p = head;
    while(p!=NULL && p->NO!=NO){
        p = p->next;
    }
    if(p==NULL){
        printf("Not found\n");
        return false;
    }
    else{
        printf("Watchpoint %d deleted\n", p->NO);
        free_wp(p);
        return true;
    }
}
```

`list_watchpoint()` 遍历链表输出即可

```
void list_watchpoint(){
    WP* p = head;
    printf("NO\tExpr\tOld Value\t\n");
    while(p!=NULL){     //遍历输出
        printf("%d\t%s\t%d\n", p->NO, p->expr, p->old_val);
        p = p->next;
    }
}
```

`scan_watchpoint()` 遍历链表，如果新旧值不同，就返回节点

```
WP* scan_watchpoint(){
    WP* p = head;
    int num;
    bool* success=false;

    while(p!=NULL){
        num = expr(p->expr, success);
        if(num != p->old_val){  //新旧值不同

            p->new_val = num;
            return p;c
        }
        p = p->next;
    }
    return p;
}
```

要完成 每当 `cpu_exec()` 执行完一条指令，就对所有待监视的表达式进行求值的操作，需要在 `cpu-exec.c` 写入以下代码

调用 `scan_watchpoint()`，如果返回的不是空指针，说明有变化，把 `nemu_state` 设为 `NEMU_STOP`，并输出相应信息，并更新数据

记得要把头文件包含进来

`/nemu/src/monitor/cpu-exec.c`

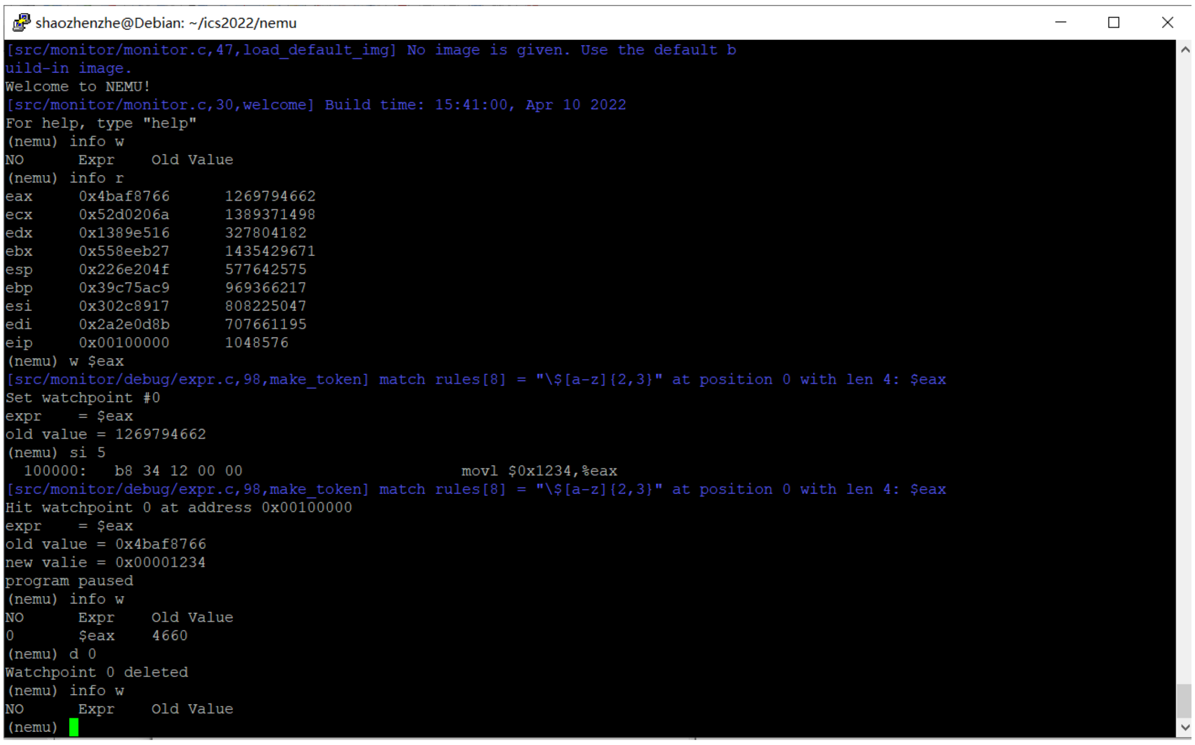
```
#include "monitor/watchpoint.h"

```

```c
#ifdef DEBUG
    /* TODO: check watchpoints here. */

    WP* p = scan_watchpoint();
    if(p!=NULL){
        nemu_state = NEMU_STOP;

        printf("Hit watchpoint %d at address 0x%08x\n", p->NO,
old_eip);//old_eip在cpu_exec()执行前保存
        printf("expr\t= %s\n", p->expr);//输出信息
        printf("old value = 0x%08x\n", p->old_val);
        printf("new valie = 0x%08x\n", p->new_val);
        printf("program paused\n");
        p->old_val = p->new_val;
    }

#endif
```

测试用例：

> w $eax　//添加监视点
>
> si 5　　//执行，查看是否会触发监视点
>
> info w　//打印监视点信息
>
> d 0　　//删除对应监视点

测试结果：正确输出



```
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 15:41:00, Apr 10 2022
For help, type "help"
(nemu) info w
NO      Expr    Old Value
(nemu) info r
eax     0x4baf8766      1269794662
ecx     0x52d0206a      1389371498
edx     0x1389e516      327804182
ebx     0x558eeb27      1435429671
esp     0x226e204f      577642575
ebp     0x39c75ac9      969366217
esi     0x302c8917      808225047
edi     0x2a2e0d8b      707661195
eip     0x00100000      1048576
(nemu) w $eax
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 0 with len 4: $eax
Set watchpoint #0
expr    = $eax
old value = 1269794662
(nemu) si 5
  100000:   b8 34 12 00 00                        movl $0x1234,%eax
[src/monitor/debug/expr.c,98,make_token] match rules[8] = "\$[a-z]{2,3}" at position 0 with len 4: $eax
Hit watchpoint 0 at address 0x00100000
expr    = $eax
old value = 0x4baf8766
new valie = 0x00001234
program paused
(nemu) info w
NO      Expr    Old Value
0       $eax    4660
(nemu) d 0
Watchpoint 0 deleted
(nemu) info w
NO      Expr    Old Value
(nemu) 
```

14. 实现软件断点

未完成

遇到的问题及解决办法

1.遇到问题：测试表达式求值时，执行完 `p *0x100000` 后 执行 `p *$eip` 报错

解决方案：尝试输出中间值，发现`str`里存的是 `$eip0000`，说明没有清空token，只要在每次记录token前清空`str`即可

2.遇到问题：测试表达式求值时，执行 `p $eip == 0x100000` 报错

解决方案：分析读取token时输出的Log，发现`$eip`和 `==` 都读取正确，但是`0x100000`只读取了0，分析后发现在`rules[]`里，hex的匹配规则要在十进制匹配规则前面，否则对于`0x`开头的十六进制，会先匹配十进制，即只匹配到0

3.遇到问题：完成监视点主要功能时，编译`cpu-exec.c`时报错 `implicit declaration of function 'scan_watchpoint'`

解决方案：上网搜索查看，一般是相关的头文件没有声明这个函数。仔细查看后，发现`watchpoint.h`没有声明这个函数，同时也发现`cpu-exec.c`没有包含`watchpoint.h`，声明且包含后解决。

实验心得

重新复习了正则匹配规则，对于输入的内容进行正则匹配，得到想要的数据。熟悉了词法分析，其中把表达式分割成token保存求值的思想很奇妙。了解了表达式递归求值的过程，学习了程序是如何判断符号，判断运算优先级从而进行运算。对于平时经常使用的eval函数有了更深的了解。对于gdb的监视点、断点有了更深的理解，完成了简单的设置监视点、触发监视点、删除监视点等操作。阅读理解大规模代码的能力提升。本次PA1.2&1.3收获很大。

其他备注

无