南京航空航天大学《计算机组成原理Ⅱ课程设计》报告

- 姓名：马睿
- 班级：1619304
- 学号：161930131
- 报告阶段：PA1.2 & 1.3
- 完成日期：2021.4.6
- 本次实验，我完成了所有内容。

目录

思考题

一、有什么办法?

利用中缀表达式进行求值，最主要是用了两个栈：一个栈用来保存需要计算的数据（操作数栈），一个用来保存计算优先符（运算符栈）。

二、一些简单的正则表达式

- 以 `0x` 开头的 `32` 位十六进制整数

  `0x[a-fA-F0-9]{1,8}`

- 英文字母和数字组成的字符串

  `[a-fA-Z0-9]+`

- C 语言中的变量名或函数名（也就是符合变量的命名规则）

  `[a-fA-Z_]+[a-fA-F0-9_]*`

- 学号 - 姓名 - PA1.1.pdf，如 `161722222 - 张三 - PA1.1.pdf`。（提示：`[\u4e00-\u9fa5]` 将匹配一个汉字）

  `[0-9]{9} - [\u4e00-\u9fa5]{1，5} - PA1.1.pdf`

三、这是为什么?

因为 `c` 语言的字符串里 `\` 也是转义字符，想要用 `c` 语言表示正则表达式中的转义字符 `\`，需要用 `\\` 来表示。

四、如何处理以上的问题?

在将输入表达式字串存储在字符串中之前，判断长度是否大于 `31`，如果大于则显示错误信息，然后终止程序。

五、递归求值的过程?

如果表达式中存在子表达式，那么先对子表达式进行递归操作再运算，直到表达式为一个数时终止递归，并返回值表达式的值。

例如：

```
4+3*(2+1)
```

```
--> 4   +   3*(2+1)
    expr + expr
--> 4   +   3   *   (2+1)
    expr + expr * expr
--> 4   +   3   *   (2   +  1)
    expr + expr * (expr + expr)
--> 4   +   3   *   (2   +1)
    expr + expr * (expr + number)
--> 4   +   3   *   (2+1)
    expr + expr * (number + number)
--> 4   +   3   *3
    expr + expr * number
--> 4   +   3*3
    expr + number * number
--> 4   +9
    expr + number
--> 4+9
    number + number
--> 13
    number
```

六、体验监视点

```
watch [-l|-location] expr [thread-id] [maskvalue]
为一个表达式设置一个监视点。当表达式expr被程序写入且其值发生变化时，GDB将中断。

rwatch [-l|-location] expr [thread-id] [maskvalue] 。
设置一个监视点，当程序读取 expr 的值时，该监视点将中断。

awatch [-l|-location] expr [thread-id] [maskvalue] 。
设置一个监视点，当程序读取或写入expr时，该监视点将被中断。
```

注意事项:

- 如果命令中包含`[thread-id]`参数，那么只有当标识的线程改变了`expr`的值时，`GDB`才会中断。如果任何其他线程改变了 `expr` 的值，`GDB`将不会中断。
- 参数`-location`告诉`GDB`监视由`expr`引用的内存。在这种情况下，`GDB`将计算`expr`，获取结果的地址，并监视该地址处的内存。结果的类型用来确定观察内存的大小。如果表达式的结果没有地址，那么`GDB`将打印一个错误。
- `[mask maskvalue]`参数允许创建掩码监视点。掩码指定在将下位机访问的地址与监视点地址进行匹配时，应忽略地址的某些位（掩码中被重置的位）。因此，一个带掩码的监视点会同时监视多个地址--那些未被掩码的位与监视点地址中未被掩码的位相同的地址。
- 如果你要监视一个以数字形式输入的地址的变化，你需要对它进行反引用，因为地址本身只是一个永远不会改变的常数。`GDB`拒绝创建一个监视一个永不变化的值的看点。

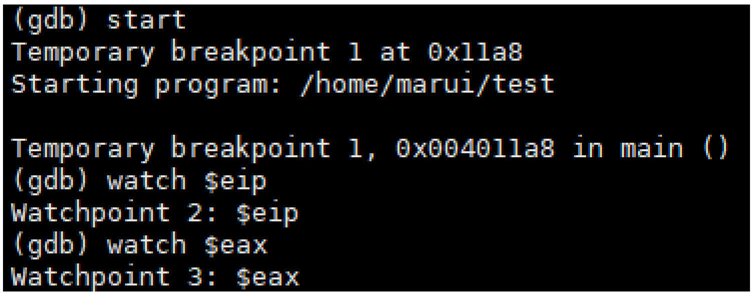
  ```
  (gdb) watch 0x600850
  不能观察常量值 0x600850

  (gdb) watch *(int *) 0x600850
  Watchpoint 1: *(int *) 6293584
  ```

- 当监视本地（自动）变量或涉及此类变量的表达式超出范围时，也就是当执行离开定义这些变量的块时，GDB会自动删除监视点。特别是，当被调试的程序终止时，所有的局部变量都会离开范围，因此只有监视全局变量的监视点仍然被设置。如果你重新运行程序，需要重新设置所有这样的观察点。一种方法是在主函数的入口处设置一个代码断点，当主函数断点时，设置所有的监视点。

```
                              生成可执行文件
touch test.c
vim test.c
gcc -g test.c -o test   编译

                              gdb常用命令

gdb test 用gdb执行test
r or run：执行程序
c or continue：继续运行程序
start：单步执行，运行程序，停在第一执行语句

1、默认情况下，run 指令会一直执行程序，直到执行结束。如果程序中手动设置有断点，则 run 指令会执行程序至第一个断点处；
2、start 指令会执行程序至 main() 主函数的起始位置，即在 main() 函数的第一行语句处停止执行（该行代码尚未执行）。
可以这样理解，使用 start 指令启动程序，完全等价于先在 main() 主函数起始位置设置一个断点，然后再使用 run 指令启动程序。另外，程序执行过程中使用 run 或者 start 指令，表示的是重新启动程序。
```

1. 使用适当的 GDB 命令新建两个监视点：

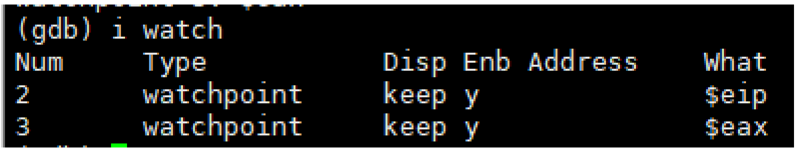
`watch $eip`，`watch $eax`



```
(gdb) start
Temporary breakpoint 1 at 0x11a8
Starting program: /home/marui/test

Temporary breakpoint 1, 0x004011a8 in main ()
(gdb) watch $eip
Watchpoint 2: $eip
(gdb) watch $eax
Watchpoint 3: $eax
```

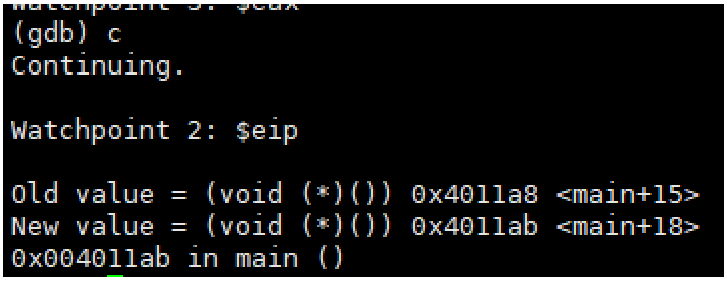
2. 使用GDB 命令显示当前所有监视点的列表：

`info watchpoints 或 i watchpoints 或 i watch`



```
(gdb) i watch
Num     Type           Disp Enb Address    What
2       watchpoint     keep y              $eip
3       watchpoint     keep y              $eax
```

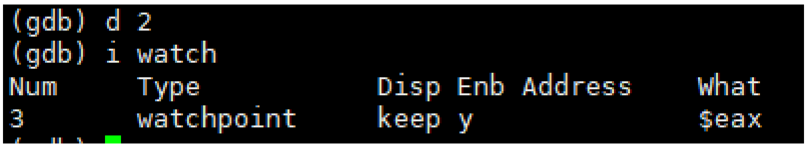
3. 运行程序，使程序命中监视点至少一次



```
Watchpoint 3: $eax
(gdb) c
Continuing.

Watchpoint 2: $eip

Old value = (void (*)()) 0x4011a8 <main+15>
New value = (void (*)()) 0x4011ab <main+18>
0x004011ab in main ()
```

4. 使用 `GDB` 命令删除任意一个之前设置的监视点：

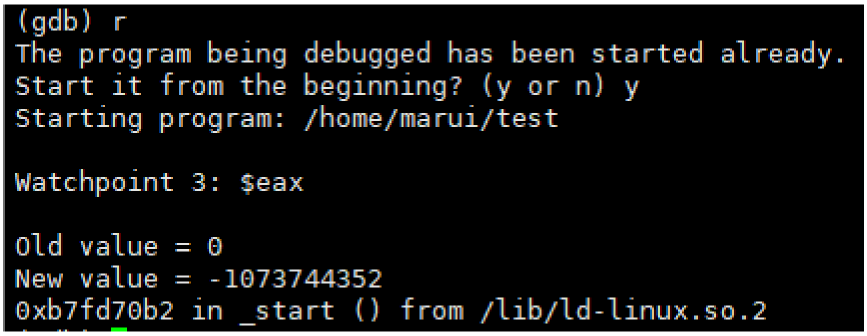
`d 2 或 delete 2`，其中2是指监视点的 `Num`（见上图）



```
(gdb) d 2
(gdb) i watch
Num     Type           Disp Enb Address    What
3       watchpoint     keep y              $eax
```

5. 不退出 `GDB`，重新运行程序，使程序不能在被删除的监视点上命中。

`run 或 r 或 start`



```
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/marui/test

Watchpoint 3: $eax

Old value = 0
New value = -1073744352
0xb7fd70b2 in _start () from /lib/ld-linux.so.2
```

七、科学起名

`stdlib.h` 中定义了 `free` 函数，使用 `free` 来作为头指针的名字。

八、温故而知新

此处 `static` 的作用是声明作用域，静态全局变量只能在定义该变量的源文件内有效，其它源文件中不能使用它。

九、 一点也不能长?

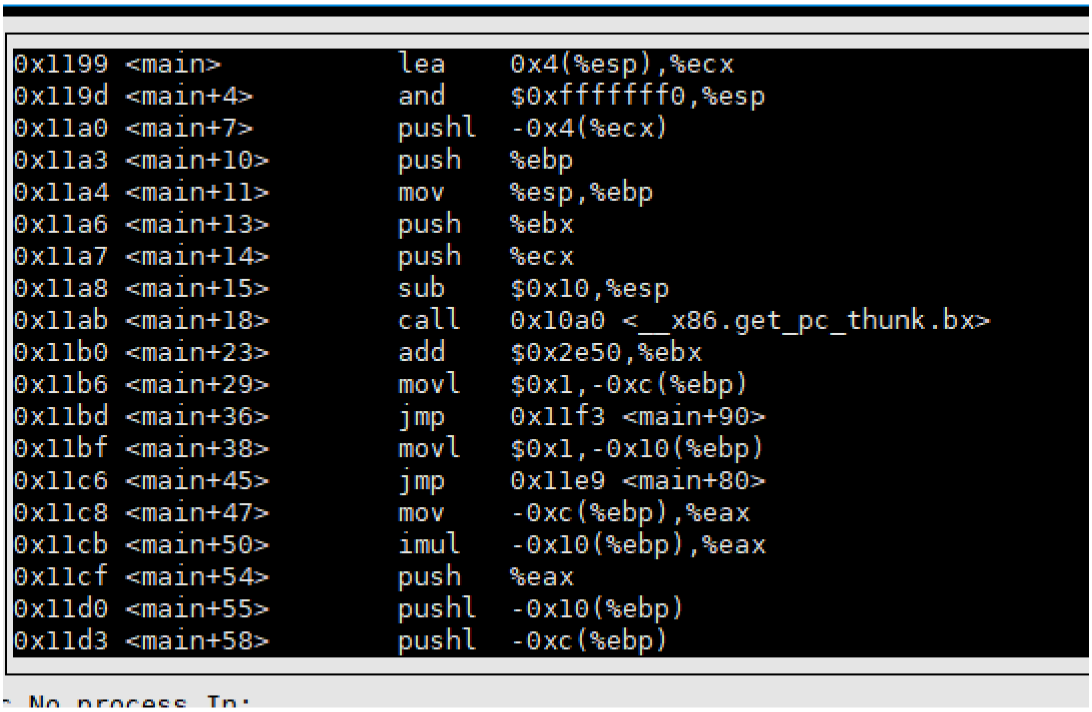
指令的长度是 1 个字节是必须的。

如果替换断点指令的长度超过1个字节，可能会被迫覆盖下一条指令的一部分，这将使该指令乱码，并可能产生完全无效的指令。

十、“随心所欲”的断点

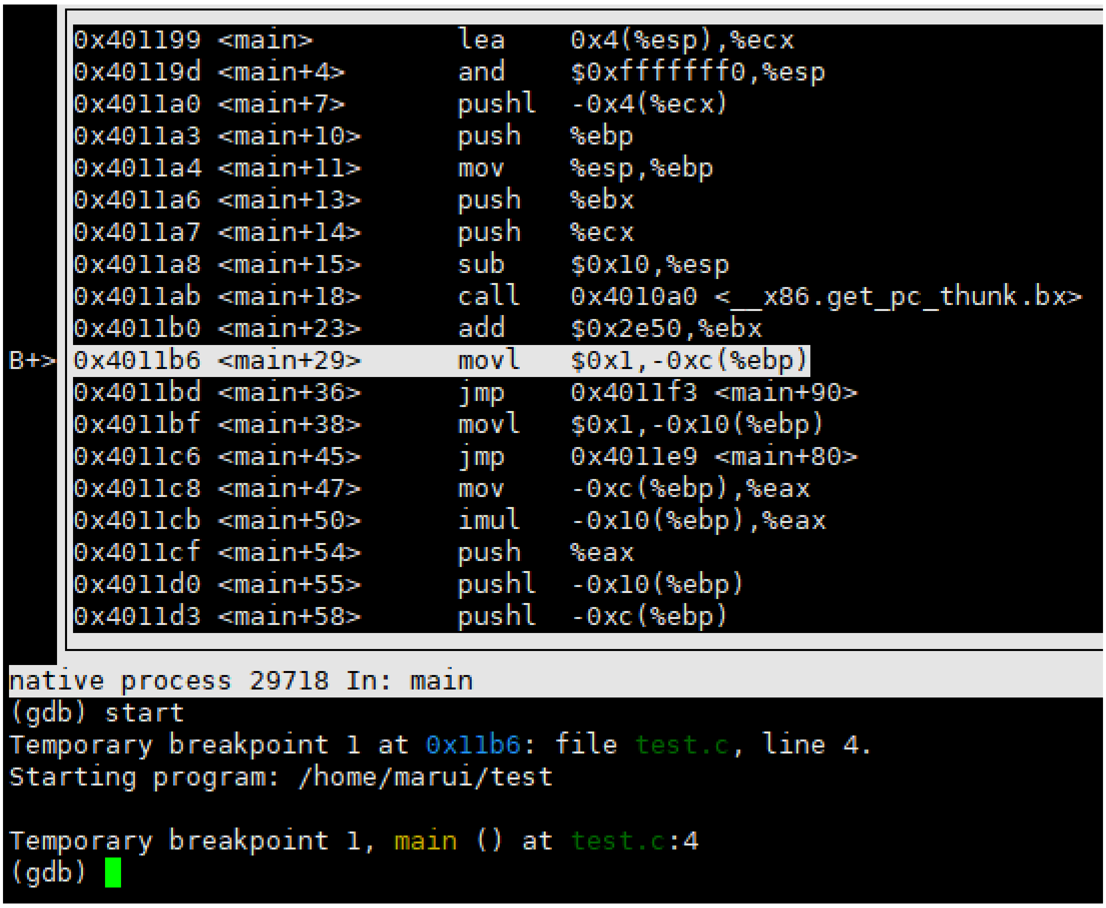
如果把断点设置在指令的非首字节，`gdb`会停止进程并且意外退出。

1. 先生成可执行文件`gcc -g test.c -o test`
2. 用`gdb`执行调试可执行文件：`gdb test`
3. 查看指令地址`layout asm`



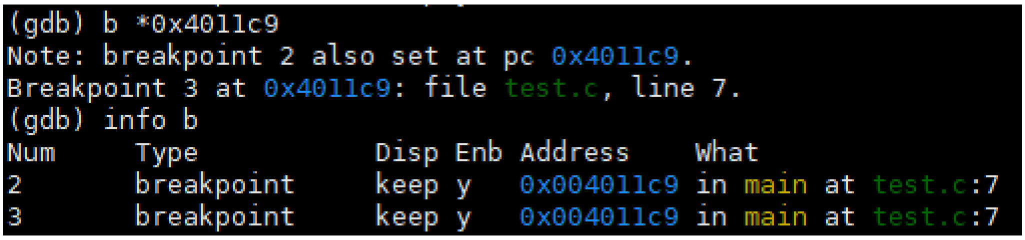
```
0x1199 <main>           lea    0x4(%esp),%ecx
0x119d <main+4>         and    $0xfffffff0,%esp
0x11a0 <main+7>         pushl  -0x4(%ecx)
0x11a3 <main+10>        push   %ebp
0x11a4 <main+11>        mov    %esp,%ebp
0x11a6 <main+13>        push   %ebx
0x11a7 <main+14>        push   %ecx
0x11a8 <main+15>        sub    $0x10,%esp
0x11ab <main+18>        call   0x10a0 <__x86.get_pc_thunk.bx>
0x11b0 <main+23>        add    $0x2e50,%ebx
0x11b6 <main+29>        movl   $0x1,-0xc(%ebp)
0x11bd <main+36>        jmp    0x11f3 <main+90>
0x11bf <main+38>        movl   $0x1,-0x10(%ebp)
0x11c6 <main+45>        jmp    0x11e9 <main+80>
0x11c8 <main+47>        mov    -0xc(%ebp),%eax
0x11cb <main+50>        imul   -0x10(%ebp),%eax
0x11cf <main+54>        push   %eax
0x11d0 <main+55>        pushl  -0x10(%ebp)
0x11d3 <main+58>        pushl  -0xc(%ebp)

- No process In:
```

4. 开启程序：`start`



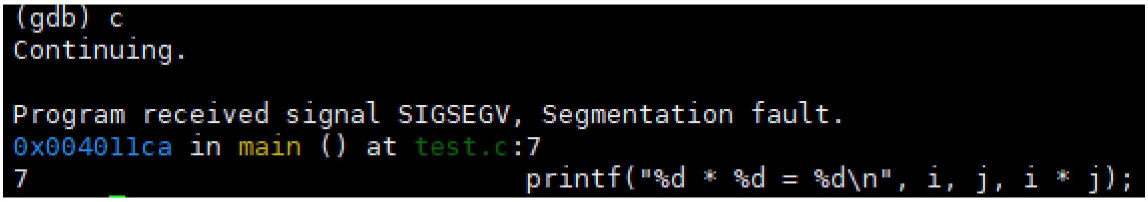
```
     0x401199 <main>           lea    0x4(%esp),%ecx
     0x40119d <main+4>         and    $0xfffffff0,%esp
     0x4011a0 <main+7>         pushl  -0x4(%ecx)
     0x4011a3 <main+10>        push   %ebp
     0x4011a4 <main+11>        mov    %esp,%ebp
     0x4011a6 <main+13>        push   %ebx
     0x4011a7 <main+14>        push   %ecx
     0x4011a8 <main+15>        sub    $0x10,%esp
     0x4011ab <main+18>        call   0x4010a0 <__x86.get_pc_thunk.bx>
     0x4011b0 <main+23>        add    $0x2e50,%ebx
B+>  0x4011b6 <main+29>        movl   $0x1,-0xc(%ebp)
     0x4011bd <main+36>        jmp    0x4011f3 <main+90>
     0x4011bf <main+38>        movl   $0x1,-0x10(%ebp)
     0x4011c6 <main+45>        jmp    0x4011e9 <main+80>
     0x4011c8 <main+47>        mov    -0xc(%ebp),%eax
     0x4011cb <main+50>        imul   -0x10(%ebp),%eax
     0x4011cf <main+54>        push   %eax
     0x4011d0 <main+55>        pushl  -0x10(%ebp)
     0x4011d3 <main+58>        pushl  -0xc(%ebp)

native process 29718 In: main
(gdb) start
Temporary breakpoint 1 at 0x11b6: file test.c, line 4.
Starting program: /home/marui/test

Temporary breakpoint 1, main () at test.c:4
(gdb) 
```

5. 设置断点：`b *0x4011c9`



```
(gdb) b *0x4011c9
Note: breakpoint 2 also set at pc 0x4011c9.
Breakpoint 3 at 0x4011c9: file test.c, line 7.
(gdb) info b
Num     Type           Disp Enb Address            What
2       breakpoint     keep y   0x004011c9 in main at test.c:7
3       breakpoint     keep y   0x004011c9 in main at test.c:7
```

6. 运行程序：`c`



```
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x004011ca in main () at test.c:7
7                               printf("%d * %d = %d\n", i, j, i * j);
```

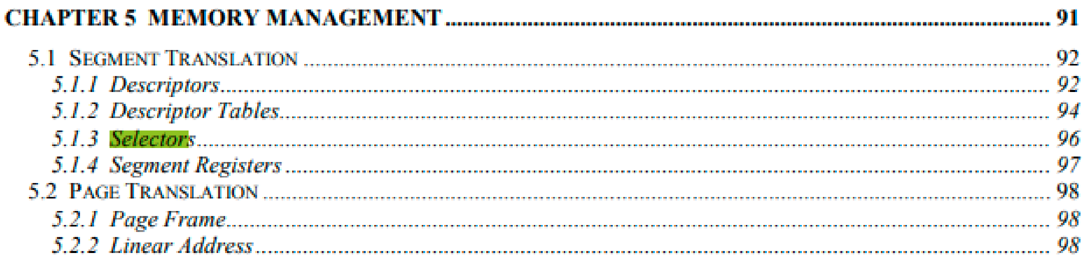
原因分析：断点如果设置在非首字节，那么在指令的首字节就不会检测到断点，就会继续执行。但是原指令发生了变化，导致一个新的指令，该指令的具体操作可能产生异常，所以退出。

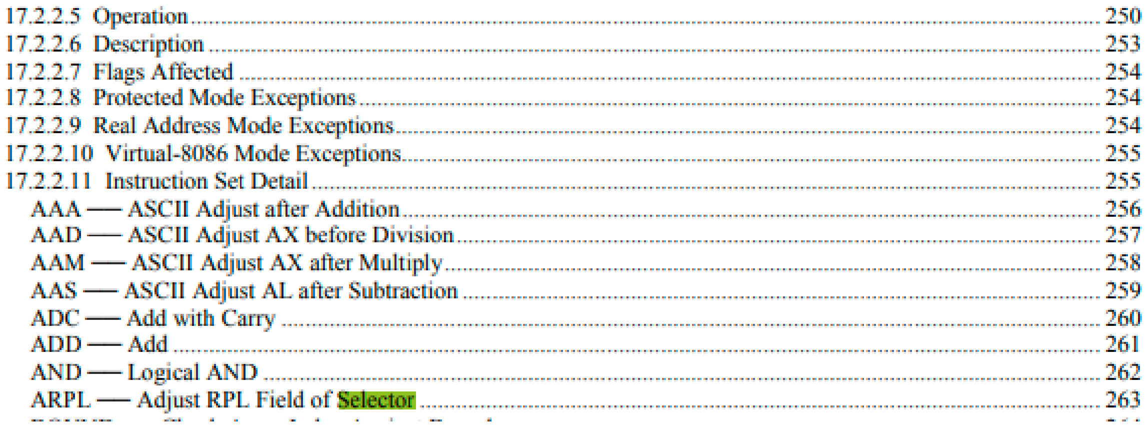
十一、NEMU的前世今生

模拟器是用于模拟一个系统内部并实现其功能的软件，而调试器是一种用于调试其它程序的计算机程序及工具。

`gdb`主要功能的实现基于系统函数`ptrace`，该函数可以让父进程观察和控制其子进程的检查、执行。

十二、尝试通过目录定位关注的问题





十三、理解基础设施

`GDB`： 500 * 0.9 * 20 * 30 = 4500min = 75h

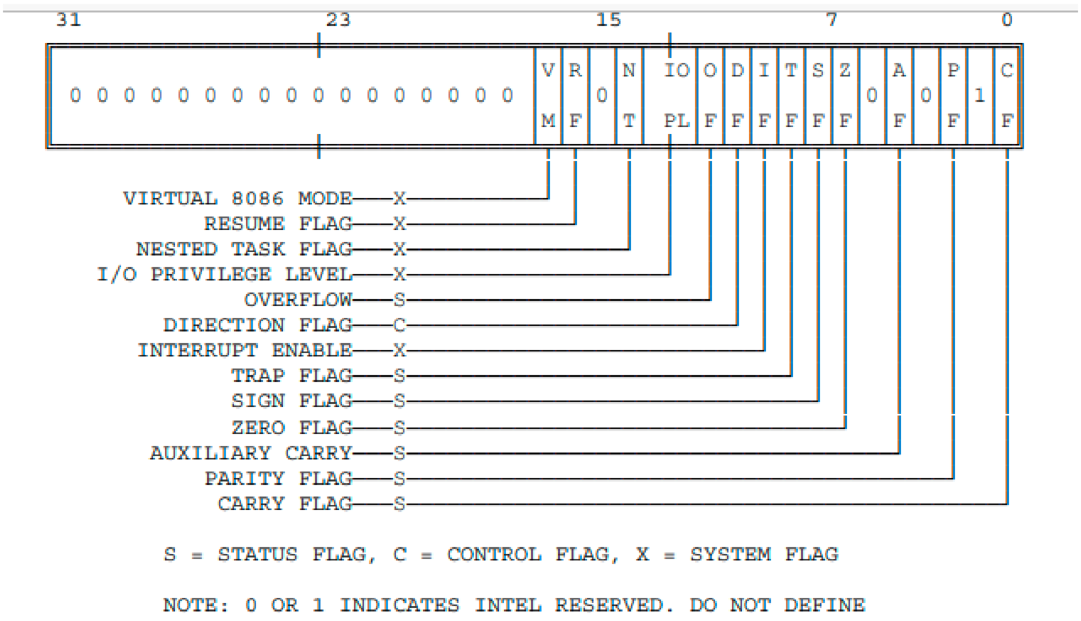
简易调试器：75 / 3 = 25h

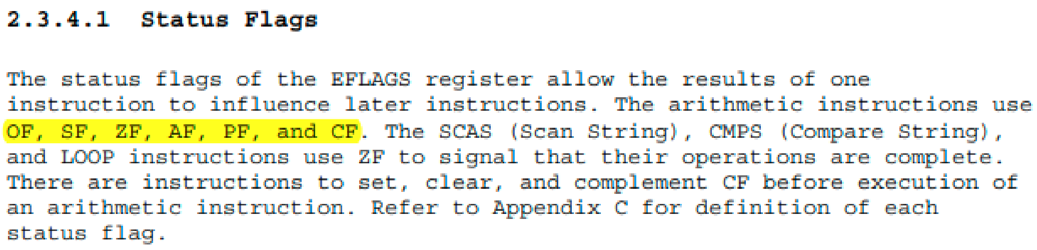
节省了50小时

十四、查阅i386手册

1. `EFLAGS`寄存器中的CF位是什么意思：进位标志

   P33 ~ 34



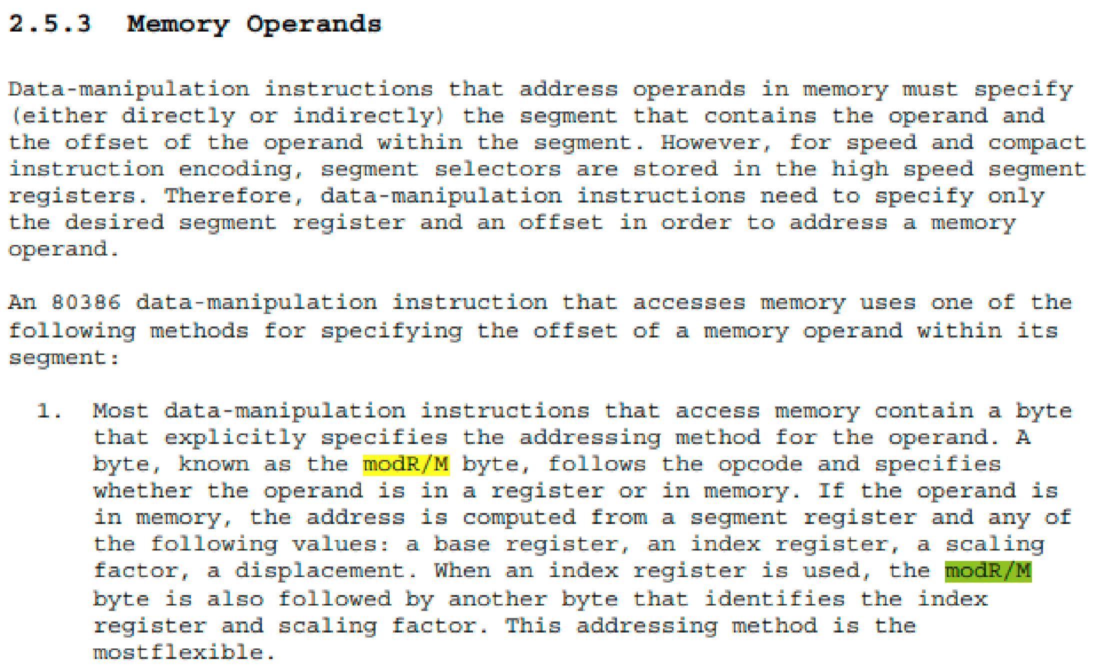


2.3.4.1 Status Flags

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.

2. `ModR/M`字节是什么：里面包含操作码并指定操作数是在寄存器中还是在内存中。

P38 ~ 39



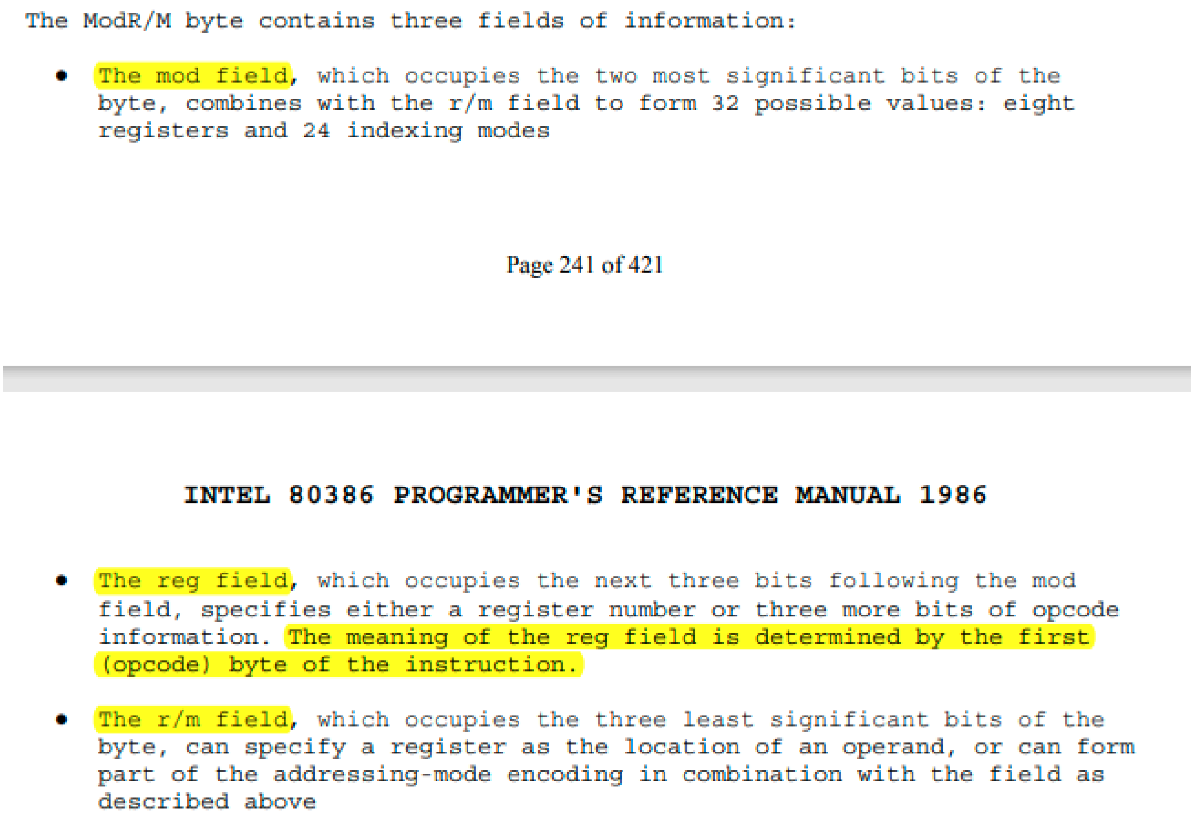
2.5.3 Memory Operands

Data-manipulation instructions that address operands in memory must specify (either directly or indirectly) the segment that contains the operand and the offset of the operand within the segment. However, for speed and compact instruction encoding, segment selectors are stored in the high speed segment registers. Therefore, data-manipulation instructions need to specify only the desired segment register and an offset in order to address a memory operand.

An 80386 data-manipulation instruction that accesses memory uses one of the following methods for specifying the offset of a memory operand within its segment:

1. Most data-manipulation instructions that access memory contain a byte that explicitly specifies the addressing method for the operand. A byte, known as the modR/M byte, follows the opcode and specifies whether the operand is in a register or in memory. If the operand is in memory, the address is computed from a segment register and any of the following values: a base register, an index register, a scaling factor, a displacement. When an index register is used, the modR/M byte is also followed by another byte that identifies the index register and scaling factor. This addressing method is the mostflexible.

P241 ~ 242



The ModR/M byte contains three fields of information:

- The mod field, which occupies the two most significant bits of the byte, combines with the r/m field to form 32 possible values: eight registers and 24 indexing modes

Page 241 of 421

INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986

- The reg field, which occupies the next three bits following the mod field, specifies either a register number or three more bits of opcode information. The meaning of the reg field is determined by the first (opcode) byte of the instruction.

- The r/m field, which occupies the three least significant bits of the byte, can specify a register as the location of an operand, or can form part of the addressing-mode encoding in combination with the field as described above

3. `mov` 指令的具体格式是怎么样的：P345 ~ P351

十五、shell 命令

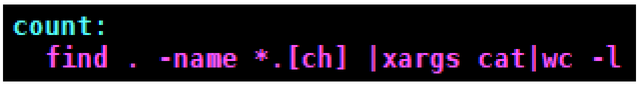
包含空行：

```
find . -name *.[ch] |xargs cat|wc -l
```

得到完成`PA1`后的总行数：4264

切换到`master`分支得到框架的总行数：3497

编辑`Makefile`文件：



```
count:
  find . -name *.[ch] |xargs cat|wc -l
```

去除空行：

```
find . -name *.[ch] |xargs cat|grep -v ^$|wc -l
```

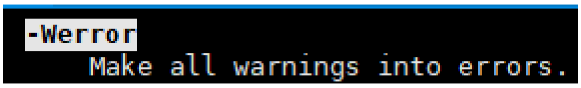
得到完成`PA1`后的总行数：3568

切换到`master`分支得到框架的总行数：2826

十六、使用`man`

```
CFLAGS   += -O2 -MMD -Wall -Werror -ggdb $(INCLUDES)
```

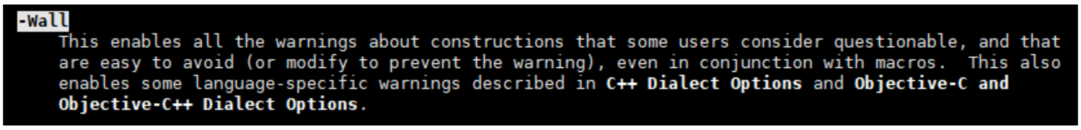
1. `man gcc`，再利用`/想要搜索的关键字`进行搜索，按`n`查看下一条，`N`查看上一条
2. `/-Werror`

   ```
   -Werror
       Make all warnings into errors.
   ```

   作用：要求`gcc`将所有的警告当成错误进行处理

   `/-Wall`

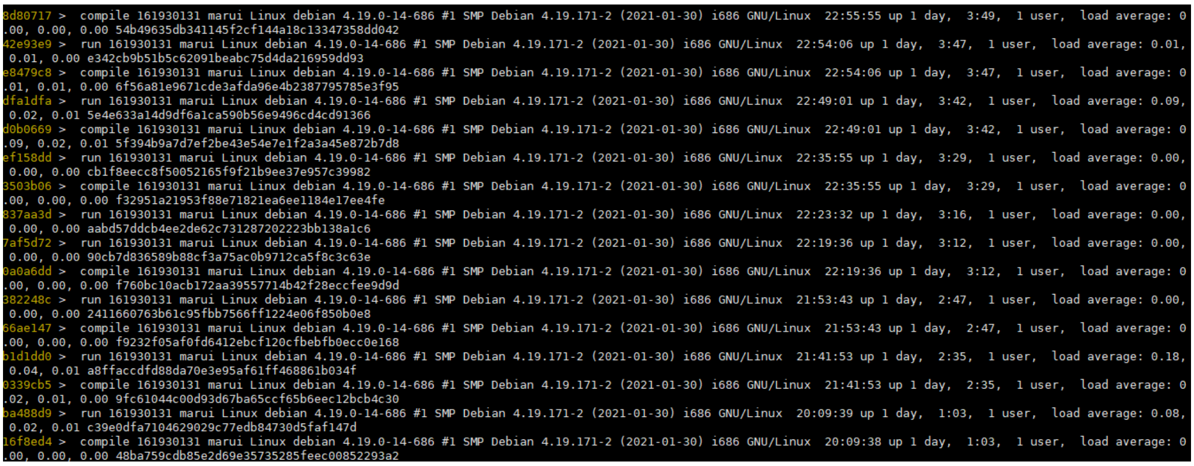
   ```
   -Wall
       This enables all the warnings about constructions that some users consider questionable, and that
       are easy to avoid (or modify to prevent the warning), even in conjunction with macros.  This also
       enables some language-specific warnings described in C++ Dialect Options and Objective-C and
       Objective-C++ Dialect Options.
   ```

   作用：打开`gcc`的所有警告。

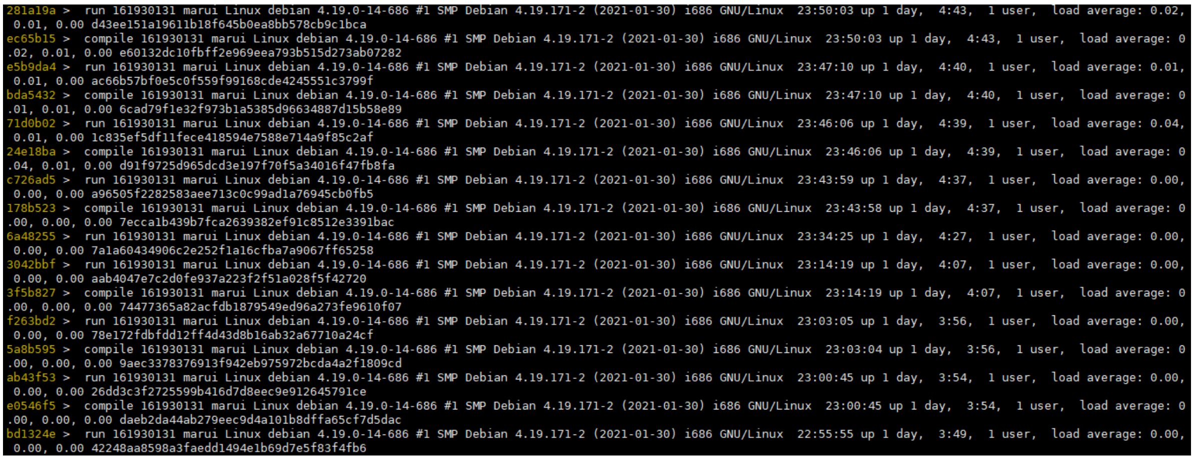
使用两者的目的：

1. 详细查错
2. 把警告直接当作错误处理，避免在之后会引起其他错误的出现。

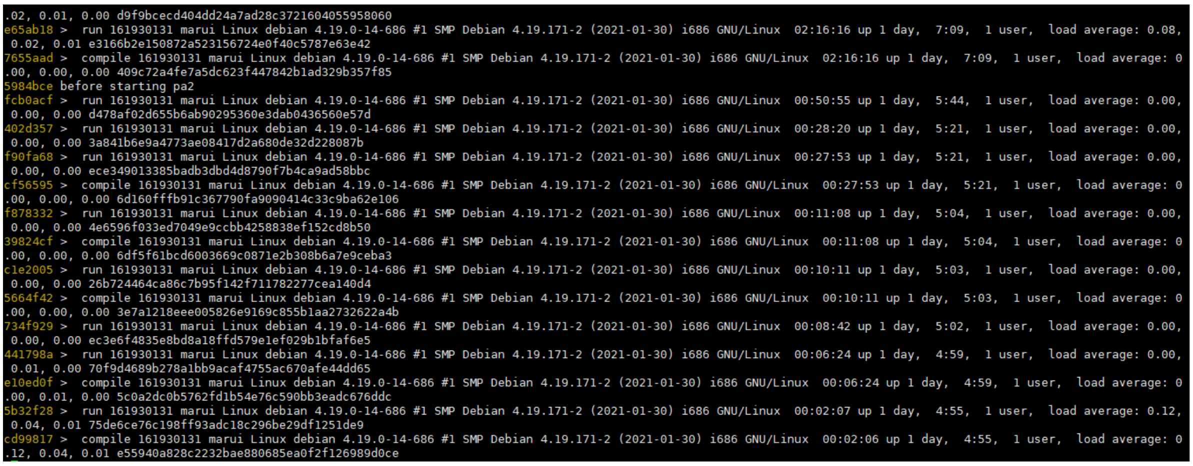
十七、`git log`和远程git仓库提交截图



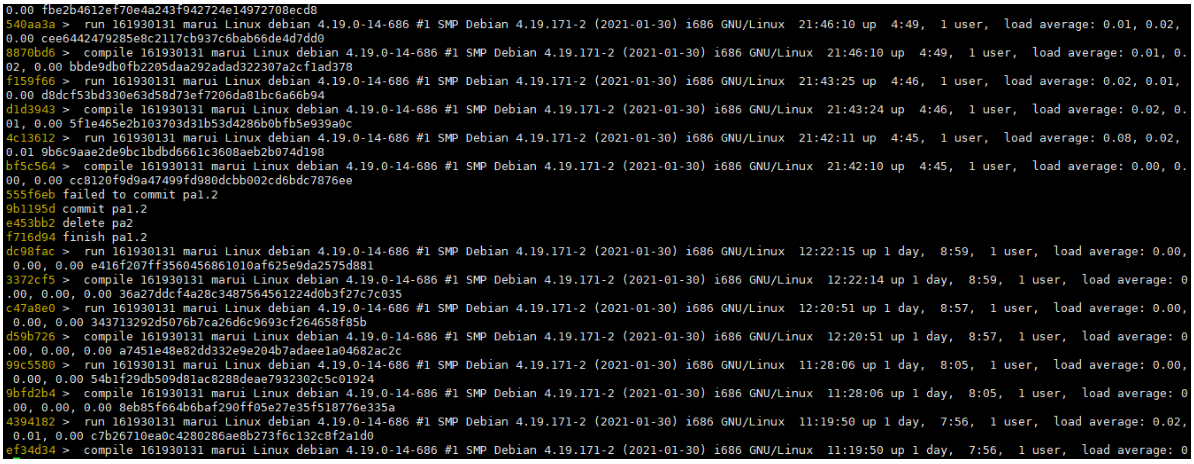
```
8d80717 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:55:55 up 1 day,  3:49,  1 user,  load average: 0
.00, 0.00, 0.00 54b49635db341145f2cf144a18c13347358dd042
42e93e9 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:54:06 up 1 day,  3:47,  1 user,  load average: 0.01,
 0.01, 0.00 e342cb9b51b5c62091beabc75d4da216959dd93
e8479c8 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:54:06 up 1 day,  3:47,  1 user,  load average: 0
.01, 0.01, 0.00 6f56a81e9671cde3afda96e4b2387795785e3f95
dfa1dfa >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:49:01 up 1 day,  3:42,  1 user,  load average: 0.09,
 0.02, 0.01 5e4e633a14d9df6a1ca590b56e9496cd4cd91366
d0b0669 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:49:01 up 1 day,  3:42,  1 user,  load average: 0
.09, 0.02, 0.01 5f394b9a7d7ef2be43e54e7e1f2a3a45e872b7d8
ef158dd >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:35:55 up 1 day,  3:29,  1 user,  load average: 0.00,
 0.00, 0.00 cb1f8eecc8f50052165f9f21b9ee37e957c39982
3503b06 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:35:55 up 1 day,  3:29,  1 user,  load average: 0
.00, 0.00, 0.00 f32951a21953f88e71821ea6ee1184e17ee4fe
837aa3d >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:23:32 up 1 day,  3:16,  1 user,  load average: 0.00,
 0.00, 0.00 aabd57ddcb4ee2de62c731287202223bb138a1c6
7af5d72 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:19:36 up 1 day,  3:12,  1 user,  load average: 0.00,
 0.00, 0.00 90cb7d836589b88cf3a75ac0b9712ca5f8c3c63e
0a0a6dd >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:19:36 up 1 day,  3:12,  1 user,  load average: 0
.00, 0.00, 0.00 f760bc10acb172aa39557714b42f28eccfee9d9d
382248c >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:53:43 up 1 day,  2:47,  1 user,  load average: 0.00,
 0.00, 0.00 2411660763b61c95fbb7566ff1224e06f850b0e8
66ae147 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:53:43 up 1 day,  2:47,  1 user,  load average: 0
.00, 0.00, 0.00 f9232f05af0fd6412ebcf120cfbebfb0ecc0e168
b1d1dd0 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:41:53 up 1 day,  2:35,  1 user,  load average: 0.18,
 0.04, 0.01 a8ffaccdfd88da70e3e95af61ff468861b034f
0339cb5 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:41:53 up 1 day,  2:35,  1 user,  load average: 0
.02, 0.01, 0.00 9fc61044c00d93d67ba65ccf65b6eec12bcb4c30
ba488d9 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:09:39 up 1 day,  1:03,  1 user,  load average: 0.08,
 0.02, 0.01 c39e0dfa7104629029c77edb84730d5faf147d
16f8ed4 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:09:38 up 1 day,  1:03,  1 user,  load average: 0
.00, 0.00, 0.00 48ba759cdb85e2d69e35735285feec00852293a2
```



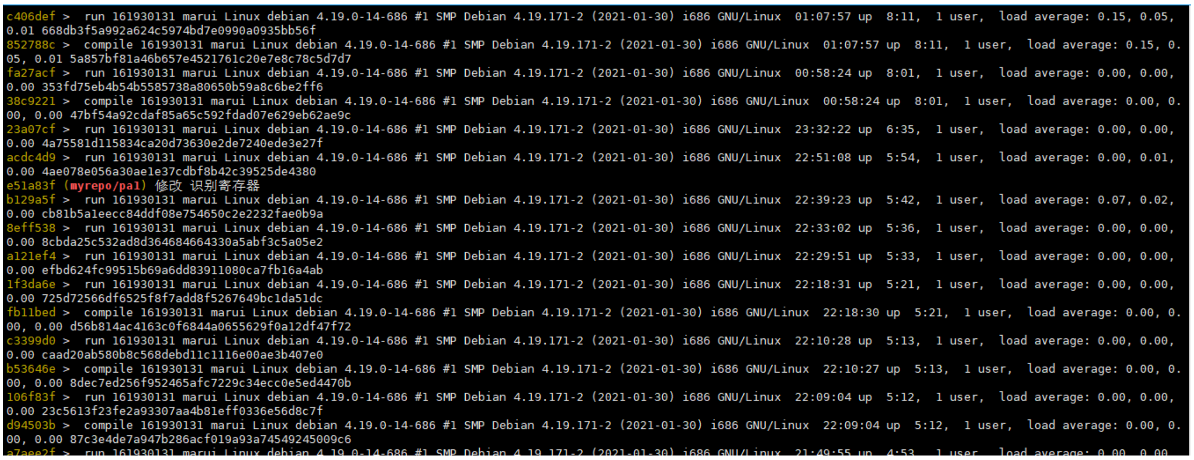
```
281a19a >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:50:03 up 1 day,  4:43,  1 user,  load average: 0.02,
 0.01, 0.00 d43ee151a19611b18f645b0ea8bb578cb9c1bca
ec65b15 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:50:03 up 1 day,  4:43,  1 user,  load average: 0
.02, 0.01, 0.00 e60132dc10fbff2e969eea793b515d273ab07282
e5b9da4 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:47:10 up 1 day,  4:40,  1 user,  load average: 0.01,
 0.01, 0.00 ac66b57bf0e5c0f559f99168cde4245551c3799f
bda5432 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:47:10 up 1 day,  4:40,  1 user,  load average: 0
.01, 0.01, 0.00 6cad79f1e32f973b1a5385d96634887d15b58e89
71d0b02 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:46:06 up 1 day,  4:39,  1 user,  load average: 0.04,
 0.01, 0.00 1c835ef5df11fece418594e7588e714a9f85c2af
24e18ba >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:46:06 up 1 day,  4:39,  1 user,  load average: 0
.04, 0.01, 0.00 d91f9725d965dcd3e197f70f5a34016f47fb8fa
c726ad5 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:43:59 up 1 day,  4:37,  1 user,  load average: 0.00,
 0.00, 0.00 a96505f2282583aee713c0c99ad1a76945cb0fb5
178b523 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:43:58 up 1 day,  4:37,  1 user,  load average: 0
.00, 0.00, 0.00 7ecca1b439b7fca2639382ef91c8512e3391bac
6a48255 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:34:25 up 1 day,  4:27,  1 user,  load average: 0.00,
 0.00, 0.00 7a1a60434906c2e252f1a16cfba7a9067ff65258
3042bbf >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:14:19 up 1 day,  4:07,  1 user,  load average: 0.00,
 0.00, 0.00 aab4047e7c2d0fe937a223f2f51a028f5f42720
3f5b827 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:14:19 up 1 day,  4:07,  1 user,  load average: 0
.00, 0.00, 0.00 74477365a82acfdb1879549ed96a273fe9610f07
f263bd2 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:03:05 up 1 day,  3:56,  1 user,  load average: 0.00,
 0.00, 0.00 78e172fdbfdd12ff4d43d8b16ab32a67710a24cf
5a8b595 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:03:04 up 1 day,  3:56,  1 user,  load average: 0
.00, 0.00, 0.00 9aec3378376913f942eb975972bcda4a2f1809cd
ab43f53 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:00:45 up 1 day,  3:54,  1 user,  load average: 0.00,
 0.00, 0.00 26dd3c3f2725599b416d7d8eec9e912645791ce
e0546f5 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:00:45 up 1 day,  3:54,  1 user,  load average: 0
.00, 0.00, 0.00 daeb2da44ab279eec9d4a101b8dffa65cf7d5dac
bd1324e >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:55:55 up 1 day,  3:49,  1 user,  load average: 0.00,
 0.00, 0.00 42248aa8598a3faedd1494e1b69d7e5f83f4fb6
```



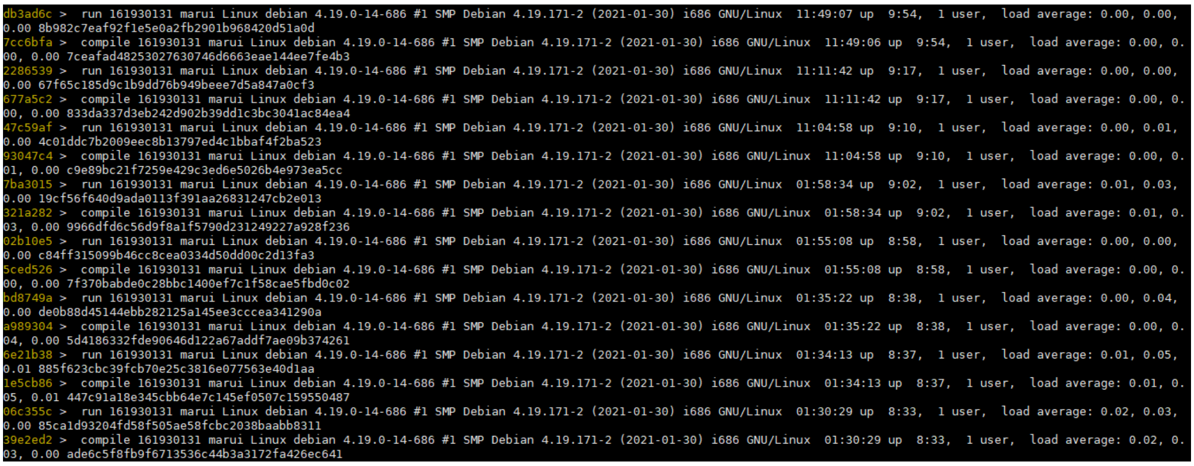
```
.02, 0.01, 0.00 d9f9bcecd404dd24a7ad28c3721604055958060
e65ab18 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  02:16:16 up 1 day,  7:09,  1 user,  load average: 0.08,
 0.02, 0.01 e3166b2e150872a523156724e0f40c5787e63e42
7655aad >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  02:16:16 up 1 day,  7:09,  1 user,  load average: 0
.00, 0.00, 0.00 409c72a4fe7a5dc623f447842b1ad329b357f85
5984bce before starting pa2
fcb0acf >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:50:55 up 1 day,  5:44,  1 user,  load average: 0.00,
 0.00, 0.00 d478af02d655b6ab90295360e3dab0436560e57d
402d357 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:28:20 up 1 day,  5:21,  1 user,  load average: 0.00,
 0.00, 0.00 3a841b6e9a4773ae08417d2a680de32d228087b
f90fa68 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:27:53 up 1 day,  5:21,  1 user,  load average: 0.00,
 0.00, 0.00 ece349013385badb3dbd4d8790f7b4ca9ad58bbc
cf56595 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:27:53 up 1 day,  5:21,  1 user,  load average: 0
.00, 0.00, 0.00 6d160fffb91c367790fa9090414c33c9ba62e106
f878332 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:11:08 up 1 day,  5:04,  1 user,  load average: 0.00,
 0.00, 0.00 4e6596f033ed7049e9ccbb4258838ef152cd8b50
39824cf >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:11:08 up 1 day,  5:04,  1 user,  load average: 0
.00, 0.00, 0.00 6df5f61bcd6003669c0871e2b308b6a7e9ceba3
c1e2005 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:10:11 up 1 day,  5:03,  1 user,  load average: 0.00,
 0.00, 0.00 26b724464ca86c7b95f142f711782277cea140d4
5664f42 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:10:11 up 1 day,  5:03,  1 user,  load average: 0
.00, 0.00, 0.00 3e7a1218eee005826e9169c855b1aa2732622a4b
734f929 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:08:42 up 1 day,  5:02,  1 user,  load average: 0.00,
 0.00, 0.00 ec3e6f4835e8bd8a18ffd579e1ef029b1bfaf6e5
441798a >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:06:24 up 1 day,  4:59,  1 user,  load average: 0.00,
 0.01, 0.00 70f9d4689b278a1bb9acaf4755ac670afe44dd65
e10ed0f >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:06:24 up 1 day,  4:59,  1 user,  load average: 0
.00, 0.01, 0.00 5c0a2dc0b5762fd1b54e76c590bb3eadc676ddc
5b32f28 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:02:07 up 1 day,  4:55,  1 user,  load average: 0.12,
 0.04, 0.01 75de6ce76c198ff93adc18c296be29df1251de9
cd99817 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:02:06 up 1 day,  4:55,  1 user,  load average: 0
.12, 0.04, 0.01 e55940a828c2232bae880685ea0f2f126989d0ce
```



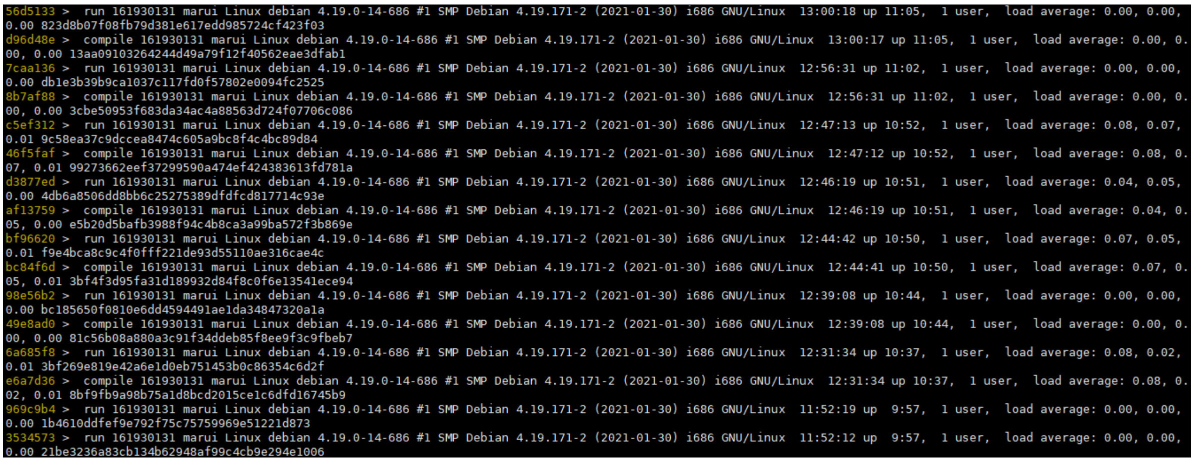
```
0.00 fbe2b4612ef70e4a243f942724e14972708ecd8
540aa3a >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:46:10 up  4:49,  1 user,  load average: 0.01, 0.02,
0.00 cee6442479285e8c2117cb937c6bab66de4d7dd0
8870bd6 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:46:10 up  4:49,  1 user,  load average: 0.01, 0.
02, 0.00 bbde9db0fb2205daa292adad322307a2cf1ad378
f159f66 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:43:25 up  4:46,  1 user,  load average: 0.02, 0.01,
0.00 d8dcf53bd330e63d58d73ef7206da81bc6a66b94
d1d3943 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:43:24 up  4:46,  1 user,  load average: 0.02, 0.
01, 0.00 5f1e465e2b103703d31b53d4286b0bfb5e939a0c
4c13612 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:42:11 up  4:45,  1 user,  load average: 0.08, 0.02,
0.01 9b6c9aae2de9bc1bdbd6661c3608aeb2b074d198
bf5c564 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:42:10 up  4:45,  1 user,  load average: 0.00, 0.
00, 0.00 cc8120f9d9a47499fd980dcbb002cd6bdc7876ee
555f6eb failed to commit pa1.2
9b1195d commit pa1.2
e453bb2 delete pa2
f716d94 finish pa1.2
dc98fac >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:22:15 up 1 day,  8:59,  1 user,  load average: 0.00,
 0.00, 0.00 e416f207ff3560456861010af625e9da2575d881
3372cf5 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:22:14 up 1 day,  8:59,  1 user,  load average: 0
.00, 0.00, 0.00 36a27ddcf4a28c3487564561224d0b3f27c7c035
c47a8e0 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:20:51 up 1 day,  8:57,  1 user,  load average: 0.00,
 0.00, 0.00 343713292d5076b7ca26d6c9693cf264658f85b
d59b726 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:20:51 up 1 day,  8:57,  1 user,  load average: 0
.00, 0.00, 0.00 a7451e48e82dd332e9e204b7adaee1a04682ac2c
99c5580 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:28:06 up 1 day,  8:05,  1 user,  load average: 0.00,
 0.00, 0.00 54b1f29db509d81ac8288deae7932302c5c01924
9bfd2b4 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:28:06 up 1 day,  8:05,  1 user,  load average: 0
.00, 0.00, 0.00 8eb85f664b6baf290ff05e27e35f518776e335a
4394182 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:19:50 up 1 day,  7:56,  1 user,  load average: 0.02,
 0.01, 0.00 c7b26710ea0c4280286ae8b273f6c132c8f2a1d0
ef34d34 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:19:50 up 1 day,  7:56,  1 user,  load average: 0
```



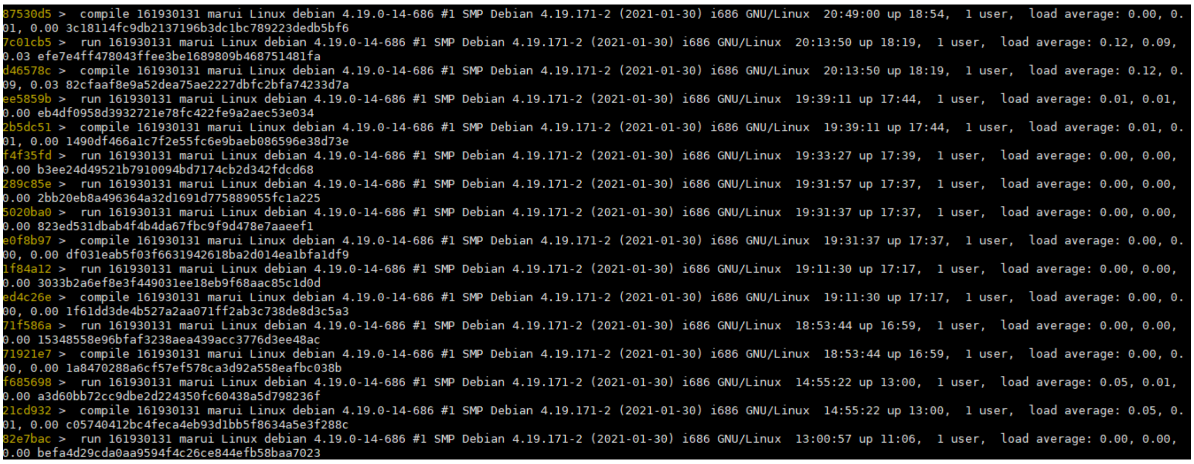
```
c406def >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:07:57 up  8:11,  1 user,  load average: 0.15, 0.05,
0.01 668db3f5a992a624c5974bd7e0990a0935bb56f
852788c >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:07:57 up  8:11,  1 user,  load average: 0.15, 0.
05, 0.01 5a857bf81a46b657e4521761c20e7e8c78c5d7d7
fa27acf >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:58:24 up  8:01,  1 user,  load average: 0.00, 0.00,
0.00 353fd75eb4b54b5585738a80650b59a8c6be2ff6
38c9221 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:58:24 up  8:01,  1 user,  load average: 0.00, 0.
00, 0.00 47bf54a92cdaf85a65c592fdad07e629eb62ae9c
23a07cf >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:32:22 up  6:35,  1 user,  load average: 0.00, 0.00,
0.00 4a75581d115834ca20d73630e2de7240ede3e27f
acdc4d9 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:51:08 up  5:54,  1 user,  load average: 0.00, 0.01,
0.00 4ae078e056a30ae1e37cdbf8b42c39525de4380
e51a83f (myrepo/pa1) 修改 识别寄存器
b129a5f >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:39:23 up  5:42,  1 user,  load average: 0.07, 0.02,
0.00 cb81b5a1eecc84ddf08e754650c2e2232fae0b9a
8eff538 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:33:02 up  5:36,  1 user,  load average: 0.00, 0.00,
0.00 8cbda25c532ad8d364684664330a5abf3c5a05e2
a121ef4 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:29:51 up  5:33,  1 user,  load average: 0.00, 0.00,
0.00 efbd624fc99515b69a6dd83911080ca7fb16a4ab
1f3da6e >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:18:31 up  5:21,  1 user,  load average: 0.00, 0.00,
0.00 725d72566df6525f8f7add8f5267649bc1da51dc
fb11bed >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:18:30 up  5:21,  1 user,  load average: 0.00, 0.
00, 0.00 d56b814ac4163c0f6844a0655629f0a12df47f72
c3399d0 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:10:28 up  5:13,  1 user,  load average: 0.00, 0.00,
0.00 caad20ab580b8c568debd11c1116e00ae3b407e0
b53646e >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:10:27 up  5:13,  1 user,  load average: 0.00, 0.
00, 0.00 8dec7ed256f952465afc7229c34ecc0e5ed4470b
106f83f >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:09:04 up  5:12,  1 user,  load average: 0.00, 0.00,
0.00 23c5613f23fe2a93307aa4b81eff0336e56d8c7f
d94503b >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  22:09:04 up  5:12,  1 user,  load average: 0.00, 0.
00, 0.00 87c3e4de7a947b286acf019a93a74549245009c6
a7aee2f >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:49:55 up  4:53,  1 user,  load average: 0.00, 0.00
```



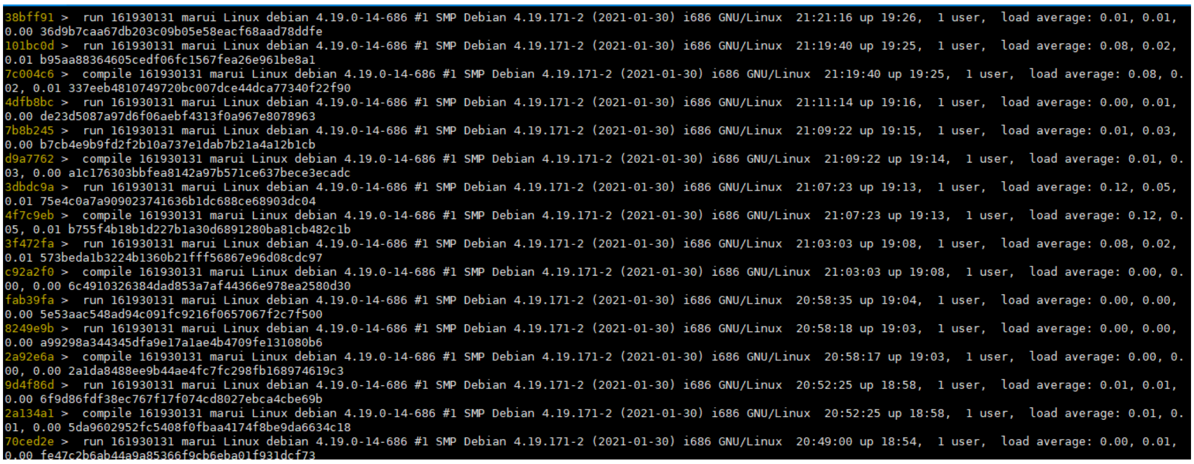
```
db3ad6c >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:49:07 up  9:54,  1 user,  load average: 0.00, 0.00,
0.00 8b982c7eaf92f1e5e0a2fb2901b968420d51a0d
7cc6bfa >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:49:06 up  9:54,  1 user,  load average: 0.00, 0.
00, 0.00 7ceafad48253027630746d6663eae144ee7fe4b3
2286539 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:11:42 up  9:17,  1 user,  load average: 0.00, 0.00,
0.00 67f65c185d9c1b9dd76b949beee7d5a847a0cf3
677a5c2 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:11:42 up  9:17,  1 user,  load average: 0.00, 0.
00, 0.00 833da337d3eb242d902b39dd1c3bc3041ac84ea4
47c59af >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:04:58 up  9:10,  1 user,  load average: 0.00, 0.01,
0.00 4c01ddc7b2009eec8b13797ed4c1bbaf4f2ba523
93047c4 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:04:58 up  9:10,  1 user,  load average: 0.00, 0.
01, 0.00 c9e89bc21f7259e429c3ed6e5026b4e973ea5cc
7ba3015 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:58:34 up  9:02,  1 user,  load average: 0.01, 0.03,
0.00 19cf56f640d9ada0113f391aa26831247cb2e013
321a282 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:58:34 up  9:02,  1 user,  load average: 0.01, 0.
03, 0.00 9966dfd6c56d9f8a1f5790d231249227a928f236
02b10e5 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:55:08 up  8:58,  1 user,  load average: 0.00, 0.00,
0.00 c84ff315099b46cc8cea0334d50dd00c2d13fa3
5ced526 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:55:08 up  8:58,  1 user,  load average: 0.00, 0.
00, 0.00 7f370babde0c28bbc1400ef7c1f58cae5fbd0c02
bd8749a >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:35:22 up  8:38,  1 user,  load average: 0.00, 0.04,
0.00 de0b88d45144ebb282125a145ee3cccea341290a
a989304 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:35:22 up  8:38,  1 user,  load average: 0.00, 0.
04, 0.00 5d4186332fde90646d122a67addf7ae09b374261
6e21b38 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:34:13 up  8:37,  1 user,  load average: 0.01, 0.05,
0.01 885f623cbc39fcb70e25c3816e077563e40d1aa
1e5cb86 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:34:13 up  8:37,  1 user,  load average: 0.01, 0.
05, 0.01 447c91a18e345cbb64e7c145ef0507c159550487
06c355c >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:30:29 up  8:33,  1 user,  load average: 0.02, 0.03,
0.00 85ca1d93204fd58f505ae58fcbc2038baabb8311
39e2ed2 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  01:30:29 up  8:33,  1 user,  load average: 0.02, 0.
03, 0.00 ade6c5f8fb9f6713536c44b3a3172fa426ec641
```



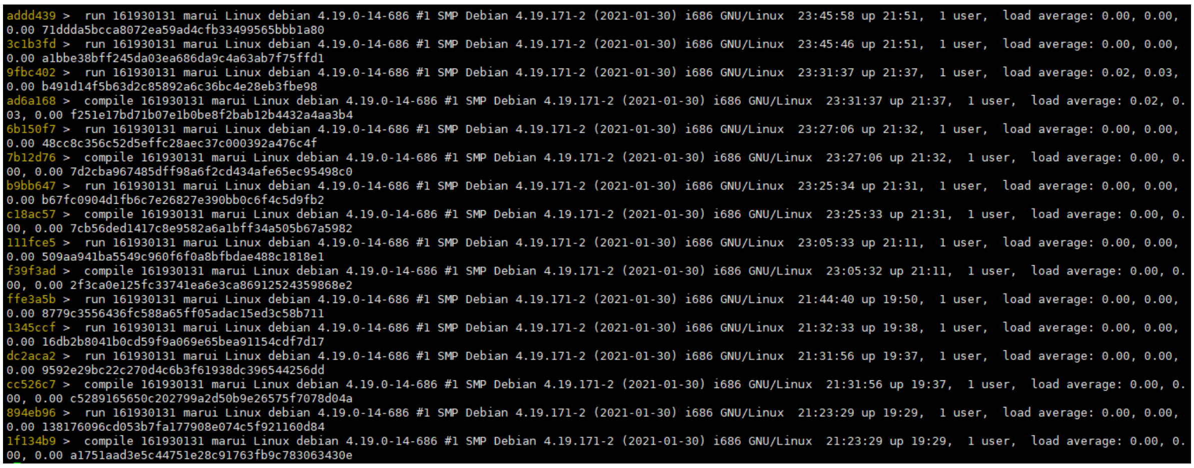
```
56d5133 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  13:00:18 up 11:05,  1 user,  load average: 0.00, 0.00,
0.00 823d8b07f08fb79d381e617edd985724cf423f03
d96d48e >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  13:00:17 up 11:05,  1 user,  load average: 0.00, 0.
00, 0.00 13aa09103264244d49a79f12f40562eae3dfab1
7caa136 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:56:31 up 11:02,  1 user,  load average: 0.00, 0.00,
0.00 db1e3b39b9ca1037c117fd0f57802e0094fc2525
8b7af88 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:56:31 up 11:02,  1 user,  load average: 0.00, 0.
00, 0.00 3cbe50953f683da34ac4a88563d724f07706c086
c5ef312 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:47:13 up 10:52,  1 user,  load average: 0.08, 0.07,
0.01 9c58ea37c9dccea8474c605a9bc8f4c4bc89d84
46f5faf >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:47:12 up 10:52,  1 user,  load average: 0.08, 0.
07, 0.01 99273662eef37299590a474ef424383613fd781a
d3877ed >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:46:19 up 10:51,  1 user,  load average: 0.04, 0.05,
0.00 4db6a8506dd8bb6c25275389dfdfcd817714c93e
af13759 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:46:19 up 10:51,  1 user,  load average: 0.04, 0.
05, 0.00 e5b20d5bafb3988f94c4b8ca3a99ba572f3b869e
bf96620 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:44:42 up 10:50,  1 user,  load average: 0.07, 0.05,
0.01 f9e4bca8c9c4f0fff221de93d55110ae316cae4c
bc84f6d >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:44:41 up 10:50,  1 user,  load average: 0.07, 0.
05, 0.01 3bf4f3d95fa31d189932d84f8c0f6e13541ece94
98e56b2 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:39:08 up 10:44,  1 user,  load average: 0.00, 0.00,
0.00 bc185650f0810e6dd4594491ae1da34847320a1a
49e8ad0 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:39:08 up 10:44,  1 user,  load average: 0.00, 0.
00, 0.00 81c56b08a880a3c91f34ddeb85f8ee9f3c9fbeb7
6a685f8 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:31:34 up 10:37,  1 user,  load average: 0.08, 0.02,
0.01 3bf269e819e42a6e1d0eb751453b0c86354c6d2f
e6a7d36 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  12:31:34 up 10:37,  1 user,  load average: 0.08, 0.
02, 0.01 8bf9fb9a98b75a1d8bcd2015ce1c6dfd16745b9
969c9b4 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:52:19 up  9:57,  1 user,  load average: 0.00, 0.00,
0.00 1b4610ddfef9e792f75c75759969e51221d873
3534573 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:52:12 up  9:57,  1 user,  load average: 0.00, 0.00,
0.00 21be3236a83cb134b62948af99c4cb9e294e1006
```



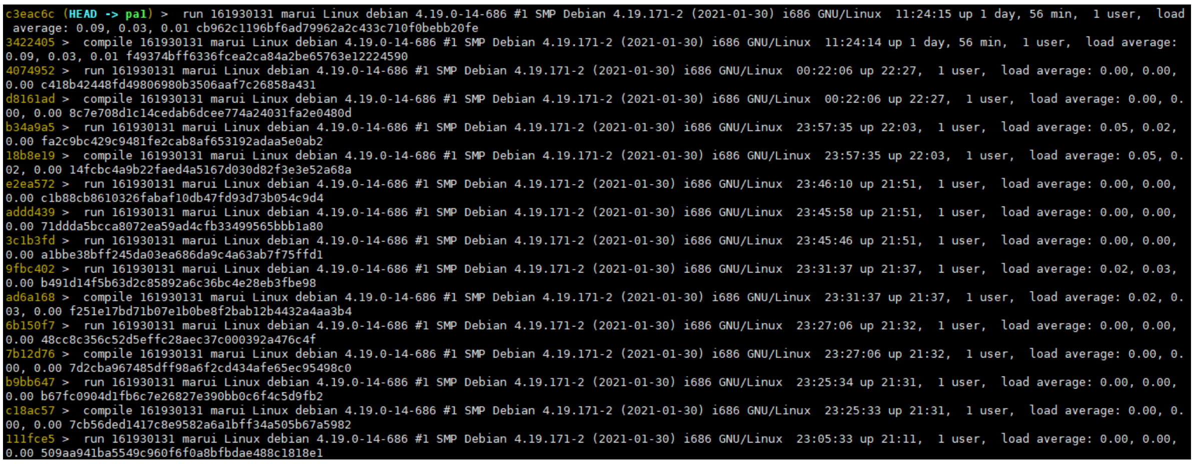
```
87530d5 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:49:00 up 18:54,  1 user,  load average: 0.00, 0.
01, 0.00 3c18114fc9db2137196b3dc1bc789223dedb5bf6
7c01cb5 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:13:50 up 18:19,  1 user,  load average: 0.12, 0.09,
0.03 efe7e4ff478043ffee3be1689809b468751481fa
d46578c >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:13:50 up 18:19,  1 user,  load average: 0.12, 0.
09, 0.03 82cfaaf8e9a52dea75ae2227dbfc2bfa74233d7a
ee5859b >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:39:11 up 17:44,  1 user,  load average: 0.01, 0.01,
0.00 eb4df0958d3932721e78fc422fe9a2aec53e034
2b5dc51 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:39:11 up 17:44,  1 user,  load average: 0.01, 0.
01, 0.00 1490df466a1c7f2e55fc6e9baeb086596e38d73e
f4f35fd >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:33:27 up 17:39,  1 user,  load average: 0.00, 0.00,
0.00 b3ee24d49521b7910094bd7174cb2d342fdcd68
289c85e >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:31:57 up 17:37,  1 user,  load average: 0.00, 0.00,
0.00 2bb20eb8a496364a32d1691d775889055fc1a225
5020ba0 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:31:37 up 17:37,  1 user,  load average: 0.00, 0.00,
0.00 823ed531dbab4f4b4da67fbc9f9d478e7aaeef1
e0f8b97 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:31:37 up 17:37,  1 user,  load average: 0.00, 0.
00, 0.00 df031eab5f03f6631942618ba2d014ea1bfa1df9
1f84a12 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:11:30 up 17:17,  1 user,  load average: 0.00, 0.00,
0.00 3033b2a6ef8e3f449031ee18eb9f68aac85c1d0d
ed4c26e >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  19:11:30 up 17:17,  1 user,  load average: 0.00, 0.
00, 0.00 1f61dd3de4b527a2aa071ff2ab3c738de8d3c5a3
71f586a >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  18:53:44 up 16:59,  1 user,  load average: 0.00, 0.00,
0.00 15348558e96bfaf3238aea439acc3776d3ee48ac
71921e7 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  18:53:44 up 16:59,  1 user,  load average: 0.00, 0.
00, 0.00 1a8470288a6cf57ef578ca3d92a558eafbc038b
f685698 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  14:55:22 up 13:00,  1 user,  load average: 0.05, 0.01,
0.00 a3d60bb72cc9dbe2d224350fc60438a5d798236f
21cd932 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  14:55:22 up 13:00,  1 user,  load average: 0.05, 0.
01, 0.00 c05740412bc4feca4eb93d1bb5f8634a5e3f288c
82e7bac >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  13:00:57 up 11:06,  1 user,  load average: 0.00, 0.00,
0.00 befa4d29cda0aa9594f4c26ce844efb58baa7023
```



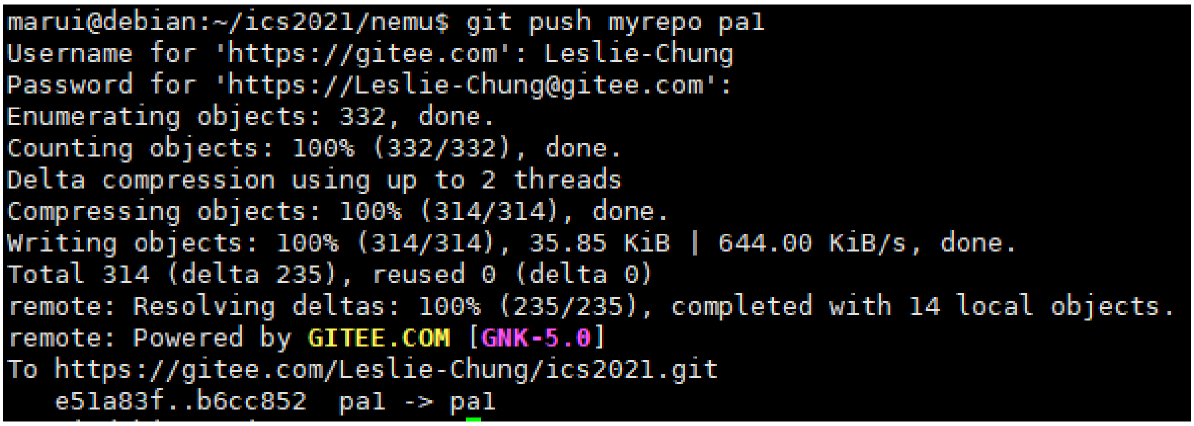
```
38bff91 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:21:16 up 19:26,  1 user,  load average: 0.01, 0.01,
0.00 36d9b7caa67db203c09b05e58eacf68aad78ddfe
101bc0d >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:19:40 up 19:25,  1 user,  load average: 0.08, 0.02,
0.01 b95aa88364605cedf06fc1567fea26e961be8a1
7c004c6 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:19:40 up 19:25,  1 user,  load average: 0.08, 0.
02, 0.01 337eeb4810749720bc007dce44dca77340f22f90
4dfb8bc >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:11:14 up 19:16,  1 user,  load average: 0.00, 0.01,
0.00 de23d5087a97d6f06aebf4313f0a967e8078963
7b8b245 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:09:22 up 19:15,  1 user,  load average: 0.01, 0.03,
0.00 b7cb4e9b9fd2f2b10a737e1dab7b21a4a12b1cb
d9a7762 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:09:22 up 19:14,  1 user,  load average: 0.01, 0.
03, 0.00 a1c176303bbfea8142a97b571ce637bece3ecadc
3dbdc9a >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:07:23 up 19:13,  1 user,  load average: 0.12, 0.05,
0.01 75e4c0a7a909023741636b1dc688ce68903dc04
4f7c9eb >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:07:23 up 19:13,  1 user,  load average: 0.12, 0.
05, 0.01 b755f4b18b1d227b1a30d6891280ba81cb482c1b
3f472fa >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:03:03 up 19:08,  1 user,  load average: 0.08, 0.02,
0.01 573beda1b3224b1360b21fff56867e96d08cdc97
c92a2f0 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:03:03 up 19:08,  1 user,  load average: 0.00, 0.
00, 0.00 6c4910326384dad853a7af44366e978ea2580d30
fab39fa >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:58:35 up 19:04,  1 user,  load average: 0.00, 0.00,
0.00 5e53aac548ad94c091fc9216f0657067f2c7f500
8249e9b >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:58:18 up 19:03,  1 user,  load average: 0.00, 0.00,
0.00 a99298a344345dfa9e17a1ae4b4709fe131080b6
2a92e6a >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:58:17 up 19:03,  1 user,  load average: 0.00, 0.
00, 0.00 2a1da8488ee9b44ae4fc7fc298fb168974619c3
9d4f86d >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:52:25 up 18:58,  1 user,  load average: 0.01, 0.01,
0.00 6f9d86fdf38ec767f17f074cd8027ebca4cbe69b
2a134a1 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:52:25 up 18:58,  1 user,  load average: 0.01, 0.
01, 0.00 5da9602952fc5408f0fbaa4174f8be9da6634c18
70ced2e >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  20:49:00 up 18:54,  1 user,  load average: 0.00, 0.01,
0.00 fe47c2b6ab44a9a85366f9cb6eba01f931dcf73
```



```
addd439 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:45:58 up 21:51,  1 user,  load average: 0.00, 0.00,
0.00 71ddda5bcca8072ea59ad4cfb33499565bbb1a80
3c1b3fd >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:45:46 up 21:51,  1 user,  load average: 0.00, 0.00,
0.00 a1bbe38bff245da03ea686da9c4a63ab7f75ffd1
9fbc402 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:31:37 up 21:37,  1 user,  load average: 0.02, 0.03,
0.00 b491d14f5b63d2c85892a6c36bc4e28eb3fbe98
ad6a168 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:31:37 up 21:37,  1 user,  load average: 0.02, 0.
03, 0.00 f251e17bd71b07e1b0be8f2bab12b4432a4aa3b4
6b150f7 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:27:06 up 21:32,  1 user,  load average: 0.00, 0.00,
0.00 48cc8c356c52d5effc28aec37c000392a476c4f
7b12d76 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:27:06 up 21:32,  1 user,  load average: 0.00, 0.
00, 0.00 7d2cba967485dff98a6f2cd434afe65ec95498c0
b9bb647 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:25:34 up 21:31,  1 user,  load average: 0.00, 0.00,
0.00 b67fc0904d1fb6c7e26827e390bb0c6f4c5d9fb2
c18ac57 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:25:33 up 21:31,  1 user,  load average: 0.00, 0.
00, 0.00 7cb56ded1417c8e9582a6a1bff34a505b67a5982
111fce5 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:05:33 up 21:11,  1 user,  load average: 0.00, 0.00,
0.00 509aa941ba5549c960f6f0a8bfbdae488c1818e1
f39f3ad >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:05:32 up 21:11,  1 user,  load average: 0.00, 0.
00, 0.00 2f3ca0e125fc33741ea6e3ca86912524359868e2
ffe3a5b >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:44:40 up 19:50,  1 user,  load average: 0.00, 0.00,
0.00 8779c3556436fc588a65ff05adac15ed3c58b711
1345ccf >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:32:33 up 19:38,  1 user,  load average: 0.00, 0.00,
0.00 16db2b8041b0cd59f9a069e65bea91154cdf7d17
dc2aca2 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:31:56 up 19:37,  1 user,  load average: 0.00, 0.00,
0.00 9592e29bc22c270d4c6b3f61938dc396544256dd
cc526c7 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:31:56 up 19:37,  1 user,  load average: 0.00, 0.
00, 0.00 c5289165650c202799a2d50b9e26575f7078d04a
894eb96 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:23:29 up 19:29,  1 user,  load average: 0.00, 0.00,
0.00 138176096cd053b7fa177908e074c5f921160d84
1f134b9 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  21:23:29 up 19:29,  1 user,  load average: 0.00, 0.
00, 0.00 a1751aad3e5c44751e28c91763fb9c783063430e
```



```
c3eac6c (HEAD -> pa1) >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:24:15 up 1 day, 56 min,  1 user,  load
 average: 0.09, 0.03, 0.01 cb962c1196bf6ad79962a2c433c710f0bebb20fe
3422405 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  11:24:14 up 1 day, 56 min,  1 user,  load average:
0.09, 0.03, 0.01 f49374bff6336fcea2ca84a2be65763e12224590
4074952 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:22:06 up 22:27,  1 user,  load average: 0.00, 0.00,
0.00 c418b42448fd49806980b3506aaf7c26858a431
d8161ad >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  00:22:06 up 22:27,  1 user,  load average: 0.00, 0.
00, 0.00 8c7e708d1c14cedab6dcee774a24031fa2e0480d
b34a9a5 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:57:35 up 22:03,  1 user,  load average: 0.05, 0.02,
0.00 fa2c9bc429c9481fe2cab8af653192adaa5e0ab2
18b8e19 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:57:35 up 22:03,  1 user,  load average: 0.05, 0.
02, 0.00 14fcbc4a9b22faed4a5167d030d82f3e3e52a68a
e2ea572 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:46:10 up 21:51,  1 user,  load average: 0.00, 0.00,
0.00 c1b88cb8610326fabaf10db47fd93d73b054c9d4
addd439 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:45:58 up 21:51,  1 user,  load average: 0.00, 0.00,
0.00 71ddda5bcca8072ea59ad4cfb33499565bbb1a80
3c1b3fd >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:45:46 up 21:51,  1 user,  load average: 0.00, 0.00,
0.00 a1bbe38bff245da03ea686da9c4a63ab7f75ffd1
9fbc402 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:31:37 up 21:37,  1 user,  load average: 0.02, 0.03,
0.00 b491d14f5b63d2c85892a6c36bc4e28eb3fbe98
ad6a168 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:31:37 up 21:37,  1 user,  load average: 0.02, 0.
03, 0.00 f251e17bd71b07e1b0be8f2bab12b4432a4aa3b4
6b150f7 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:27:06 up 21:32,  1 user,  load average: 0.00, 0.00,
0.00 48cc8c356c52d5effc28aec37c000392a476c4f
7b12d76 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:27:06 up 21:32,  1 user,  load average: 0.00, 0.
00, 0.00 7d2cba967485dff98a6f2cd434afe65ec95498c0
b9bb647 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:25:34 up 21:31,  1 user,  load average: 0.00, 0.00,
0.00 b67fc0904d1fb6c7e26827e390bb0c6f4c5d9fb2
c18ac57 >  compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:25:33 up 21:31,  1 user,  load average: 0.00, 0.
00, 0.00 7cb56ded1417c8e9582a6a1bff34a505b67a5982
111fce5 >  run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux  23:05:33 up 21:11,  1 user,  load average: 0.00, 0.00,
0.00 509aa941ba5549c960f6f0a8bfbdae488c1818e1
```



```
marui@debian:~/ics2021/nemu$ git push myrepo pa1
Username for 'https://gitee.com': Leslie-Chung
Password for 'https://Leslie-Chung@gitee.com':
Enumerating objects: 332, done.
Counting objects: 100% (332/332), done.
Delta compression using up to 2 threads
Compressing objects: 100% (314/314), done.
Writing objects: 100% (314/314), 35.85 KiB | 644.00 KiB/s, done.
Total 314 (delta 235), reused 0 (delta 0)
remote: Resolving deltas: 100% (235/235), completed with 14 local objects.
remote: Powered by GITEE.COM [GNK-5.0]
To https://gitee.com/Leslie-Chung/ics2021.git
   e51a83f..b6cc852  pa1 -> pa1
```

实验内容

`PA1.2`：除了`cmd_p`命令加在`nemu/src/debug/ui.c`外，其余都加在`nemu/src/debug/expr.c`中

PA1.2.1 编写匹配规则(1) + (2)

注意：

先检测16进制，再检测10进制，否则`0x`会被当做`0`和`x`。（`!=`要写在`!`前面......）

```
enum {
  TK_NOTYPE = 256,
  TK_EQ,
  TK_HEX,
  TK_DEC,
  /*TK_EAX,
  TK_EBX,
  TK_ECX,
  TK_EDX,
  TK_EDI,
  TK_ESI,
  TK_EBP,
  TK_ESP,
  TK_EIP,*/
  TK_REG,
  TK_NQ,
  TK_AND,
  TK_OR,
  TK_MINUS,//负号
  TK_DEREF,//指针解引用
  TK_LE,
  TK_GE,
  TK_ML,
  TK_MR
};

static struct rule {
  char *regex;
  int token_type;
} rules[] = {
    {" +", TK_NOTYPE},    // spaces
    {"==", TK_EQ},         // equal
    {"0x[0-9a-fA-F]{1,8}", TK_HEX}, //先检测16进制，再检测10进制，否则0x会被当做0  x
    {"[0-9]+", TK_DEC},
    /*{"\\$eax", TK_EAX},
    {"\\$ebx", TK_EBX},
    {"\\$ecx", TK_ECX},
    {"\\$edx", TK_EDX},
    {"\\$edi", TK_EDI},
    {"\\$esi", TK_ESI},
    {"\\$ebp", TK_EBP},
    {"\\$esp", TK_ESP},
    {"\\$eip", TK_EIP},*/
    {"\\$[a-zA-Z]{2,3}", TK_REG},
    {"\\(", '('},
    {"\\)", ')'},
    {"\\+", '+'},
    {"-", '-'},
    {"\\*", '*'},
    {"/", '/'},
    {"!=", TK_NQ},
    {"&&", TK_AND},
    {"\\|\\|", TK_OR},
    {"!", '!'},

    {"~", '~'},
    {"%", '%'},
    {"\\|", '|'},
    {"&", '&'},
    {"\\^", '^'},
    {"<=", TK_LE},
    {">=", TK_GE},
    {"<<", TK_ML},
    {">>", TK_MR},
    {"<", '<'},
    {">", '>'}
};
```

PA1.2.2 添加 p命令

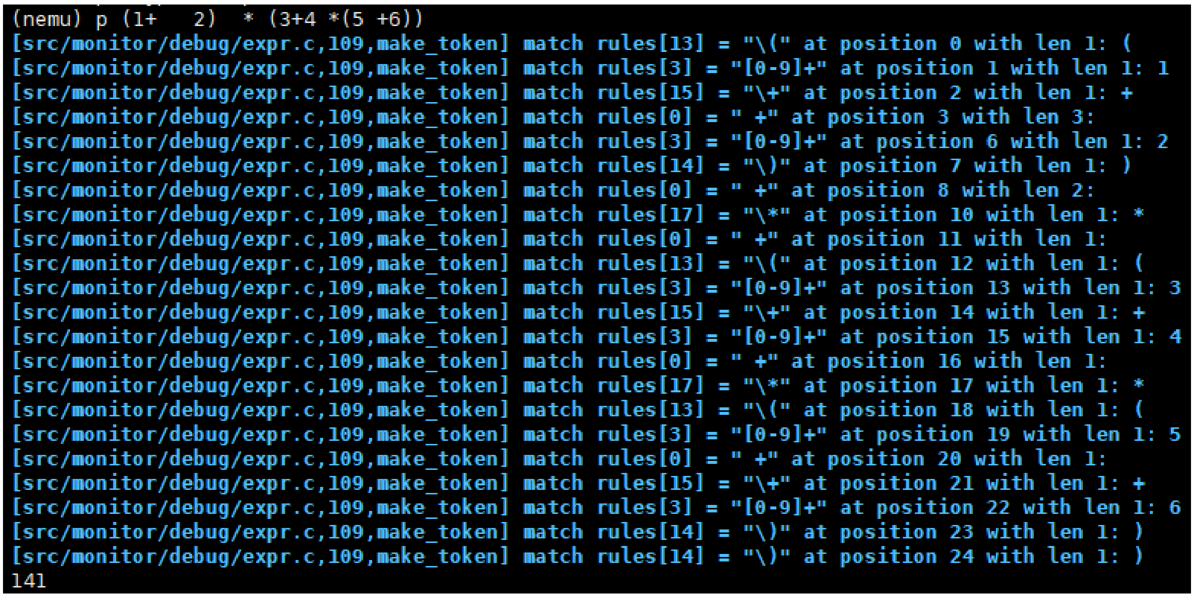
声明并定义函数

```
static int cmd_p(char * args);
...
static int cmd_p(char * args){
    bool success = true;
    uint32_t value = expr(args,&success);
    if(success){
        printf("%u\n", value);
    }
    return 0;
}
```

将`p`命令加入指令列表中：

```
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","Usage: si [N]\n"\
          "     Execute the program with N(default: 1) step", cmd_si },
  { "info", "Show information about registers with argument 'r' and show information about watchpoint with argument 'w'", cmd_info},
  { "x", "Usage: x [N] [EXPR]\n" \
      "    Calculate the value of the expression EXPR, and output N consecutive 4 bytes starting from EXPR in hexadecimal form", cmd_x },
  { "p", "Usage: p [EXPR]\n" "    Calculate the value of the expression EXPR",
cmd_p},
  /* TODO: Add more commands */
};
```

测试样例：



```
(nemu) p (1+   2)  * (3+4 *(5 +6))
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 1 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 2 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 3 with len 3:
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 6 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 7 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 8 with len 2:
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 10 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 12 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 13 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 14 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 15 with len 1: 4
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 16 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 17 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 18 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 19 with len 1: 5
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 20 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 21 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 22 with len 1: 6
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 23 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 24 with len 1: )
141
```

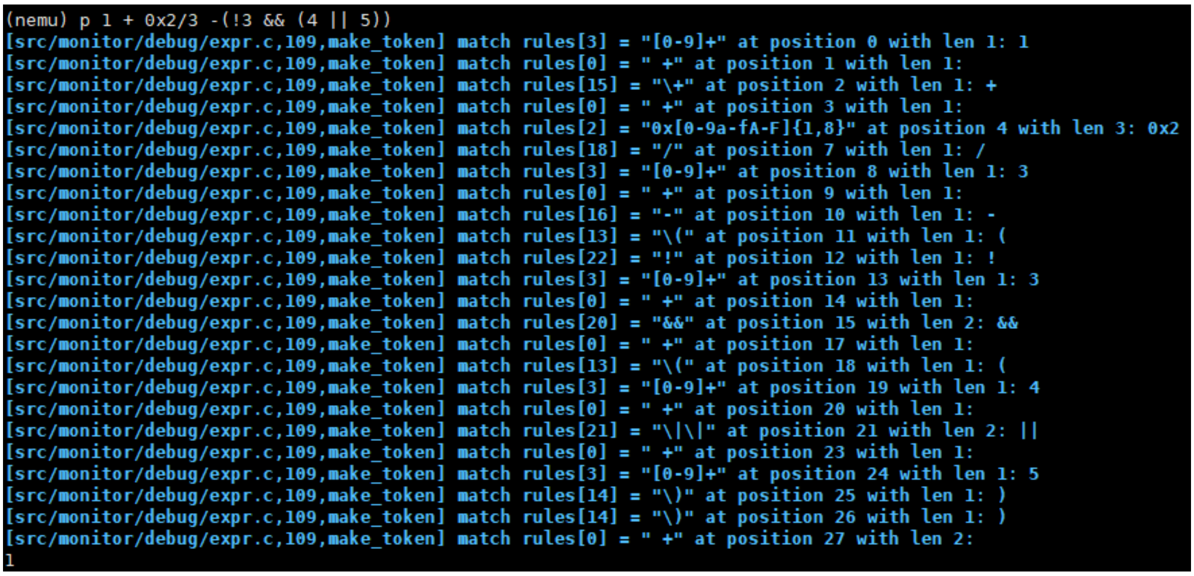
PA1.2.3 识别并存储 token

完善`make_token`：

```
static bool make_token(char *e) {
    int position = 0;
    int i;
    regmatch_t pmatch;
    if(e == NULL) return false;
    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.
             */
            if(substr_len >= 32){
                printf("%.*s  The length of the substring is too long.\n", substr_len, substr_start);
                //在用*%.\*s*时,后面跟着两个参数,一个表示输出数据占得位置的大小,一个表示要输出的内容
                return false;
            }

            if(nr_token >= 32) {
                printf("The count of tokens(nr_token) is out of the maximum count(32)\n");
                return false;
        }
            switch (rules[i].token_type) {
                case TK_NOTYPE:
                    break;
                case TK_DEC:
                case TK_HEX:
                    strncpy(tokens[nr_token].str, substr_start, substr_len);
                    tokens[nr_token].str[substr_len] = '\0';

                default:
                    tokens[nr_token].type = rules[i].token_type;
                    nr_token++;
                    break;
            }

            break;
            }
        }

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

    return true;
}
```

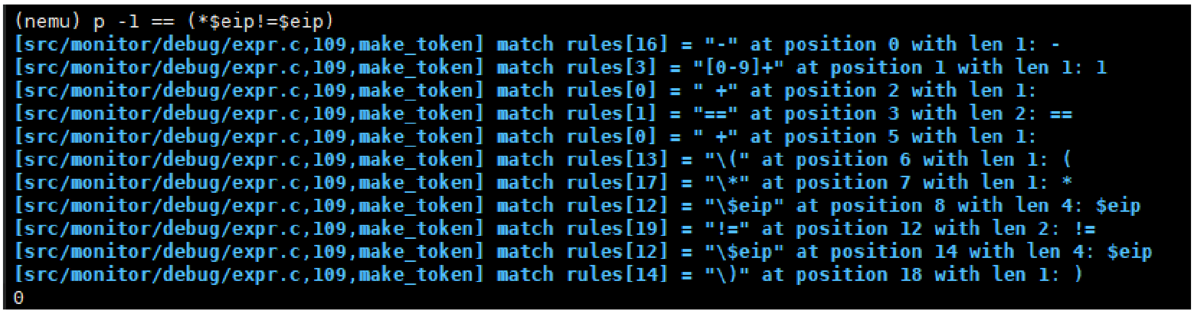
测试样例:

`p 1 + 0x2/3 - (!3 && (4 || 5))`



```
(nemu) p 1 + 0x2/3 -(!3 && (4 || 5))
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 0 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 1 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 2 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 3 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[2] = "0x[0-9a-fA-F]{1,8}" at position 4 with len 3: 0x2
[src/monitor/debug/expr.c,109,make_token] match rules[18] = "/" at position 7 with len 1: /
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 8 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 9 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 10 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 11 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[22] = "!" at position 12 with len 1: !
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 13 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 14 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[20] = "&&" at position 15 with len 2: &&
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 17 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 18 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 19 with len 1: 4
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 20 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[21] = "\|\|" at position 21 with len 2: ||
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 23 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 24 with len 1: 5
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 25 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 26 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 27 with len 2: 
1
```

`p -1 == (*$eip != $eip)`



```
(nemu) p -1 == (*$eip!=$eip)
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 0 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 1 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 2 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[1] = "==" at position 3 with len 2: ==
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 5 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 6 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 7 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[12] = "\$eip" at position 8 with len 4: $eip
[src/monitor/debug/expr.c,109,make_token] match rules[19] = "!=" at position 12 with len 2: !=
[src/monitor/debug/expr.c,109,make_token] match rules[12] = "\$eip" at position 14 with len 4: $eip
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 18 with len 1: )
0
```

PA1.2.4 实现括号匹配

思路：

先看左右括号以及数量是否匹配，如果符合，则再看最外层有没有被`()`包含；

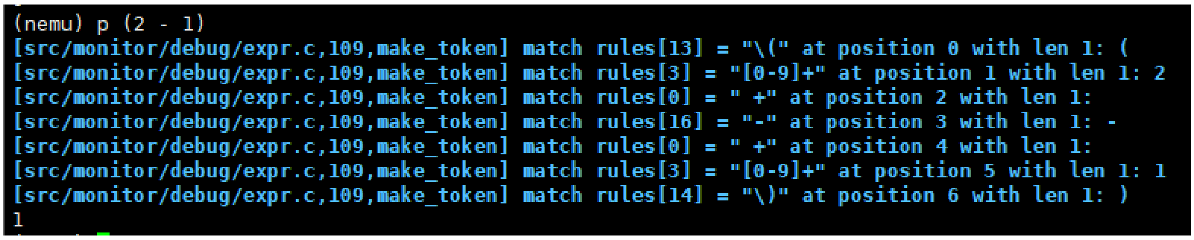
最后再看去除最外层的`()`后，其余的括号是否匹配

代码实现：

```
bool check_parentheses(int, int);
bool check_parentheses(int p, int q){
    bool lr = false;
    if (tokens[p].type == '(' && tokens[q].type == ')'){
        lr = true;
    }
    int i, l = 0;//l用来记录左括号的数量
    for(i = p; i <= q; i++){// 先看括号是否匹配
        if(tokens[i].type == '(')  l++;
        else if(tokens[i].type == ')')  l--;
        if(l < 0){
            /*右括号先出现，如())*/
            printf("Bad Expression!\n");
            assert(0);
        }
    }
    if(l != 0){//左括号数量 > 右
        printf("Bad Expression!\n");
        assert(0);
    }

    /*括号匹配，但是最外层没有()
        4 + 3 * (2 - 1)
    */
    if(!lr) return false;

    /*考虑这种情况
        (4 + 3) * (2 - 1)
    */
    //此时 l == 0
    q-- , p++;

    for(i = p; i <= q; i++){
        if(tokens[i].type == '(')  l++;
        else if(tokens[i].type == ')')  l--;
        if(l < 0){
            return false;
        }
    }
    return true;
}
```

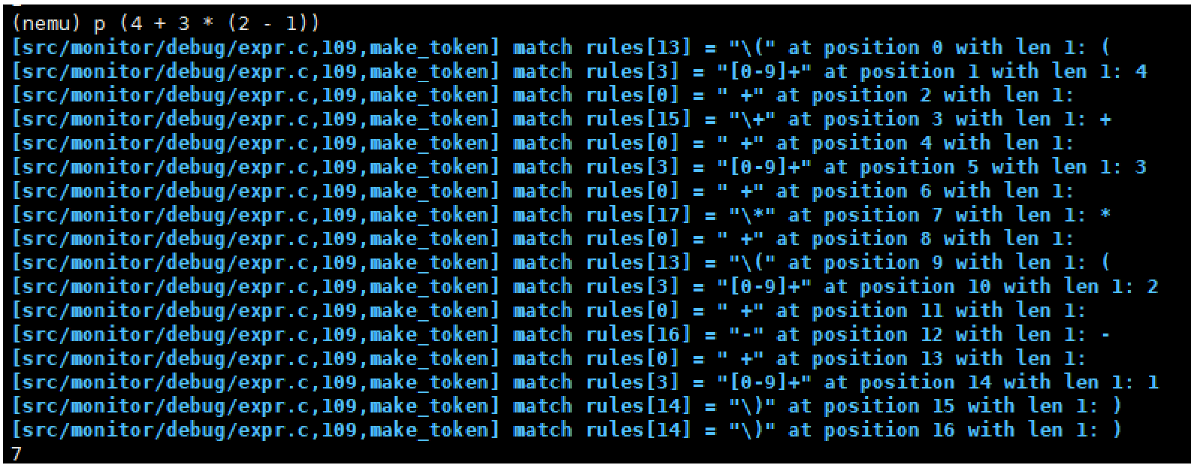
测试样例：

`p (2 - 1)`



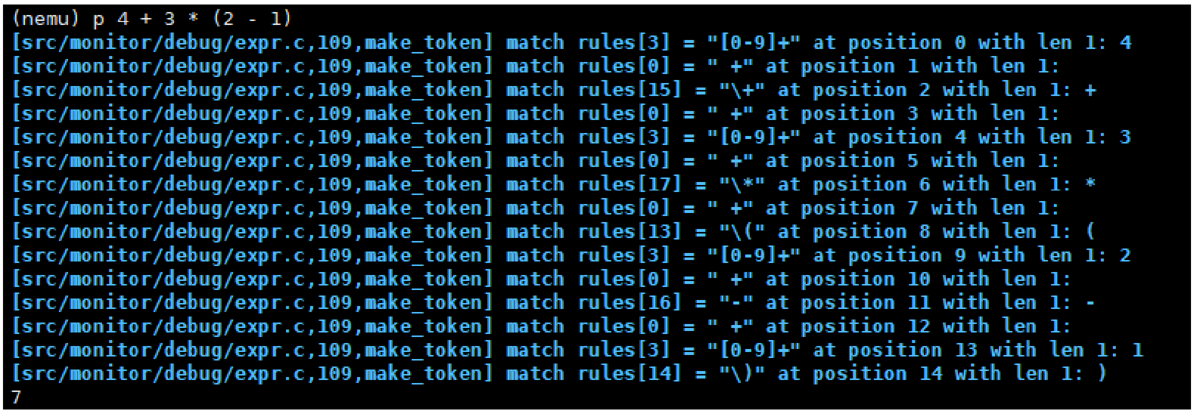
```
(nemu) p (2 - 1)
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 1 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 3 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 5 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 6 with len 1: )
1
```

`p (4 + 3 * (2 - 1))`



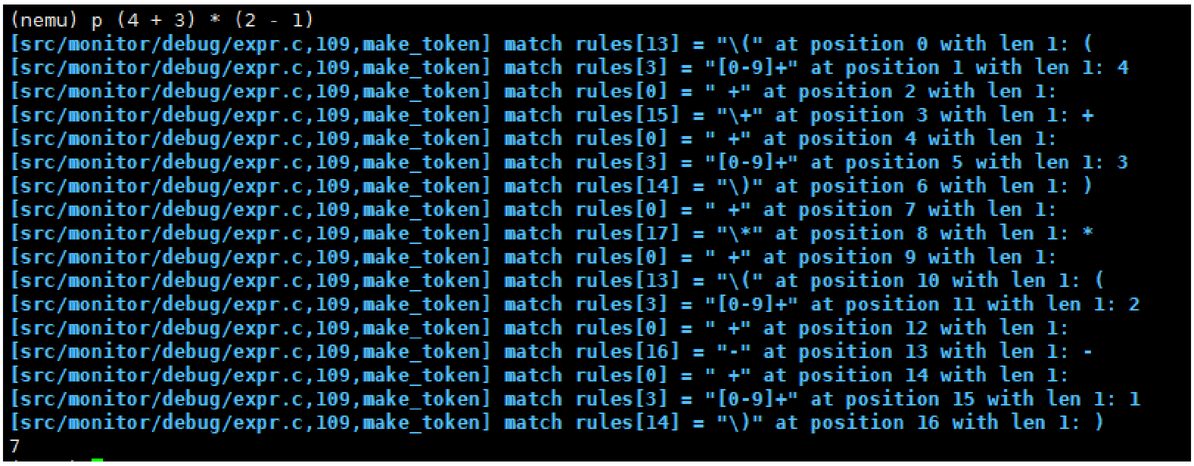
```
(nemu) p (4 + 3 * (2 - 1))
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 2 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 6 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 7 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 9 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 10 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 12 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 13 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 14 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 15 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 16 with len 1: )
7
```

```
p 4 + 3 * (2 - 1)
```



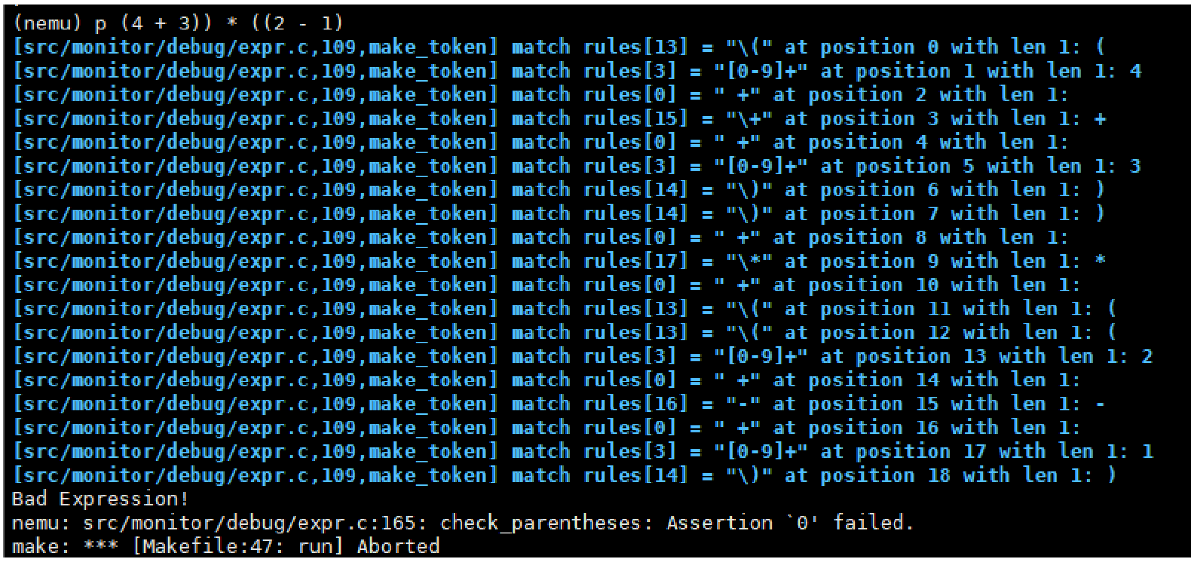
```
(nemu) p 4 + 3 * (2 - 1)
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 0 with len 1: 4
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 1 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 2 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 3 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 4 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 5 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 6 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 7 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 8 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 9 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 10 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 11 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 12 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 13 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 14 with len 1: )
7
```

```
p (4 + 3)) * ((2 - 1)
```



```
(nemu) p (4 + 3) * (2 - 1)
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 2 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 4 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 6 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 7 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 8 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 9 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 10 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 11 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 12 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 13 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 14 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 15 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 16 with len 1: )
7
```

```
p (4 + 3) * (2 - 1)
```



```
(nemu) p (4 + 3)) * ((2 - 1)
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 0 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 1 with len 1: 4
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 2 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 3 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 4 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 5 with len 1: 3
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 6 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 7 with len 1: )
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 8 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 9 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 10 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 11 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 12 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 13 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 14 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 15 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 16 with len 1: 
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 17 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 18 with len 1: )
Bad Expression!
nemu: src/monitor/debug/expr.c:165: check_parentheses: Assertion `0' failed.
make: *** [Makefile:47: run] Aborted
```

PA1.2.5 寻找当前子表达式的中心操作符

思路：

先判断是否为运算符；如果是，判断它的等级，等级高于已经遍历过的运算符，则跳过；如果小于等于，则重新赋值；特殊情况：当运算符是单目运算符的时候，最左面的优先级最小。

检测到左括号，则一直跳过，直到找到右括号。 注意：不会存在括号不匹配的情况，如果存在程序已经终止

代码实现：

```
int op_priority(int op){ //获取运算符优先级
    int level;
    switch (op) {
        case TK_OR:// ||
            level = 1;
            break;
        case TK_AND:// &&
            level = 2;
            break;
        case '|':
            level = 3;
            break;
        case '^':
            level = 4;
            break;
        case '&':
            level = 5;
            break;
        case TK_EQ:// ==  !=
        case TK_NQ:
            level = 6;
            break;
        case '<':
        case '>':
        case TK_LE:
        case TK_GE:
            level = 7;
            break;
        case TK_ML:
        case TK_MR:
            level = 8;
            break;
        case '+':
        case '-':
            level = 9;
            break;
        case '*':
        case '/':
        case '%':
            level = 10;
            break;
        case TK_DEREF:// 解引用
        case TK_MINUS:// 负号
        case '!':
        case '~':
            level = 11;//level == 11 一般都是单目运算符
            break;
        default://不会出现，因为在使用正则表达式匹配时会扫描该运算符存不存在
            assert(0);
    }
    return level;
}

int compare(int i, int pos){
    int priorityi = op_priority(tokens[i].type);
    int prioritypos = op_priority(tokens[pos].type);
    if(priorityi == prioritypos && prioritypos == 11) return 1;//如果是单目运算符，则最前面的优先级最小(即递归要从最前面的运算符开始)
    return priorityi - prioritypos;
}

bool is_op(int ch){//是否为运算符
    return ch == '+' || ch == '-' || ch == '*' || ch == '/'
            || ch == '!' || ch == TK_AND || ch == TK_OR || ch == TK_EQ || ch == TK_NQ || ch == TK_DEREF || ch == TK_MINUS || ch == '|' || ch == '&' || ch == '^' || ch == '<' || ch == '>' || ch == TK_LE || ch == TK_GE || ch == TK_ML || ch == TK_MR || ch == '%' || ch == '~';
}

int find_dominated_op(int p, int q){
    /*先判断是否为运算符
    如果是，判断它的等级，等级高于已经遍历过的运算符，则跳过；如果小于等于，则重新赋值；
    检测到（，则一直跳过，直到找到）   注意：不会存在括号不匹配的情况，如果存在程序已经终止
    */
    int pos = -1, i, opType, l = 0;//l 进行括号匹配
    for(i = p; i <= q; i++){
        opType = tokens[i].type;

        if(l == 0 && is_op(opType)){
            if(pos == -1 || compare(i, pos) <= 0) pos = i; //如果是第一个运算符，或者i的优先级小于等于pos的
        }
        else if(opType == '(') l++;
        else if(opType == ')') l--;
    }
    return pos;
}
```

选做任务：带有负数的表达式求值

思路：

判断`-`的前一个`token`的类型是不是寄存器、数字、右括号，如果是则`-`不是负号

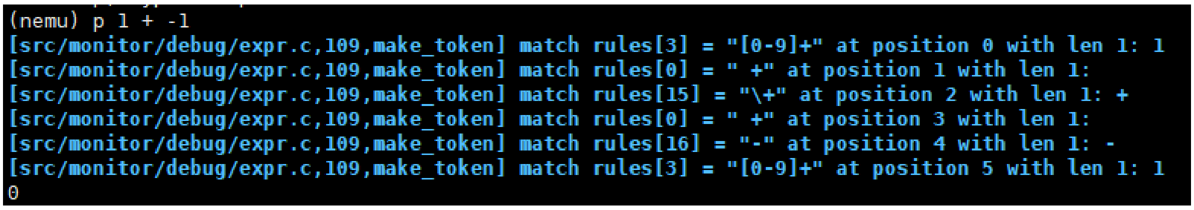
如果`-`是第一个`token`，则也是负号

代码实现：

```
bool opEvalMinus(int op){
    switch(op){
        case TK_OR:// ||
        case TK_AND:// &&
        case TK_EQ:// ==  !=
        case TK_NQ:
        case '+':
        case '-':
        case '*':
        case '/':
        case '(':
        case '!':
        case '|':
        case '&':
        case '^':
        case '<':
        case '>':
        case '%':
        case '~':
        case TK_LE:
        case TK_GE:
        case TK_ML:
        case TK_MR:
        case TK_DEREF:
        case TK_MINUS:// 负号
            return true;
        default:
            return false;
    }
}

uint32_t expr(char *e, bool *success) {
    if (!make_token(e)) {
    *success = false;
    return 0;
    }
/* TODO: Implement code to evaluate the expression. */
    int i;
    for (i = 0; i < nr_token; i ++) {
        if (tokens[i].type == '-' && (i == 0 || opEvalMinus(tokens[i - 1].type)
)) {
            tokens[i].type = TK_MINUS;
        }
    }
}
```

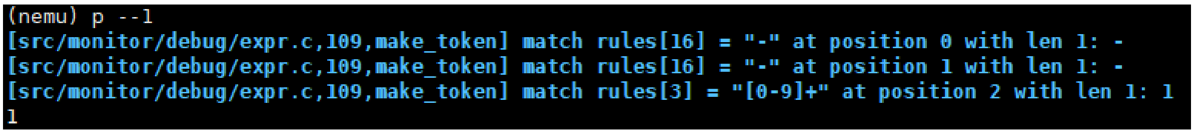
测试样例

`p 1 + -1`



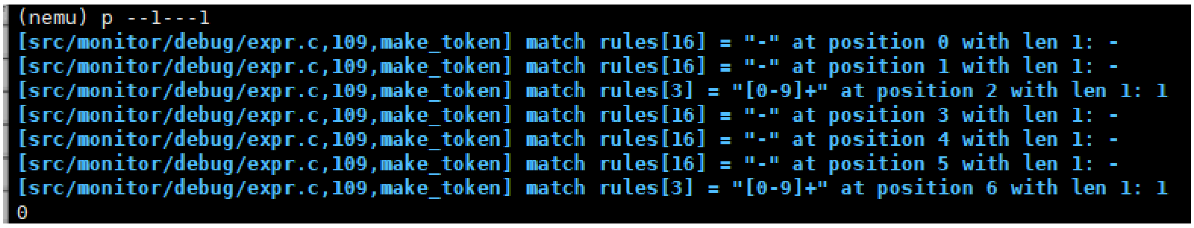
```
(nemu) p 1 + -1
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 0 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 2 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 3 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 4 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 5 with len 1: 1
0
```

`p --1`



```
(nemu) p --1
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 0 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 1 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 2 with len 1: 1
1
```

`p --1---1`



```
(nemu) p --1---1
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 0 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 1 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 2 with len 1: 1
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 3 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 4 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 5 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 6 with len 1: 1
0
```

PA1.2.6 实现指针解引用

思路：

判断`*`的前一个`token`的类型是不是寄存器、数字、右括号，如果是则`*`不是指针解引用

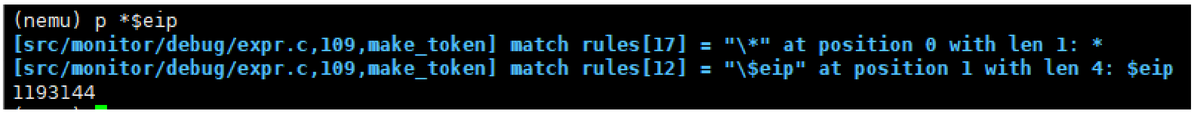
如果`*`是第一个`token`，则也是指针解引用

代码实现：

```
bool opEvalDeref(int op){
    switch(op){
        case TK_OR:// ||
        case TK_AND:// &&
        case TK_EQ:// ==  !=
        case TK_NQ:
        case '+':
        case '-':
        case '*':
        case '/':
        case '(':
        case '!':
        case '|':
        case '&':
        case '^':
        case '<':
        case '>':
        case '%':
        case '~':
        case TK_LE:
        case TK_GE:
        case TK_ML:
        case TK_MR:
        case TK_DEREF:
        case TK_MINUS:// 负号
            return true;
        default:
            return false;
    }
}

uint32_t expr(char *e, bool *success) {
    if (!make_token(e)) {
    *success = false;
    return 0;
    }
/* TODO: Implement code to evaluate the expression. */
    ...
    for (i = 0; i < nr_token; i ++) {
        if (tokens[i].type == '*' && (i == 0 || opEvalDeref(tokens[i -
1].type))) {
            tokens[i].type = TK_DEREF;
        }
    }
}
```

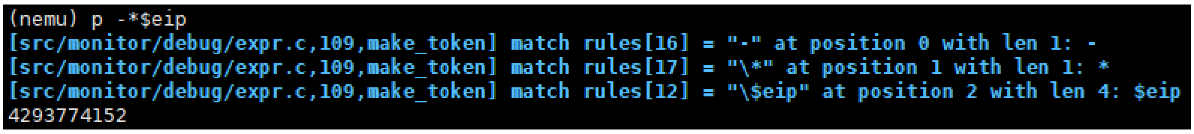
测试样例：

`p *$eip`



```
(nemu) p *$eip
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 0 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[12] = "\$eip" at position 1 with len 4: $eip
1193144
```

`p -*$eip`



```
(nemu) p -*$eip
[src/monitor/debug/expr.c,109,make_token] match rules[16] = "-" at position 0 with len 1: -
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 1 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[12] = "\$eip" at position 2 with len 4: $eip
4293774152
```

PA1.2.7 实现表达式求值

思路：

`p == q`：如果是寄存器、十进制、十六进制数，则直接返回相应的值，此时肯定达到了递归终止的条件

如果成功匹配括号，则对括号内的表达式递归求值

否则，找到表达式中优先级最低的运算符位置，然后根据其单目 / 双目计算子表达式。

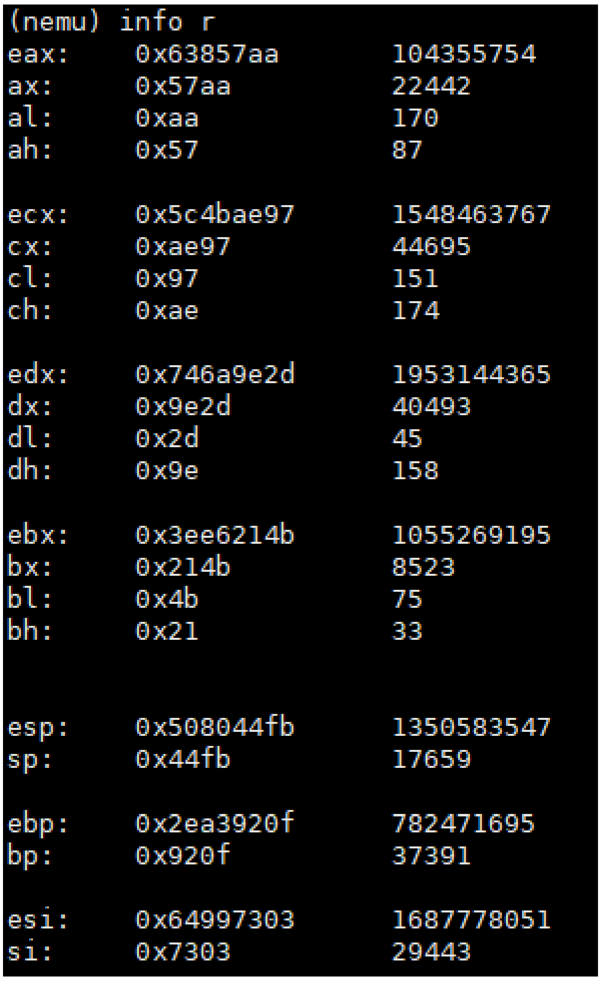
其次，在表达式求值之前，要先判断 `*` 是乘法还是指针解引用、 `-`是减法还是负号。

代码实现：

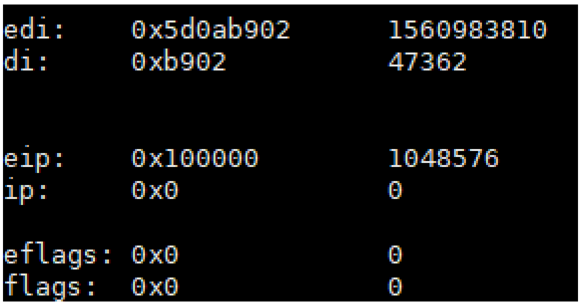
```
uint32_t is_reg(char *str){
    int i;
    for(i = R_EAX; i <= R_EDI; ++ i) {
        if(strcmp(str, regsl[i]) == 0) return reg_l(i);
    }
    for(i = R_AX; i <= R_DI; ++ i){
        if(strcmp(str, regsw[i]) == 0) return reg_w(i);
    }
    for(i = R_AL; i <= R_BH; ++ i){
        if(strcmp(str, regsb[i]) == 0) return reg_b(i);
    }
    if(strcmp(str, "eip") == 0) return cpu.eip;
    printf("Reg doesn't exit!\n");
    assert(0);
}

uint32_t eval(int p, int q) {
    if (p > q) {
        printf("Wrong: p > q\n");
        assert(0);
    }
    else if(p == q){
        switch (tokens[q].type) {
/*              case TK_EAX:
                return cpu.eax;
            case TK_EBX:
                return cpu.ebx;
            case TK_ECX:
                return cpu.ecx;
            case TK_EDX:
                return cpu.edx;
            case TK_EDI:
                return cpu.edi;
            case TK_ESI:
                return cpu.esi;
            case TK_EBP:
                return cpu.ebp;
            case TK_ESP:
                return cpu.esp;
            case TK_EIP:
                return cpu.eip;*/
            case TK_REG:
                return is_reg(tokens[q].str + 1);
            case TK_DEC:
                return atoi(tokens[p].str);
            case TK_HEX:
                uint32_t hexNum = 0;
                sscanf(tokens[p].str, "%x", &hexNum);
                return hexNum;
            default:
                assert(0);
        }
    }
    else if(check_parentheses(p, q) == true) {
        return eval(p + 1, q - 1);
    }
    else {
        int op, val1, val2;
        op = find_dominated_op(p, q);
        if(op == p){//单目运算
            switch (tokens[op].type) {
                case '~':
                    return ~eval(op + 1, q);
                case TK_MINUS:
                    return -eval(op + 1, q);
                case '!':
                    return !eval(op + 1, q);
                case TK_DEREF:
                    return vaddr_read(eval(op + 1, q), 4);
                default:
                    assert(0);
            }
        }
        val1 = eval(p, op - 1);
        val2 = eval(op + 1, q);
        switch (tokens[op].type) {//双目运算
            case '+':
                return val1 + val2;
            case '-':
                return val1 - val2;
            case '*':
                return val1 * val2;
            case '/':
                return val1 / val2;
            case '%':
                return val1 % val2;
            case '|':
                return val1 | val2;
            case '&':
                return val1 & val2;
            case '<':
                return val1 < val2;
            case '>':
                return val1 > val2;
            case '^':
                return val1 ^ val2;
            case TK_EQ:
                return val1 == val2;
            case TK_NQ:
                return val1 != val2;
            case TK_AND:
                return val1 && val2;
            case TK_OR:
                return val1 || val2;
            case TK_LE:
                return val1 <= val2;
            case TK_GE:
                return val1 >= val2;
            case TK_ML:
                return val1 << val2;
            case TK_MR:
                return val1 >> val2;
            default:
                assert(0);
        }
    }
    return 0;//其实不会执行到这个
}

uint32_t expr(char *e, bool *success) {
    if (!make_token(e)) {
        *success = false;
        return 0;
    }
    int i;
    for (i = 0; i < nr_token; i ++) {
        if (tokens[i].type == '*' && (i == 0 || opEvalDeref(tokens[i - 1].type))
{
            tokens[i].type = TK_DEREF;
        }
    }
    for (i = 0; i < nr_token; i ++) {
        if (tokens[i].type == '-' && (i == 0 || opEvalMinus(tokens[i - 1].type))
{
            tokens[i].type = TK_MINUS;
        }
    }
    return eval(0, nr_token - 1);
}
```

测试样例：

先执行`info r`

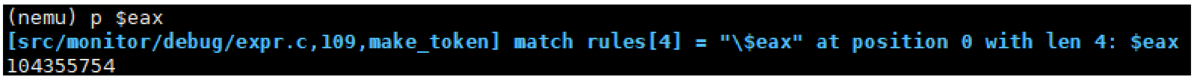


```
(nemu) info r
eax:    0x63857aa       104355754
ax:     0x57aa          22442
al:     0xaa            170
ah:     0x57            87

ecx:    0x5c4bae97      1548463767
cx:     0xae97          44695
cl:     0x97            151
ch:     0xae            174

edx:    0x746a9e2d      1953144365
dx:     0x9e2d          40493
dl:     0x2d            45
dh:     0x9e            158

ebx:    0x3ee6214b      1055269195
bx:     0x214b          8523
bl:     0x4b            75
bh:     0x21            33


esp:    0x508044fb      1350583547
sp:     0x44fb          17659

ebp:    0x2ea3920f      782471695
bp:     0x920f          37391

esi:    0x64997303      1687778051
si:     0x7303          29443
```



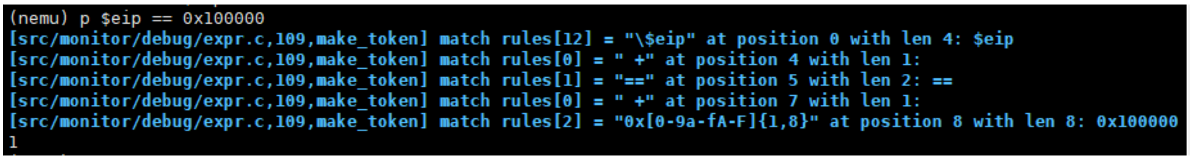
```
edi:    0x5d0ab902      1560983810
di:     0xb902          47362


eip:    0x100000        1048576
ip:     0x0             0

eflags: 0x0             0
flags:  0x0             0
```

再执行`p $eax`



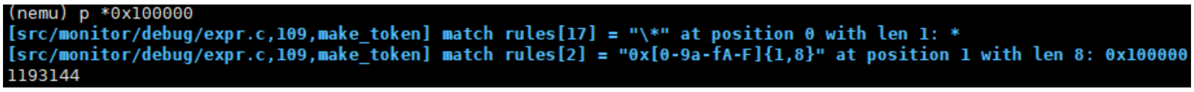
```
(nemu) p $eax
[src/monitor/debug/expr.c,109,make_token] match rules[4] = "\$eax" at position 0 with len 4: $eax
104355754
```

`p $eip == 0x100000`



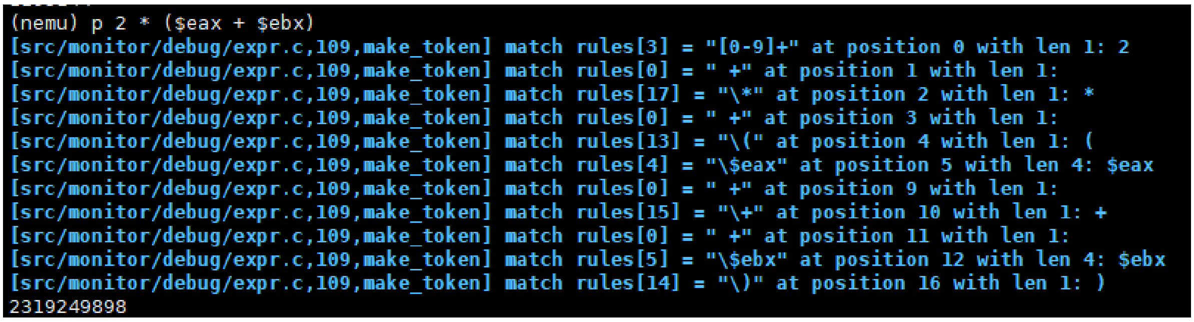
```
(nemu) p $eip == 0x100000
[src/monitor/debug/expr.c,109,make_token] match rules[12] = "\$eip" at position 0 with len 4: $eip
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[1] = "==" at position 5 with len 2: ==
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 7 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[2] = "0x[0-9a-fA-F]{1,8}" at position 8 with len 8: 0x100000
1
```

`p *$eip`



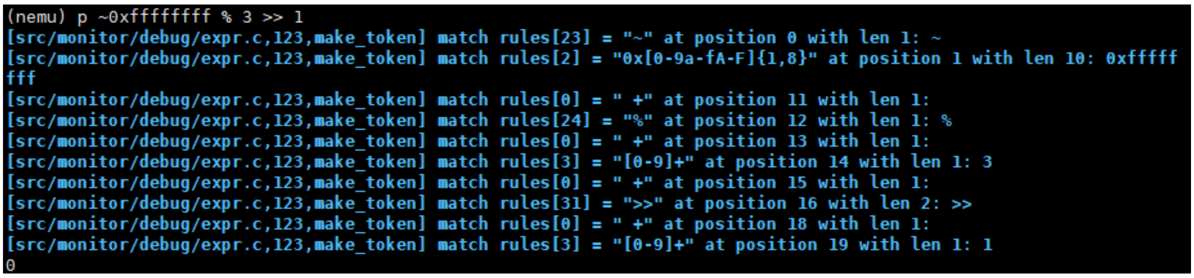
```
(nemu) p *0x100000
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 0 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[2] = "0x[0-9a-fA-F]{1,8}" at position 1 with len 8: 0x100000
1193144
```

`p 2 * ($eax + $ebx)`



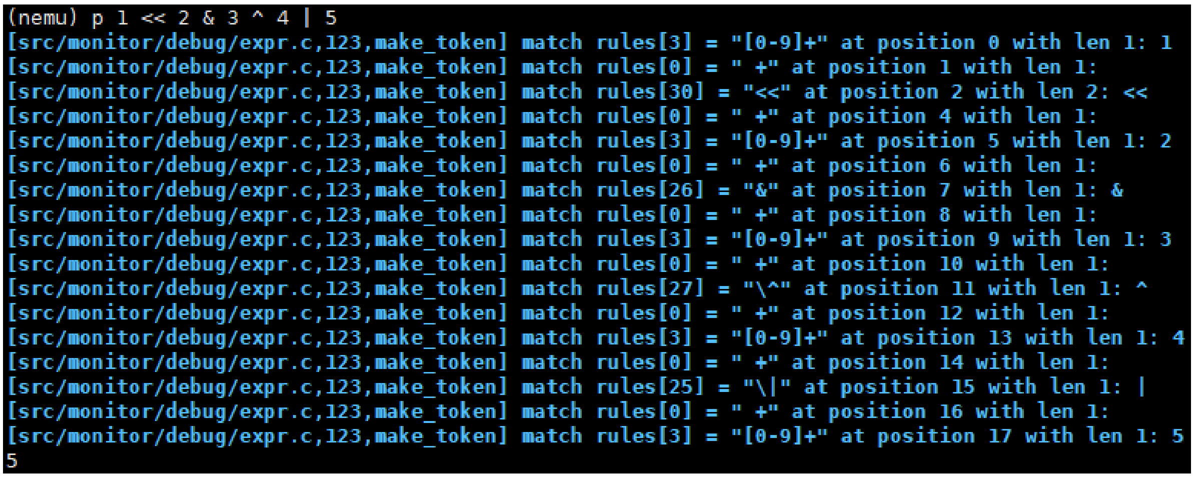
```
(nemu) p 2 * ($eax + $ebx)
[src/monitor/debug/expr.c,109,make_token] match rules[3] = "[0-9]+" at position 0 with len 1: 2
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[17] = "\*" at position 2 with len 1: *
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 3 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[13] = "\(" at position 4 with len 1: (
[src/monitor/debug/expr.c,109,make_token] match rules[4] = "\$eax" at position 5 with len 4: $eax
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 9 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[15] = "\+" at position 10 with len 1: +
[src/monitor/debug/expr.c,109,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,109,make_token] match rules[5] = "\$ebx" at position 12 with len 4: $ebx
[src/monitor/debug/expr.c,109,make_token] match rules[14] = "\)" at position 16 with len 1: )
2319249898
```

`p ~0xffffffff % 3 >> 1`



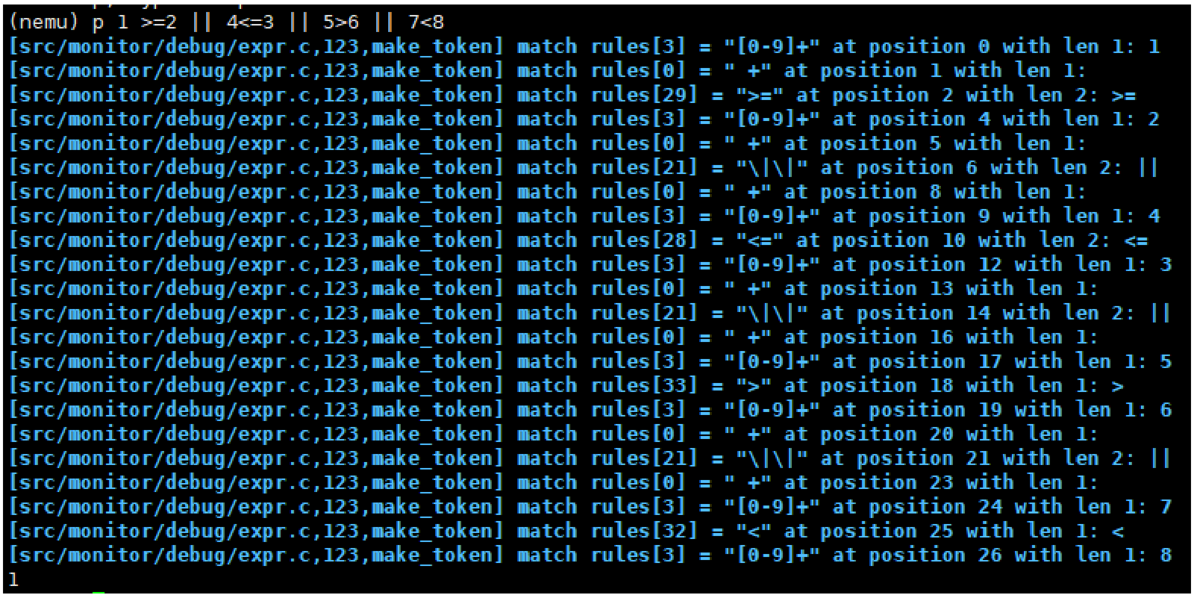
```
(nemu) p ~0xffffffff % 3 >> 1
[src/monitor/debug/expr.c,123,make_token] match rules[23] = "~" at position 0 with len 1: ~
[src/monitor/debug/expr.c,123,make_token] match rules[2] = "0x[0-9a-fA-F]{1,8}" at position 1 with len 10: 0xfffff
fff
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 11 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[24] = "%" at position 12 with len 1: %
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 13 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 14 with len 1: 3
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 15 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[31] = ">>" at position 16 with len 2: >>
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 18 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 19 with len 1: 1
0
```

`p 1 << 2 & 3 ^ 4 | 5`



```
(nemu) p 1 << 2 & 3 ^ 4 | 5
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 0 with len 1: 1
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[30] = "<<" at position 2 with len 2: <<
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 4 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 5 with len 1: 2
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 6 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[26] = "&" at position 7 with len 1: &
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 9 with len 1: 3
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 10 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[27] = "\^" at position 11 with len 1: ^
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 12 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 13 with len 1: 4
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 14 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[25] = "\|" at position 15 with len 1: |
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 16 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 17 with len 1: 5
5
```

`p 1 >=2 || 4<=3 || 5>6 || 7<8`



```
(nemu) p 1 >=2 || 4<=3 || 5>6 || 7<8
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 0 with len 1: 1
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 1 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[29] = ">=" at position 2 with len 2: >=
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 4 with len 1: 2
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 5 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[21] = "\|\|" at position 6 with len 2: ||
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 8 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 9 with len 1: 4
[src/monitor/debug/expr.c,123,make_token] match rules[28] = "<=" at position 10 with len 2: <=
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 12 with len 1: 3
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 13 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[21] = "\|\|" at position 14 with len 2: ||
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 16 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 17 with len 1: 5
[src/monitor/debug/expr.c,123,make_token] match rules[33] = ">" at position 18 with len 1: >
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 19 with len 1: 6
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 20 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[21] = "\|\|" at position 21 with len 2: ||
[src/monitor/debug/expr.c,123,make_token] match rules[0] = " +" at position 23 with len 1:
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 24 with len 1: 7
[src/monitor/debug/expr.c,123,make_token] match rules[32] = "<" at position 25 with len 1: <
[src/monitor/debug/expr.c,123,make_token] match rules[3] = "[0-9]+" at position 26 with len 1: 8
1
```

其他运算符样例已经在前面的任务中展示。

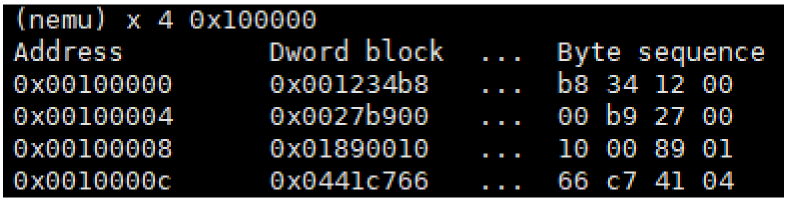
选做任务：实现x命令使用表达式求值

修改`cmd_x`函数，返回表达式表示的值

```c
static int cmd_x(char *args){
    char *arg1 = strtok(NULL, " ");
    char *arg2 = strtok(NULL, " ");
    if (arg1 == NULL || arg2 == NULL) {
        printf("A parameter is missing!\n");
        return 0;
    }
    int n = atoi(arg1); //读取要读取的次数

    if (n < 1){
        printf("Invalid arguments for x!\n");
        return 0;
    }
    char *arg3 = strtok(NULL, " ");
    while(arg3 != NULL){
        strcat(arg2, arg3);
        arg3 = strtok(NULL, " ");
    }

    int i;
    uint32_t data, addr;
    bool success = true;
    addr = expr(arg2, &success);
    if(!success) return 0;
    printf("Address         Dword block   ...  Byte sequence\n");
    //循环使用 vaddr_read 函数来读取内存
    for (i = 1; i <= n; i++, addr += 4){
        data = vaddr_read(addr, 4);
        printf("0x%08x\t", addr);
        printf("0x%08x", data);
        byteSequence_dispaly(data);
    }
    return 0;
}
```

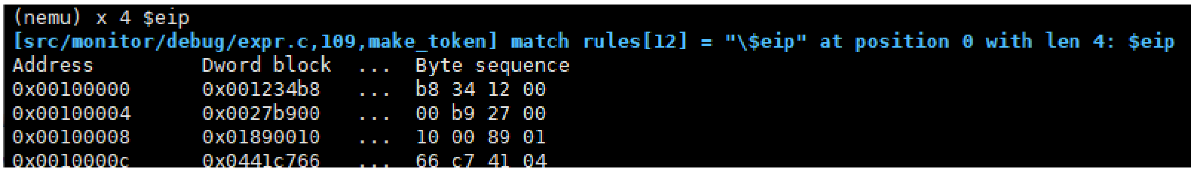
测试样例：

`x 4 0x100000`



```
(nemu) x 4 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
```

`x 4 $eip`



```
(nemu) x 4 $eip
[src/monitor/debug/expr.c,109,make_token] match rules[12] = "\$eip" 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
```

在此之后先将`make_token()`函数中的`Log`语句注释了，不然会输出许多匹配正则表达式的信息。

PA1.3.1 监视点结构体

修改`nemu\include\monitor\watchpoint.h`

```
typedef struct watchpoint
{
  int NO; //监视点的序号
  struct watchpoint *next;
  char expr[32];
  uint32_t new_val;
  uint32_t old_val;
  uint8_t type;//0 is watchpoint, 1 is breakpoint，为后续的软件断点准备
} WP;

WP* new_wp();
void free_wp(WP *wp);
```

PA1.3.2 监视点池的管理

修改`nemu\src\monitor\debug\watchpoint.c`

```
WP* new_wp(){
    if(free_ == NULL) {
        printf("There is no more memory to set watchpoint. You should delete some watchpoints to free memory.\n");
        return NULL;
    }
    WP* p = free_;
    free_ = free_->next;
    p->next = head;
    head = p;
    return head;
}

void free_wp(WP *wp){
    if(head == NULL){
        printf("There is no watchpoint to free.\n");
        return ;
    }
    if(wp == head){
        head = head->next;
        wp->next = free_;
        free_ = wp;
        return ;
    }
    WP *p = head;
    while(p->next != wp) p = p->next;
    p->next = wp->next;
    wp->next = free_;
    free_ = wp;
}
```

PA1.3.3 监视点加入调试器

修改`nemu\src\monitor\debug\ui.c`

1. `w` 命令：根据给予的表达式 `expr` 设置一个新的监视点

```
在 ui.c 中声明
static int cmd_w(char *args);
static int cmd_d(char *args);

static int cmd_w(char *args){
    char *arg1 = strtok(NULL, " ");
    if(arg1 == NULL){
        printf("A parameter is missing!\n");
        return 0;
    }
    char *arg2 = strtok(NULL, " ");
    while(arg2 != NULL){
        strcat(arg1, arg2);
        arg2 = strtok(NULL, " ");
    }

    int NO = set_watchpoint(arg1);
    if(NO != -1){
        bool success = true;
        printf("Set watchpoint #%d\n", NO);
        printf("expr      = %s\n", args);
        printf("old value = 0x%08x\n", expr(args, &success));
    }
    return 0;
}
```

2. `d` 命令：根据给予的监视点编号 `NO` 删除该监视点：

```
static int cmd_d(char *args)
{
    char *arg = strtok(NULL, " ");
    if(arg == NULL){
        printf("A parameter is missing!\n");
        return 0;
    }
    int NO = atoi(args);
    if(NO < 0){
        printf("Invalid arguments for d!\n");
        return 0;
    }
    if(delete_watchpoint(NO)){
        printf("Watchpoint/breakpoint %d deleted\n", NO);
    }
    else{
        printf("Watchpoint/breakpoint %d not found\n", NO);
    }
    return 0;
}
```

3. `info w` 命令：显示当前所有监视点

```
static int cmd_info(char *args){
    char *arg = strtok(NULL, " ");
    if(arg == NULL){
        printf("A parameter is missing!\n");
        return 0;
    }
    if (strcmp(arg, "r") == 0){
        reg_display();
    }
    else if (strcmp(arg, "w") == 0) {
        list_watchpoint();
    }
    else{
        printf("Unknown command '%s'\n", arg);
    }
    return 0;
}
```

在`src/monitor/cpu-exec.c`的`cpu_exec`函数调用`scan_watchpoint`，并引入头文件`monitor/watchpoint.h`

```
void cpu_exec(uint64_t n)
{
    ...

    for (; n > 0; n--){
    /* Execute one instruction, including instruction fetch,
     * instruction decode, and the actual execution. */
        exec_wrapper(print_flag);

#ifdef DEBUG
        WP *p = scan_watchpoint();
        if (p){
            printf("Hit watchpoint %d at address %#010x\n", p->NO, cpu.eip);
            printf("expr      = %s\n", p->expr);
            printf("old value = 0x%08x\n", p->old_val);
            printf("new value = 0x%08x\n", p->new_val);
            printf("promgram paused\n");
            p->old_val = p->new_val;
            nemu_state = NEMU_STOP;
            return;
        }
#endif
    ...
    }
    ...
}
```

将 `cmd_w`、`cmd_d` 添加到命令列表：

```
static struct {
  char *name;
  char *description;
  int (*handler) (char *);
} cmd_table [] = {
    ...
    { "info", "Show information about registers with argument 'r' and show information about watchpoint with argument 'w'", cmd_info},
    { "w", "Usage: w [EXPR]\n" "    set watchpoint for the [EXPR].", cmd_w},
    { "d", "usage: d [N]\n" "    delete watchpoint whose id is N", cmd_d},
};
```

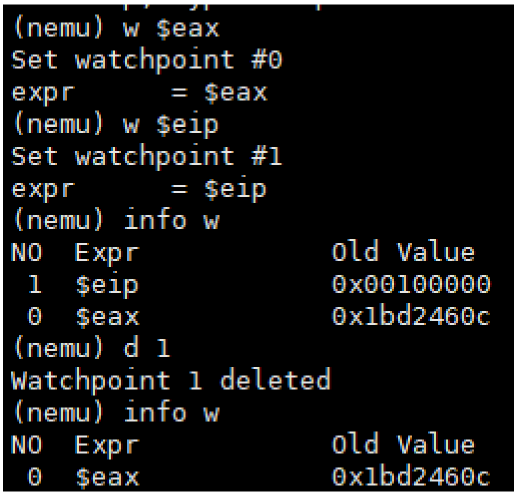
PA1.3.4 监视点主要功能

修改 `nemu\src\monitor\debug\watchpoint.c`：

```
在 watchpoint.c 中声明
int set_watchpoint(char *e);    //给予一个表达式e，构造以该表达式为监视目标的监视点，并返回编号
bool delete_watchpoint(int NO); //给予一个监视点编号，从已使用的监视点中归还该监视点到池中
void list_watchpoint(void);     //显示当前在使用状态中的监视点列表
WP* scan_watchpoint(void);      //扫描所有使用中的监视点，返回触发的监视点指针，若无触发返回NULL

int set_watchpoint(char *e){
    WP* wp = new_wp();
    if(wp == NULL) return -1;
    memset(wp->expr, 0, sizeof(wp->expr));
    strcpy(wp->expr, e);
    wp->type = 0;
    bool success = true;
    wp->old_val = expr(e, &success);
    if(!success){
        printf("set watchpoint failed. Please check your exprssion!\n");
        free_wp(wp);
        return -1;
    }
    return wp->NO;
}

bool delete_watchpoint(int NO)
{
    WP *wp = head;
    while (wp && wp->NO != NO){
        wp = wp->next;
    }
    if (wp){
        free_wp(wp);
        return true;
    }
    else  return false;
}

void list_watchpoint(void)
{
    if(head == NULL) {
        printf("There is no watchpoints!\n");
        return;
    }
    printf("NO  Expr            Old Value\n");
    WP *p = head;
    while (p) {
        if (p->type == 0) {
            printf("%2d  %-16s%#010x\n", p->NO, p->expr, p->old_val);
        }

        p = p->next;
    }
}

WP* scan_watchpoint(){
    WP *p = head;
    bool success;
    uint32_t new_value = 0;
    while(p){
        if(p->type == 0){
            success = true;
            new_value = expr(p->expr, &success);
            if(p->old_val != new_value){
                p->new_val = new_value;
                return p;
            }
        }
        p = p->next;
    }
    return NULL;
}
```

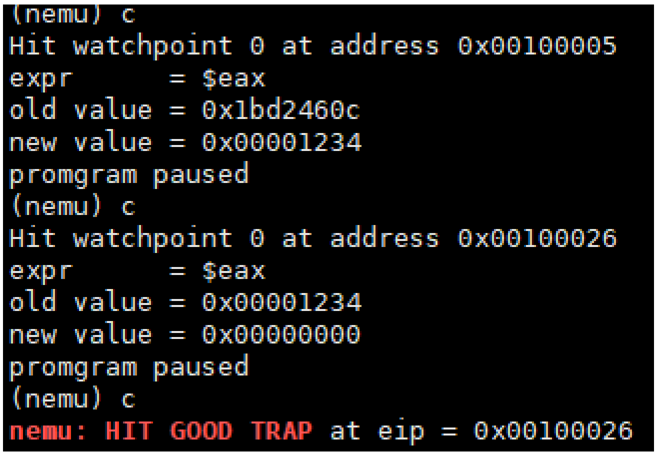
测试样例：

`w $eax`, `w $eip`, `info w`, `d 1`, `info w`



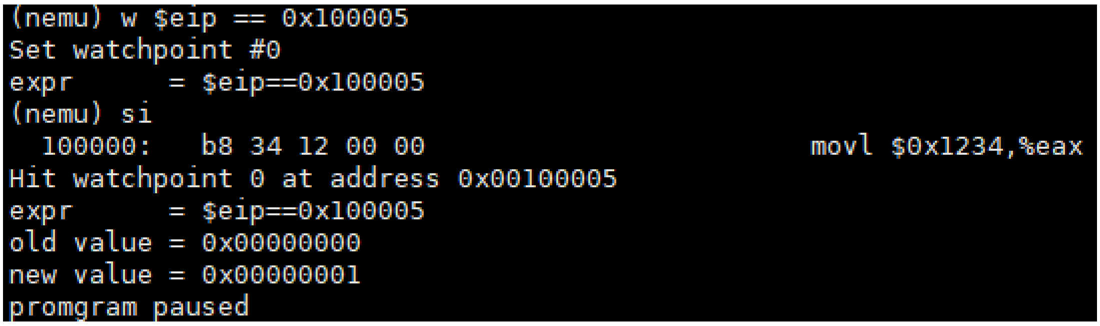
```
(nemu) w $eax
Set watchpoint #0
expr      = $eax
(nemu) w $eip
Set watchpoint #1
expr      = $eip
(nemu) info w
NO  Expr            Old Value
 1  $eip            0x00100000
 0  $eax            0x1bd2460c
(nemu) d 1
Watchpoint 1 deleted
(nemu) info w
NO  Expr            Old Value
 0  $eax            0x1bd2460c
```

`c`



```
(nemu) c
Hit watchpoint 0 at address 0x00100005
expr       = $eax
old value = 0x1bd2460c
new value = 0x00001234
promgram paused
(nemu) c
Hit watchpoint 0 at address 0x00100026
expr       = $eax
old value = 0x00001234
new value = 0x00000000
promgram paused
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100026
```

PA1.3.5 使用模拟断点



```
(nemu) w $eip == 0x100005
Set watchpoint #0
expr       = $eip==0x100005
(nemu) si
  100000:    b8 34 12 00 00                          movl $0x1234,%eax
Hit watchpoint 0 at address 0x00100005
expr       = $eip==0x100005
old value = 0x00000000
new value = 0x00000001
promgram paused
```

选做任务 实现软件断点

参考`http://www.voidcn.com/article/p-sdcuyscy-bme.html`和`https://eli.thegreenplace.net/2011/01/27/how-debuggers-work-part-2-breakpoints`

硬件中断和软件中断

硬件中断：通常是专用的电信号，附加了特殊的“响应电路”。该电路注意到中断的激活，并使CPU停止其当前执行，保存其状态，并跳转到该中断的处理程序所在的预定义地址。处理程序完成工作后，CPU从停止处恢复执行。

软件中断在原理上与之相似。

INT 3 指令

INT 3指令生成一个特殊的一字节操作码（CC），用于调用调试异常处理程序。（此一个字节的形式很有价值，因为它可用于用断点替换任何指令的第一个字节，包括其他一个字节的指令，而不会覆盖其他代码）。

要在跟踪的进程中的某个目标地址处设置断点，调试器将执行以下操作：

1. 记住存储在目标地址中的数据
2. 将目标地址的第一个字节替换为int 3指令（指令码均替换为`0xcc`）

然后，当调试器要求OS运行该进程时，该进程将运行并最终到达`int3`处，它将停止并由OS发送一个信号。这是调试器再次进入的地方，接收到其子级（或跟踪的进程）已停止的信号。然后：

1. 将目标地址处的int 3指令替换为原始指令
2. 将跟踪的进程的指令指针减 1 。这是必要的，因为该指令指针现在指向`INT 3`的下一条指令。
3. 允许用户以某种方式与流程进行交互，因为流程仍在所需的目标地址处暂停。这是调试器允许您窥视变量值，调用堆栈等的部分。
4. 当用户希望继续运行时，调试器将负责将断点放回目标地址（因为在步骤1中已将其删除），除非用户要求取消断点。（可能该处是循环）

断点异常（INT 3）属于陷阱类异常，当CPU产生异常时，其程序指针是指向导致异常的下一条指令的，因为导致该异常的指令已经执行完成。但是，现在我们观察到的结果却是指向导致异常的这条指令的。这是为什么呢？简单地说，是操作系统为了支持调试对程序指针做了调整。

先完成第一个任务：程序开始运行时（`c` 命令或 `si x` 命令），根据当前断点列表中所存储信息，将列表中的地址上的指令码均替换为 `0xcc`，并保存该地址上实际字节；

```
在 watchpoint.c 中引入 memory/memory.h 头文件
int set_breakpoint(char *e)
{
    WP* wp = new_wp();
    if(wp == NULL) return -1;
    memset(wp->expr, 0, sizeof(wp->expr));
    wp->type = 1; //断点
    bool success = true;

    wp->old_val = expr(e, &success); //old_value保存 设置断点的地址（程序执行到该地址处中断）
    if(!success){
        printf("set breakpoint failed. Please check your exprssion!\n");
        free_wp(wp);
        return -1;
    }
    wp->new_val = 0;
    /*初始为0；
    当程序到达断点之后（注意是之后），此时需要将eip - 1。这是必要的，因为该eip现在指向断点的后一个指令，并将new_val设置为2
    当值为2的时候，要恢复断点（可能该处是循环等情况），再将其值设置为0
    */
    *wp->expr = *(char *)guest_to_host(wp->old_val);//expr保存原来的操作码

    /*
    #define guest_to_host(p) ((void *)(pmem + (unsigned)p))
    convert the guest physical address in the guest program to host virtual address in NEMU
    */
    *(char *)guest_to_host(wp->old_val) = 0xcc;//从内存中修改断点地址对应的操作码（机器指令中，第一个字节就是操作码）为0xcc

    return wp->NO;
}

static int cmd_b(char *args)
{
    char *arg1 = strtok(NULL, " ");
    if(arg1 == NULL){
        printf("A parameter is missing!\n");
        return 0;
    }
    char *arg2 = strtok(NULL, " ");
    while(arg2 != NULL){
        strcat(arg1, arg2);
        arg2 = strtok(NULL, " ");
    }

    int NO = set_breakpoint(arg1);
    if(NO != -1){
        bool success = true;
        uint32_t address = expr(args, &success);
        printf("Set breakpoint #%d\n", NO);
        printf("address    = %#010x\n", address);
        printf("old value = 0x%08x\n", vaddr_read(address, 4));
    }
    return 0;
}
```

加入到命令列表

```
static struct {
  char *name;
  char *description;
  int (*handler) (char *);
} cmd_table [] = {
    ...
    {"b", "b EXPR:set a breakpoint for eip as the value of EXPR", cmd_b},
};
```

`d`命令只需修改一下`printf`语句即可:

```
static int cmd_d(char *args){
   ...
   printf("Watchpoint/breakpoint %d deleted\n", NO);
}
```

再完成第二个任务：程序命中 `0xcc` 指令码时，`NEMU` 执行 `int3` 的指令处理函数。处理函数的逻辑为：将原有指令替换回所有被 `int3` 所占据的位置，并设置 `nemu_state` 变量为 `NEMU_STOP` 。

设置`0xcc`指令码的指令处理函数，在`nemu/src/cpu/exec/exec.c`中（从`PA2`得知）

（先了解一下`opcode_entry`的定义）

```
typedef struct {
  DHelper decode;// typedef void (*DHelper) (vaddr_t *);
  EHelper execute;// typedef void (*EHelper) (vaddr_t *);
  int width;
  /*译码函数，执行函数，以及操作数宽度*/
  /*通过查表的方式得知这条指令的操作数和操作码。这个过程叫译码*/
} opcode_entry;
```

找到对应的译码查找表数组`opcode_table`，这一张表通过操作码 `opcode` 来索引，每一个 `opcode` 对应相应指令的译码函数, 执行函数, 以及操作数宽度。

找到操作码`0xcc`的位置，设置执行函数（程序命中 `0xcc` 指令码时，`NEMU` 执行 `int3` 的指令处理函数）

```
opcode_entry opcode_table [512] = {
    ...
/* 0xcc */  EX(int3), EMPTY, EMPTY, EMPTY,

    /*
    该文件中有以下的宏定义
    #define EXW(ex, w)          {NULL, concat(exec_, ex), w}
    #define EX(ex)              EXW(ex, 0)
    #define EMPTY               EX(inv)

    所以EX(int3) == EXW(int3, w) == {NULL, concat(exec_, int3), w}
                                 == {NULL, exec_int3, w}
    */

    ...
}
```

因为还有如下的宏定义：

```
#define make_EHelper(name) void concat(exec_, name) (vaddr_t *eip)
```

所以，`exec_int3()` 函数得通过宏 `make_EHelper` 来定义

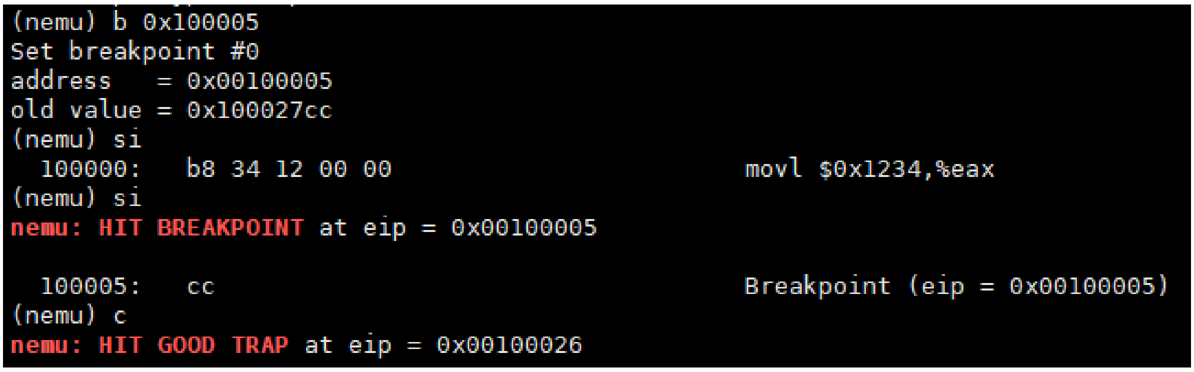
```
在src/cpu/exec/all-instr.h 中声明该函数！！
引用 monitor/watchpoint.h 、monitor/monitor.h 头文件
在src/cpu/exec/system.c中定义

make_EHelper(int3) {
    /*int3 指令的执行函数
    根据上方的宏定义，等价于 exec_int3(vaddr_t *eip){}
    */
    print_asm("Breakpoint (eip = %0#10x)", cpu.eip);
    printf("\33[1;31mnemu: HIT BREAKPOINT\33[0m at eip = %0#10x\n\n", cpu.eip);
    //上面两个输出仿照nemu/src/cpu/exec/special.c 的 make_EHelper(nemu_trap) 函数
    recover_int3();
    nemu_state = NEMU_STOP;
}
```

```
声明在 watchpoint.h 中
引入 cpu/reg.h
定义 在watchpoint.c 因为要head指针
void recover_int3() {
    WP *p = head;
    while (p) {
        if (p->type == 1 && p->old_val == cpu.eip){//到了断点位置
            *(char *)guest_to_host(p->old_val) = *p->expr;//恢复
            p->new_val = 1;
            break;
        }
        p = p->next;
    }
}
```

修改`scan_watchpoint`函数

```
WP* scan_watchpoint(){
  WP *p = head;
  bool success;
  uint32_t new_value = 0;
  while(p){
    if(p->type == 0){
      success = true;
      new_value = expr(p->expr, &success);
      if(p->old_val != new_value){
        p->new_val = new_value;
        return p;
      }
    }
    else{
      if(p->new_val == 1){//eip回退1
        cpu.eip -= 1;
        p->new_val = 2;
      }
      else if(p->new_val == 2){
        *p->expr = *(char *)guest_to_host(p->old_val);
        *(char *)guest_to_host(p->old_val) = 0xcc;
        p->new_val = 0;
      }
    }

    p = p->next;
  }
  return NULL;
}
```

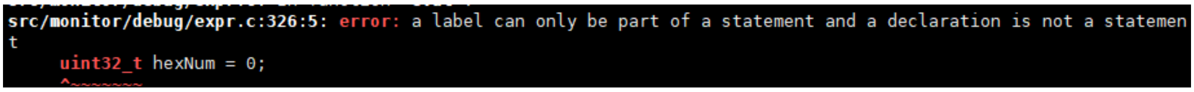
-1是因为还要执行断点处的指令



```
(nemu) b 0x100005
Set breakpoint #0
address   = 0x00100005
old value = 0x100027cc
(nemu) si
  100000:   b8 34 12 00 00                              movl $0x1234,%eax
(nemu) si
nemu: HIT BREAKPOINT at eip = 0x00100005

  100005:   cc                                          Breakpoint (eip = 0x00100005)
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100026
```

遇到的问题及解决办法

1. 在switch的case中声明变量出现的错误：

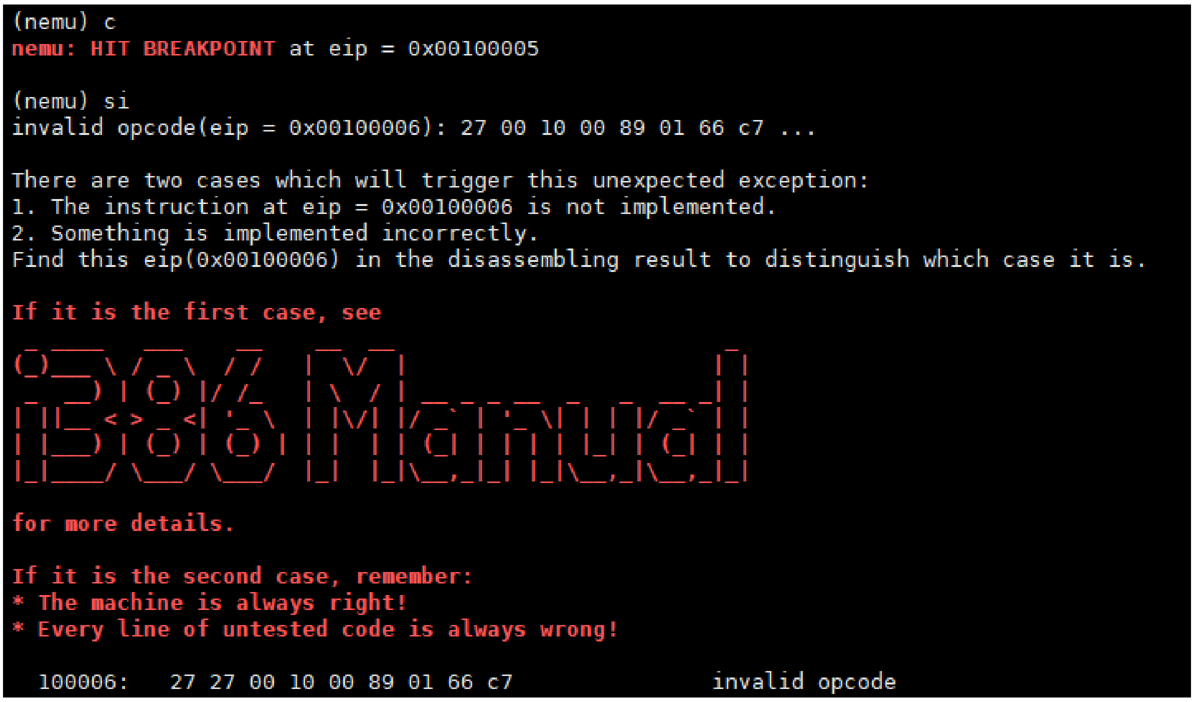


```
src/monitor/debug/expr.c:326:5: error: a label can only be part of a statement and a declaration is not a statemen
t
     uint32_t hexNum = 0;
     ^~~~~~~~
```

由于`switch`的几个case语句在同一个作用域（因为`case`语句只是标签，它们共属于一个`swtich`语句块），所以如果在某个`case`下面声明变量的话，对象的作用域是在俩个花括号之间 也就是整个`switch`语句，其他的`case`语句也能看到，这样的话就可能导致错误。

解决方案：在`case`语句后边加大括号

2. `eip=0x00100006`处的指令未实现。



```
(nemu) c
nemu: HIT BREAKPOINT at eip = 0x00100005

(nemu) si
invalid opcode(eip = 0x00100006): 27 00 10 00 89 01 66 c7 ...

There are two cases which will trigger this unexpected exception:
1. The instruction at eip = 0x00100006 is not implemented.
2. Something is implemented incorrectly.
Find this eip(0x00100006) 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!

  100006:   27 27 00 10 00 89 01 66 c7                 invalid opcode
```

原因：在检测到断点后，`eip`指向断点指令的后一条指令，不应该从该处继续进行程序。

解决方法：在检测到断点之后，将`eip`回退1。

3. 问题：`gdb` 中添加监视点`w i`时，显示`No symbol “***” in current context`

原因：`gcc`与`gdb`的版本不匹配

解决方法：下载`gdb-8.3`

实验心得

这次实验花费了好长时间，时间主要花在了表达式的括号匹配和软件断点模块。

括号匹配并不是非常难，但是要考虑各种情况，不仅仅是简单的检测左括号是否有相应右括号匹配。

最最耗时间的就是软件断点模块了，这个模块涉及到一些更底层的代码，例如如何找到指令码、如何添加执行函数、如何修改指令码等，除了讲义外还要查阅许多信息。

最后成功的做出了所有任务，还是比较有成就感的。

其他备注

无