

# Report for ICS PA1

by 项远方 (221300012)

## Part I: Reading Source Code

Met problem when using command `make gdb` to test the source code. Solution: installed `ccache` and modified relevant environment variable paths to ensure `make` compile `nemu` programs through `ccache`.

```
(gdb) l
No symbol table is loaded. Use the "file" command.
(gdb) file nemu-main.c
"/home/yfxiang/njupa/ics2023/nemu/src/nemu-main.c": not in executable format: file format not recognized
(gdb) q
yfxiang@Lenovo-Yoga-3-11:~/njupa/ics2023/nemu/src$ cd ..
yfxiang@Lenovo-Yoga-3-11:~/njupa/ics2023/nemu$ make gdb
+ LD /home/yfxiang/njupa/ics2023/nemu/build/riscv32-nemu-interpret
gdb -s /home/yfxiang/njupa/ics2023/nemu/build/riscv32-nemu-interpret --args /home/yfxiang/njupa/ics2023/nemu/build/riscv32-nemu-interpret --log=/home/yfxiang/njupa/ics2023/nemu/build/nemu-log.txt
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /home/yfxiang/njupa/ics2023/nemu/build/riscv32-nemu-interpret...
(gdb) q
yfxiang@Lenovo-Yoga-3-11:~/njupa/ics2023/nemu$ which gcc
/usr/bin/gcc
yfxiang@Lenovo-Yoga-3-11:~/njupa/ics2023/nemu$
```

before installing `ccache`, using command `make gdb` would trigger an error like this.

Mainly focused on codes in `/src` directory. Came up with and solved several questions, for example, in `src/engine/interpreter/init.c` called function `cpu-exce()` with an argument `-1`, which receives argument of `unsigned int`. The reason is that `-1` here is interpreted as max uint value, i.e. 4294967295 here. This had deepened my understanding of computer data representation.

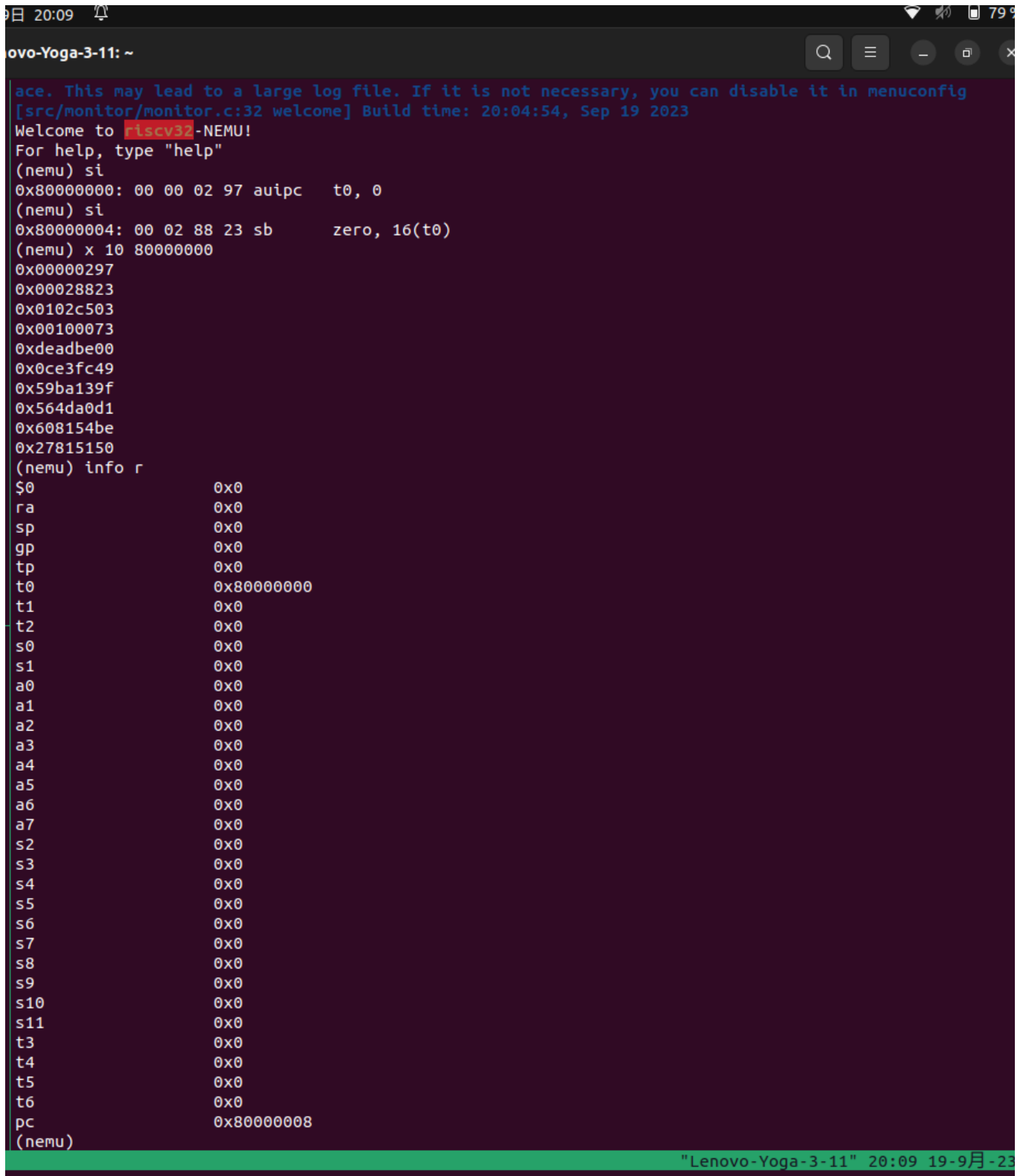
Used several `gdb` features to help understanding the source code, e.g. stepping into function definitions in the order of program running.

### exercises:

1.  $(0, x, x) \rightarrow (1, 0, x) \rightarrow (2, 0, 0) \rightarrow (3, 0, 1) \rightarrow (4, 1, 1) \rightarrow (5, 1, 2) \rightarrow (6, 3, 2) \rightarrow (7, 3, 3) \rightarrow (8, 6, 3) \rightarrow (9, 6, 4) \rightarrow \dots \rightarrow (199, 4851, 99) \rightarrow (200, 4950, 99) \rightarrow (201, 4950, 100) \rightarrow (202, 5050, 100)$
2. The error is caused by incorrect value of `nemu_state` when quit NEMU with command `q`. Can be solved by modifying `nemu_state` in definition of `cmd_q()`.

## Part II: Basic SDB Commands

Completed the implementation of single step execute, print registers information and scan memory by comands `si`, `info r` and `x`.



```

ovo-Yoga-3-11: ~
ace. This may lead to a large log file. If it is not necessary, you can disable it in menuconfig
[src/monitor/monitor.c:32 welcome] Build time: 20:04:54, Sep 19 2023
Welcome to riscv32-NEMU!
For help, type "help"
(nemu) si
0x80000000: 00 00 02 97 auipc    t0, 0
(nemu) si
0x80000004: 00 02 88 23 sb      zero, 16(t0)
(nemu) x 10 80000000
0x00000297
0x00028823
0x0102c503
0x00100073
0xdeadbe00
0x0ce3fc49
0x59ba139f
0x564da0d1
0x608154be
0x27815150
(nemu) info r
$0          0x0
ra          0x0
sp          0x0
gp          0x0
tp          0x0
t0          0x80000000
t1          0x0
t2          0x0
s0          0x0
s1          0x0
a0          0x0
a1          0x0
a2          0x0
a3          0x0
a4          0x0
a5          0x0
a6          0x0
a7          0x0
s2          0x0
s3          0x0
s4          0x0
s5          0x0
s6          0x0
s7          0x0
s8          0x0
s9          0x0
s10         0x0
s11         0x0
t3          0x0
t4          0x0
t5          0x0
t6          0x0
pc          0x80000008
(nemu)

```

## Part III: Expression Evaluation

Implemented evaluating a mathematical expression through command `p`. Includes matching patterns with regular expression, analyzing tokens and evaluating expressions using its recursion definition.

Also implemented test for expr-eval through generating expressions.

Major bug: when testing with generated expression, long expressions with many brackets would cause `segmentation fault (core dumped)`. Locating this bug spend more than 5 hours, using `printf`, `gdb` and other methods. (Also wasted time for unfamiliar with `gdb`'s using. I didn't know command `run <filename>` can pass argument to `gdb`, and didn't get answer from STFW due to improper key word. Thanks my friend `yyf` for assistance when STFWing.)

The bug is caused by a constant number. The max length of generated expr is `65535` so I modified **almost** constant values of `expr len`. But I forgot one, in function `check_parentheses()`. This bug told me a lesson, always use `static const` variable instead of nubers.

## Part IV: Watchpoint

Enabled more features for `expr-eval`, can evaluate expressions including and/or, `eq/neq`, dereference, register name and negative number (but unsigned).

Implemented `adding` and `removeing` of watchpoints using two linked lists, and iterate through all watchpoints once after `cpu-exec` is called. Finally added a menu config macro `CONFIG_WATCHPOINT` to enable or disable watchpoint features in config menu.

---

### Met some problem.

When implementing `expr.c`, I tried to use regular expression to match memory address dereference (`*<expr>`) and negative number (`-<expr>`), but triggered a `panic`. I STFWed the regex rule and ensured its correctness with some online tool. I RTFSCed the source code regarding regex init and check, and noticed that some arguments passed to regex library functions, which seems like enabled some configurations about the function.

Finally I gave up to match dereference with regex. (Is it banned to use some certain regex in PA's sourcecode?)

```
[src/utils/log.c:28 init_log] Log is written to /home/yfxiang/njupa/ics2023/nemu/build/nemu-log.txt
[src/memory/paddr.c:56 init_mem] physical memory area [0x80000000, 0x87ffffff]
[src/monitor/monitor.c:49 load_img] No image is given. Use the default build-in image.
regex compilation failed: Invalid preceding regular expression
(?<![0-9|\)]\))*
$0          0x0
ra          0x0
sp          0x0
gp          0x0
tp          0x0
t0          0x0
t1          0x0
t2          0x0
s0          0x0
```

正则表达式在线测试

生成代码

匹配数字

匹配字母

匹配中文

测试实例

可视化图

/

(?<![0-9|\)]\))\*

/g

修饰符

语法参考

correct example: `*(0x80000000) + *a-*t3**2`

wrong example: `1*9-(2+9)*21`

correct example: `(0x80000000) + *a-*t3**2`

wrong example: `1*9-(2+9)*21`

Part V Exercises

- 1. 5.1 Segment Translation - 5.1.3 Selectors
- 2. Might more than 70 hours. SDB might save more than 50 hours.
- 3. chap1.3, chap2.2, chap15, chap7
- 4. c file: 21534 lines, h file: 2713 lines. Comparing to pa0, added 439 lines in c and 7 lines in h files. 18705 lines in total without blanks.
- 5. -wall: enable all kinds of warnings, -werror: make all warnings into errors. They are used in PA to make us become better programmers (I suppose?)