南京航空航天大学《计算机组成原理Ⅱ课程设计》报告

- 姓名：邵震哲
- 班级：1620204
- 学号：162020130
- 报告阶段：PA1.1
- 完成日期：2022.3.24
- 本次实验，我完成了所有内容。

目录

思考题

1.存放的是什么?

存放的是下一条要执行的指令的地址。如果存放指令，每条指令的长度都不一样，难以使用，每次存放都要计算指令长度，否则会有错误。而地址是一串长度相同的数字，很容易就能存放进去，而且用地址取指的方式便于执行跳转指令。

2.贵圈真乱

```
+----------------------------------+
|      "Hello World" program       |
+----------------------------------+
|     Emulated x86 instruction     |
+----------------------------------+
|               NEMU               |
+----------------------------------+
|             GNU/Linux            |
+----------------------------------+
| VirtualBox (Simulated Hardware)  |
+----------------------------------+
|      Host Operating System       |
+----------------------------------+
|        Computer hardware         |
+----------------------------------+
```

3.虚拟机和模拟器的区别

虚拟机指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。在实体计算机中能够完成的工作在虚拟机中都能够实现。在计算机中创建虚拟机时，需要将实体机的部分硬盘和内存容量作为虚拟机的硬盘和内存容量。每个虚拟机都有独立的CMOS、硬盘和操作系统，可以像使用实体机一样对虚拟机进行操作。

模拟器只是虚拟机概念的子集，只是用代码模拟了简化的x86架构，只能完成虚拟机的部分功能。

4.从哪开始阅读代码呢

从`main()`函数开始阅读，而`main()`在`main.c`文件中

5.究竟要执行多久

`/nemu/src/monitor/cpu-exec.c` 发现`cpu_exec()`的参数是无符号整型，因此传入-1相当于传入了无符号最大的数字，那么函数里的for循环可以执行最大次数的循环

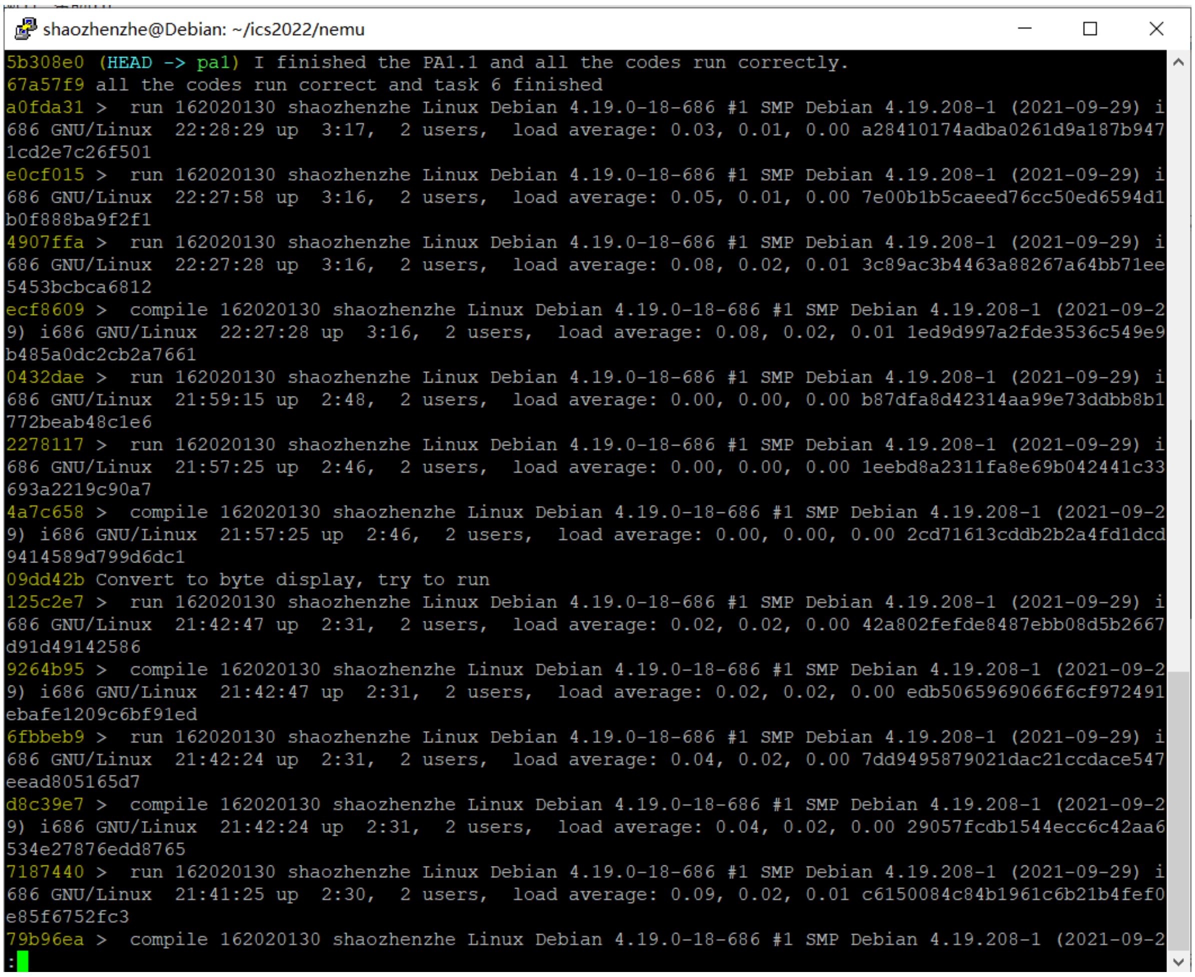
6.谁来指示程序的结束?

`main`函数执行完之后还要去执行一些诸如释放空间之类的操作，并且全局对象的析构函数和用`atexit`注册的函数也会在`main`函数之后执行。`main`函数结束时会隐式的调用`exit()`函数，运行时会执行由`atexit()`函数登记的函数，做一些清理，刷新所有输出流，关闭所有打开的流。所以应该是由`exit()`指示结束

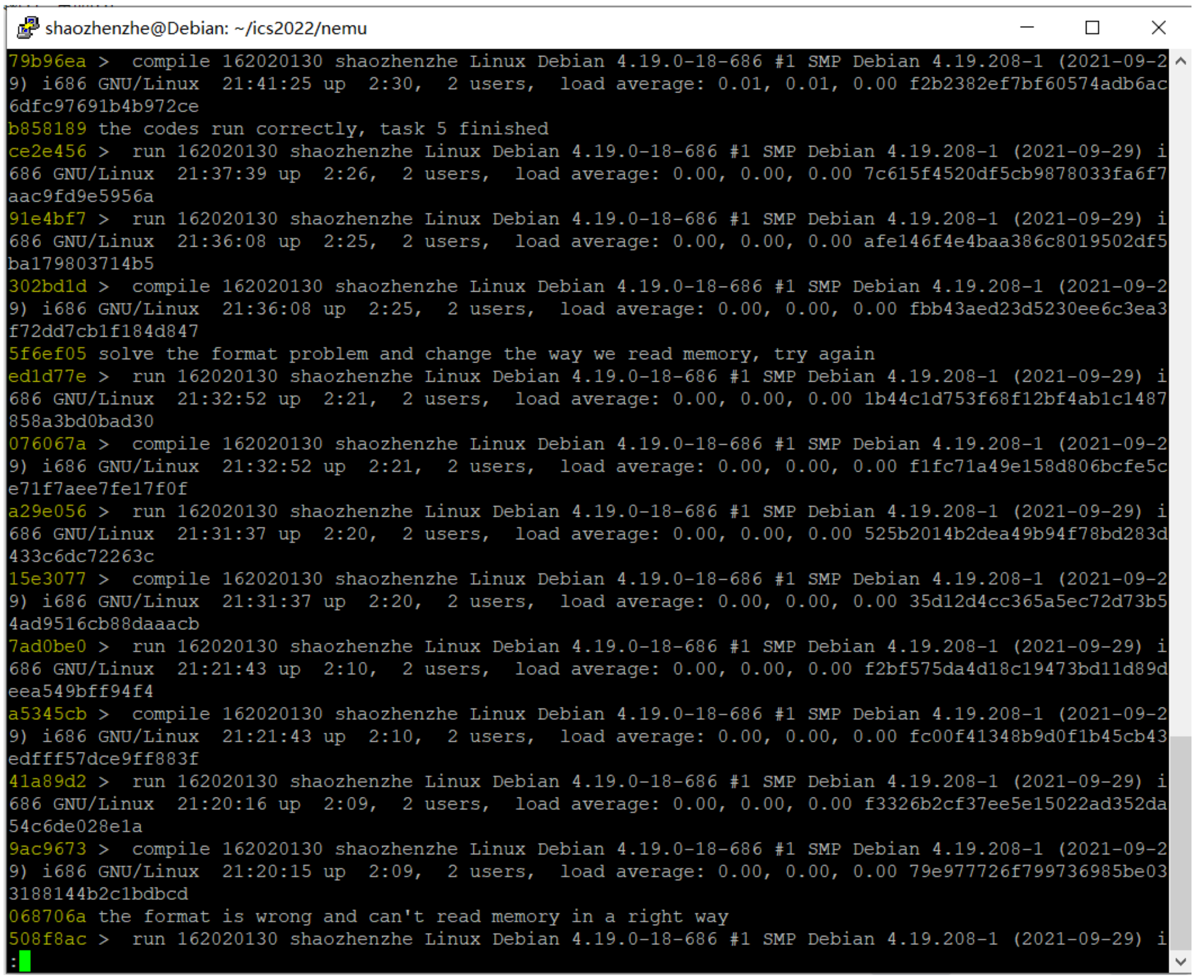
7.为什么会这样?

计算机系统中内存是以字节为单位进行编址的，采用小端存储模式，在寄存器中，一个字（4字节）的表示是右边应该属于低位，左边属于高位，所以4字节数输出时右边是低位，1字节输出时，低位先输出，因此根据输出的顺序低位在左边

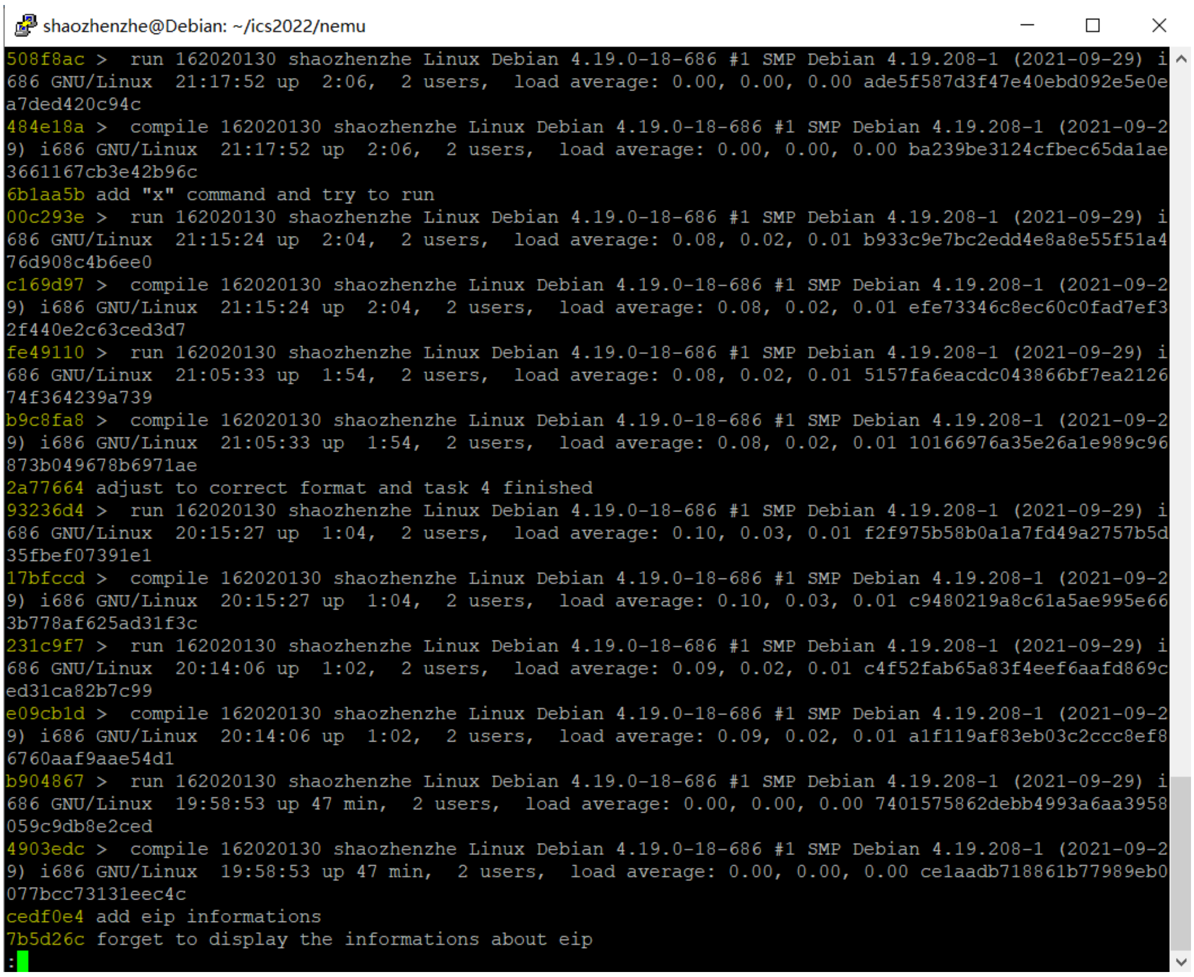
8.Git Log截图，在 pa1 分支下使用命令 git log --oneline 并截图，尽量完整。


shaozhenzhe@Debian: ~/ics2022/nemu

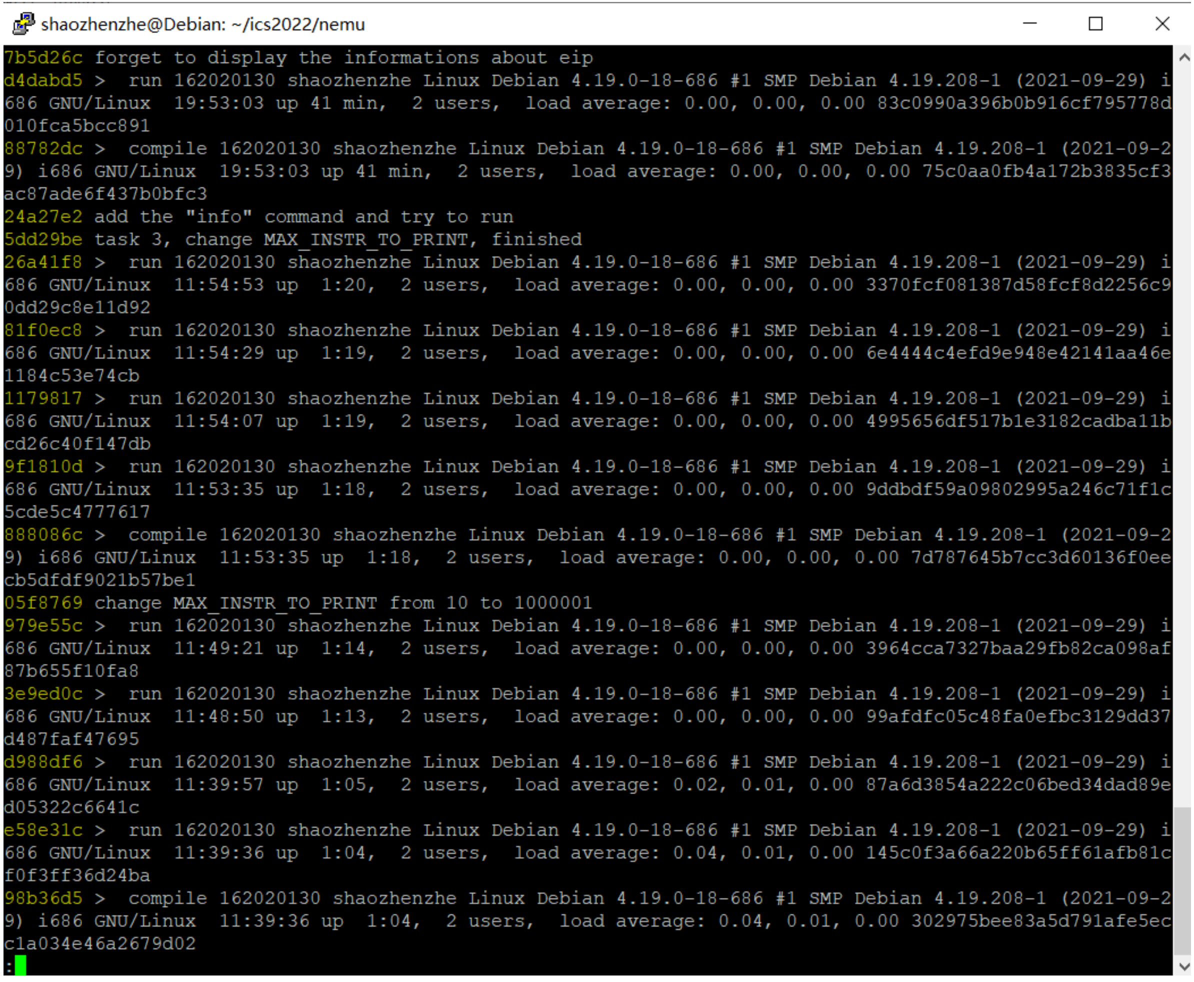
```
5b308e0 (HEAD -> pa1) I finished the PA1.1 and all the codes run correctly.
67a57f9 all the codes run correct and task 6 finished
a0fda31 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  22:28:29 up  3:17,  2 users,  load average: 0.03, 0.01, 0.00 a28410174adba0261d9a187b947
1cd2e7c26f501
e0cf015 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  22:27:58 up  3:16,  2 users,  load average: 0.05, 0.01, 0.00 7e00b1b5caeed76cc50ed6594d1
b0f888ba9f2f1
4907ffa >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  22:27:28 up  3:16,  2 users,  load average: 0.08, 0.02, 0.01 3c89ac3b4463a88267a64bb71ee
5453bcbca6812
ecf8609 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  22:27:28 up  3:16,  2 users,  load average: 0.08, 0.02, 0.01 1ed9d997a2fde3536c549e9
b485a0dc2cb2a7661
0432dae >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:59:15 up  2:48,  2 users,  load average: 0.00, 0.00, 0.00 b87dfa8d42314aa99e73ddbb8b1
772beab48c1e6
2278117 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:57:25 up  2:46,  2 users,  load average: 0.00, 0.00, 0.00 1eebd8a2311fa8e69b042441c33
693a2219c90a7
4a7c658 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:57:25 up  2:46,  2 users,  load average: 0.00, 0.00, 0.00 2cd71613cddb2b2a4fd1dcd
9414589d799d6dc1
09dd42b Convert to byte display, try to run
125c2e7 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:42:47 up  2:31,  2 users,  load average: 0.02, 0.02, 0.00 42a802fefde8487ebb08d5b2667
d91d49142586
9264b95 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:42:47 up  2:31,  2 users,  load average: 0.02, 0.02, 0.00 edb5065969066f6cf972491
ebafe1209c6bf91ed
6fbbeb9 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:42:24 up  2:31,  2 users,  load average: 0.04, 0.02, 0.00 7dd9495879021dac21ccdace547
eead805165d7
d8c39e7 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:42:24 up  2:31,  2 users,  load average: 0.04, 0.02, 0.00 29057fcdb1544ecc6c42aa6
534e27876edd8765
7187440 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:41:25 up  2:30,  2 users,  load average: 0.09, 0.02, 0.01 c6150084c84b1961c6b21b4fef0
e85f6752fc3
79b96ea >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
:
```


shaozhenzhe@Debian: ~/ics2022/nemu

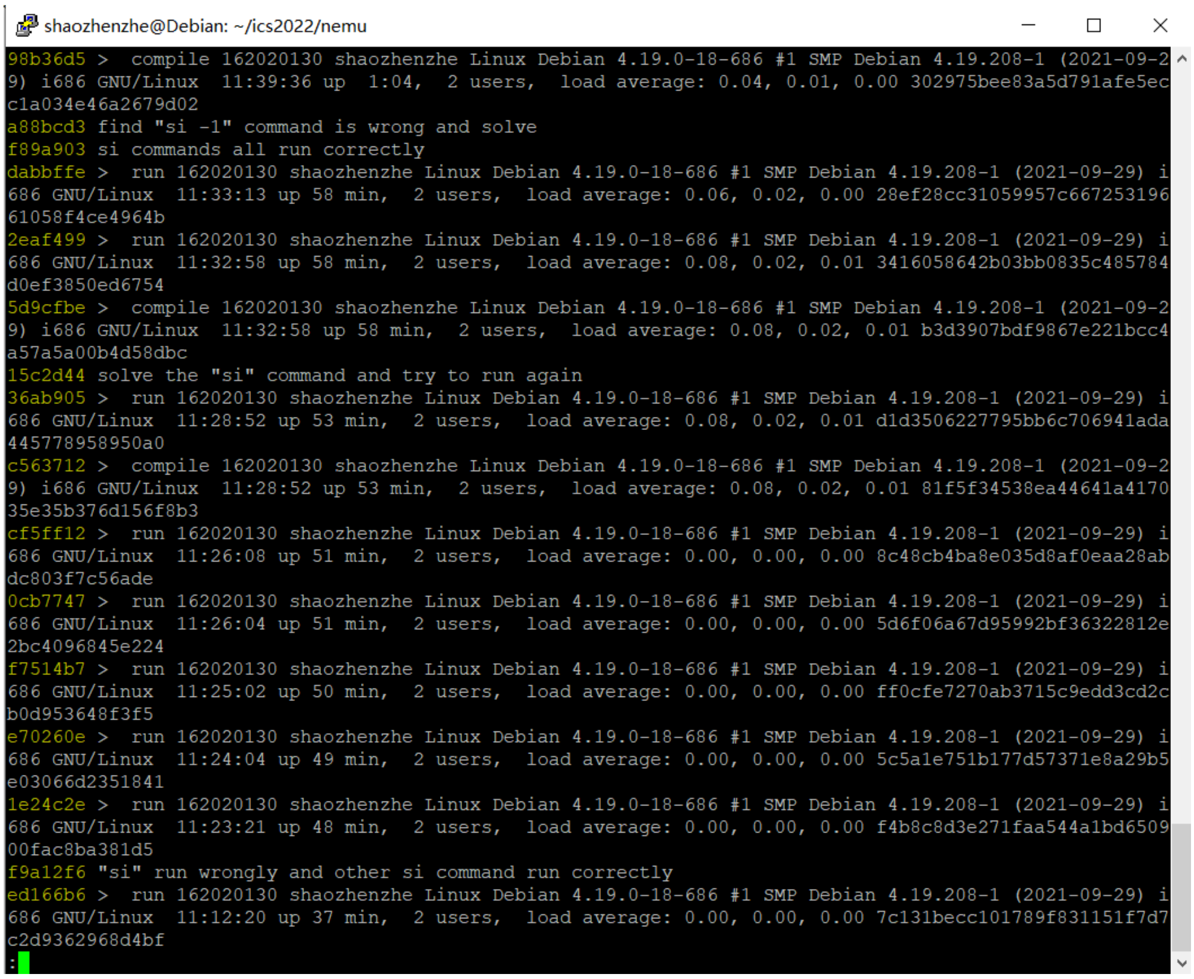
```
79b96ea >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:41:25 up  2:30,  2 users,  load average: 0.01, 0.01, 0.00 f2b2382ef7bf60574adb6ac
6dfc97691b4b972ce
b858189 the codes run correctly, task 5 finished
ce2e456 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:37:39 up  2:26,  2 users,  load average: 0.00, 0.00, 0.00 7c615f4520df5cb9878033fa6f7
aac9fd9e5956a
91e4bf7 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:36:08 up  2:25,  2 users,  load average: 0.00, 0.00, 0.00 afe146f4e4baa386c8019502df5
ba179803714b5
302bd1d >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:36:08 up  2:25,  2 users,  load average: 0.00, 0.00, 0.00 fbb43aed23d5230ee6c3ea3
f72dd7cb1f184d847
5f6ef05 solve the format problem and change the way we read memory, try again
ed1d77e >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:32:52 up  2:21,  2 users,  load average: 0.00, 0.00, 0.00 1b44c1d753f68f12bf4ab1c1487
858a3bd0bad30
076067a >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:32:52 up  2:21,  2 users,  load average: 0.00, 0.00, 0.00 f1fc71a49e158d806bcfe5c
e71f7aee7fe17f0f
a29e056 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:31:37 up  2:20,  2 users,  load average: 0.00, 0.00, 0.00 525b2014b2dea49b94f78bd283d
433c6dc72263c
15e3077 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:31:37 up  2:20,  2 users,  load average: 0.00, 0.00, 0.00 35d12d4cc365a5ec72d73b5
4ad9516cb88daaacb
7ad0be0 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:21:43 up  2:10,  2 users,  load average: 0.00, 0.00, 0.00 f2bf575da4d18c19473bd11d89d
eea549bff94f4
a5345cb >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:21:43 up  2:10,  2 users,  load average: 0.00, 0.00, 0.00 fc00f41348b9d0f1b45cb43
edfff57dce9ff883f
41a89d2 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:20:16 up  2:09,  2 users,  load average: 0.00, 0.00, 0.00 f3326b2cf37ee5e15022ad352da
54c6de028e1a
9ac9673 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:20:15 up  2:09,  2 users,  load average: 0.00, 0.00, 0.00 79e977726f799736985be03
3188144b2c1bdbcd
068706a the format is wrong and can't read memory in a right way
508f8ac >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
:
```


shaozhenzhe@Debian: ~/ics2022/nemu

```
508f8ac >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:17:52 up  2:06,  2 users,  load average: 0.00, 0.00, 0.00 ade5f587d3f47e40ebd092e5e0e
a7ded420c94c
484e18a >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:17:52 up  2:06,  2 users,  load average: 0.00, 0.00, 0.00 ba239be3124cfbec65da1ae
3661167cb3e42b96c
6b1aa5b add "x" command and try to run
00c293e >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:15:24 up  2:04,  2 users,  load average: 0.08, 0.02, 0.01 b933c9e7bc2edd4e8a8e55f51a4
76d908c4b6ee0
c169d97 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:15:24 up  2:04,  2 users,  load average: 0.08, 0.02, 0.01 efe73346c8ec60c0fad7ef3
2f440e2c63ced3d7
fe49110 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  21:05:33 up  1:54,  2 users,  load average: 0.08, 0.02, 0.01 5157fa6eacdc043866bf7ea2126
74f364239a739
b9c8fa8 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  21:05:33 up  1:54,  2 users,  load average: 0.08, 0.02, 0.01 10166976a35e26a1e989c96
873b049678b6971ae
2a77664 adjust to correct format and task 4 finished
93236d4 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  20:15:27 up  1:04,  2 users,  load average: 0.10, 0.03, 0.01 f2f975b58b0a1a7fd49a2757b5d
35fbef07391e1
17bfccd >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  20:15:27 up  1:04,  2 users,  load average: 0.10, 0.03, 0.01 c9480219a8c61a5ae995e66
3b778af625ad31f3c
231c9f7 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  20:14:06 up  1:02,  2 users,  load average: 0.09, 0.02, 0.01 c4f52fab65a83f4eef6aafd869c
ed31ca82b7c99
e09cb1d >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  20:14:06 up  1:02,  2 users,  load average: 0.09, 0.02, 0.01 a1f119af83eb03c2ccc8ef8
6760aaf9aae54d1
b904867 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:58:53 up 47 min,  2 users,  load average: 0.00, 0.00, 0.00 7401575862debb4993a6aa3958
059c9db8e2ced
4903edc >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  19:58:53 up 47 min,  2 users,  load average: 0.00, 0.00, 0.00 ce1aadb718861b77989eb0
077bcc73131eec4c
cedf0e4 add eip informations
7b5d26c forget to display the informations about eip
:
```

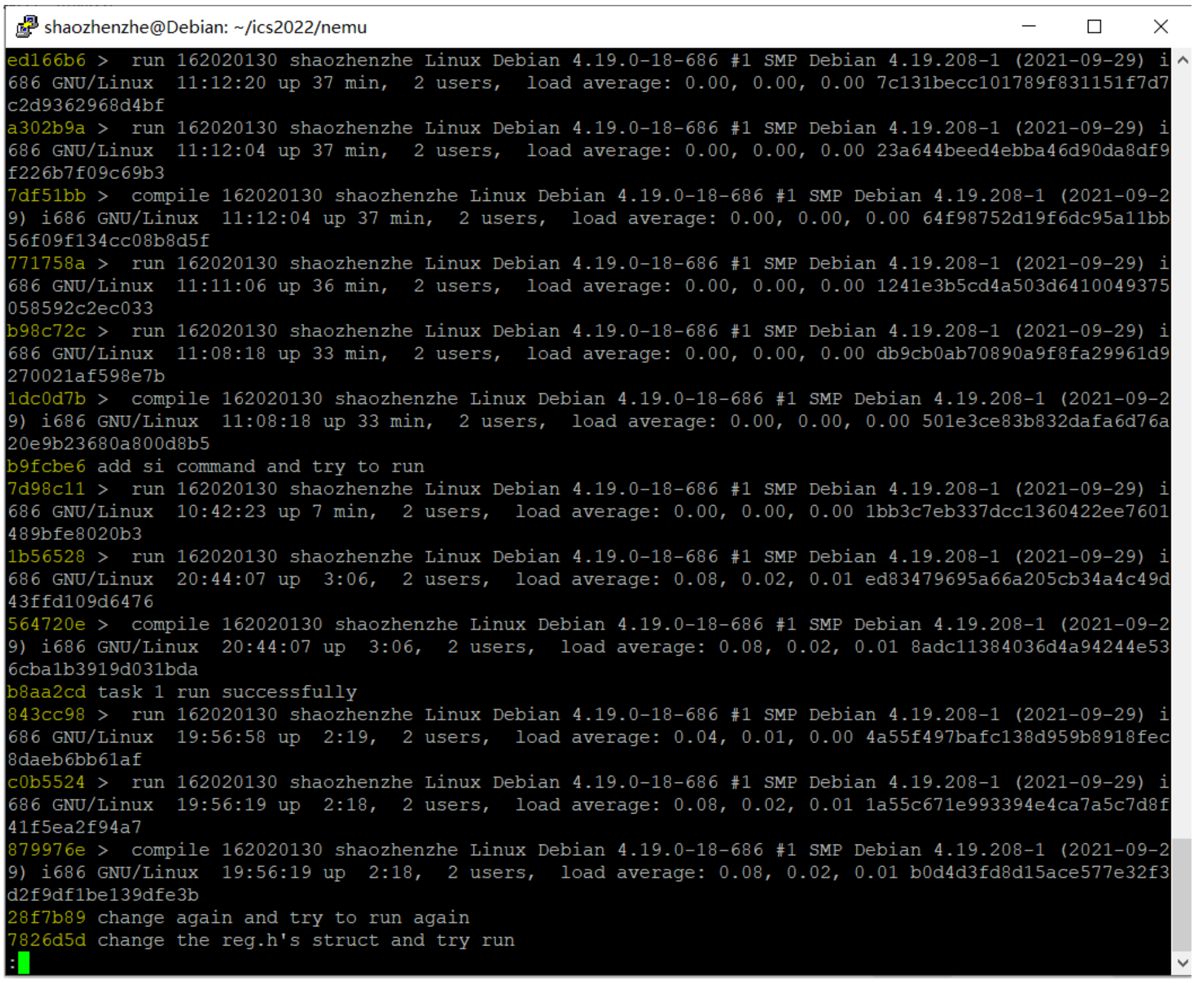

shaozhenzhe@Debian: ~/ics2022/nemu

```
7b5d26c forget to display the informations about eip
d4dabd5 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:53:03 up 41 min,  2 users,  load average: 0.00, 0.00, 0.00 83c0990a396b0b916cf795778d
010fca5bcc891
88782dc >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  19:53:03 up 41 min,  2 users,  load average: 0.00, 0.00, 0.00 75c0aa0fb4a172b3835cf3
ac87ade6f437b0bfc3
24a27e2 add the "info" command and try to run
5dd29be task 3, change MAX_INSTR_TO_PRINT, finished
26a41f8 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:54:53 up  1:20,  2 users,  load average: 0.00, 0.00, 0.00 3370fcf081387d58fcf8d2256c9
0dd29c8e11d92
81f0ec8 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:54:29 up  1:19,  2 users,  load average: 0.00, 0.00, 0.00 6e4444c4efd9e948e42141aa46e
1184c53e74cb
1179817 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:54:07 up  1:19,  2 users,  load average: 0.00, 0.00, 0.00 4995656df517b1e3182cadba11b
cd26c40f147db
9f1810d >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:53:35 up  1:18,  2 users,  load average: 0.00, 0.00, 0.00 9ddbdf59a09802995a246c71f1c
5cde5c4777617
888086c >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:53:35 up  1:18,  2 users,  load average: 0.00, 0.00, 0.00 7d787645b7cc3d60136f0ee
cb5dfdf9021b57be1
05f8769 change MAX_INSTR_TO_PRINT from 10 to 1000001
979e55c >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:49:21 up  1:14,  2 users,  load average: 0.00, 0.00, 0.00 3964cca7327baa29fb82ca098af
87b655f10fa8
3e9ed0c >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:48:50 up  1:13,  2 users,  load average: 0.00, 0.00, 0.00 99afdfc05c48fa0efbc3129dd37
d487faf47695
d988df6 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:39:57 up  1:05,  2 users,  load average: 0.02, 0.01, 0.00 87a6d3854a222c06bed34dad89e
d05322c6641c
e58e31c >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:39:36 up  1:04,  2 users,  load average: 0.04, 0.01, 0.00 145c0f3a66a220b65ff61afb81c
f0f3ff36d24ba
98b36d5 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:39:36 up  1:04,  2 users,  load average: 0.04, 0.01, 0.00 302975bee83a5d791afe5ec
c1a034e46a2679d02
:
```



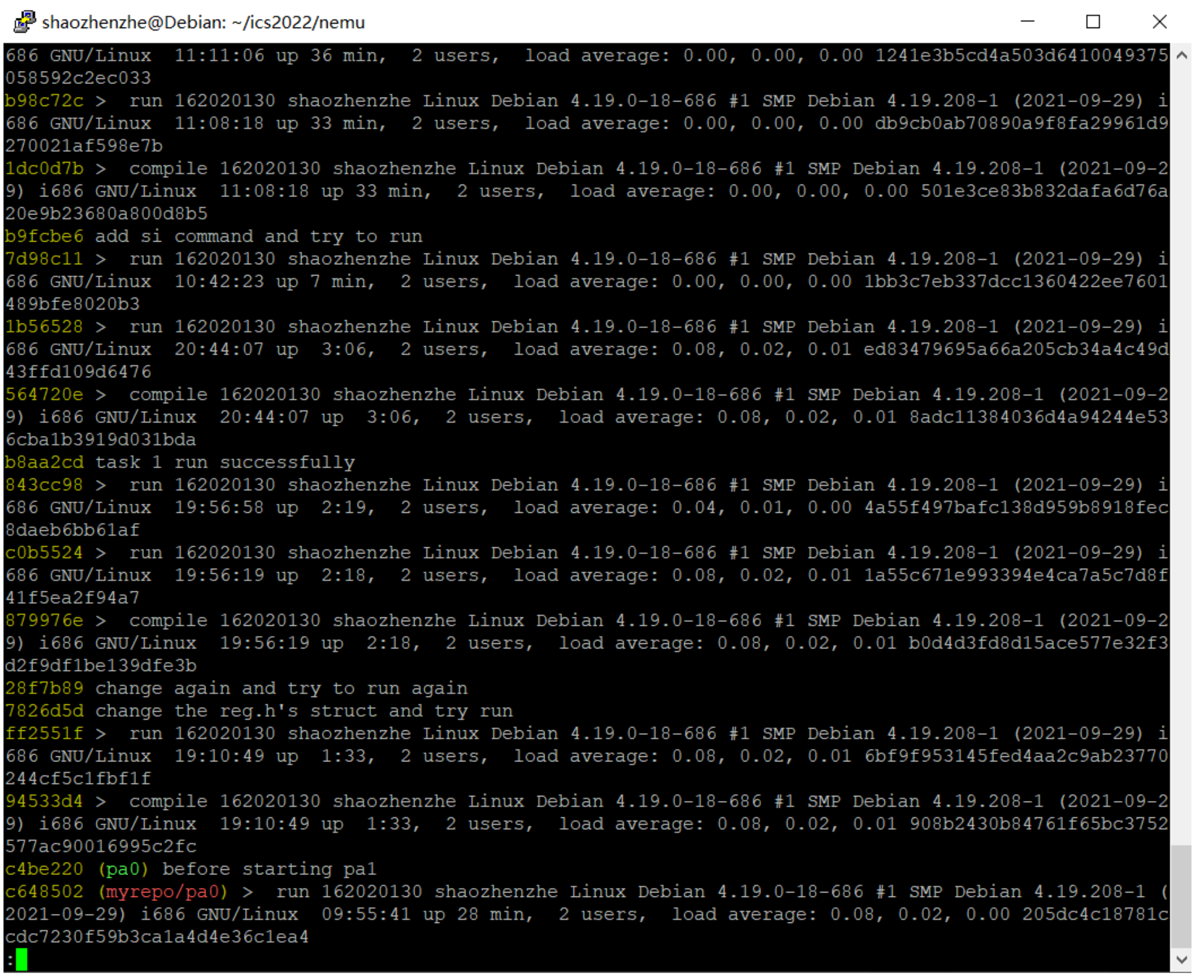
shaozhenzhe@Debian: ~/ics2022/nemu

```
98b36d5 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:39:36 up  1:04,  2 users,  load average: 0.04, 0.01, 0.00 302975bee83a5d791afe5ec
c1a034e46a2679d02
a88bcd3 find "si -1" command is wrong and solve
f89a903 si commands all run correctly
dabbffe >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:33:13 up 58 min,  2 users,  load average: 0.06, 0.02, 0.00 28ef28cc31059957c667253196
61058f4ce4964b
2eaf499 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:32:58 up 58 min,  2 users,  load average: 0.08, 0.02, 0.01 3416058642b03bb0835c485784
d0ef3850ed6754
5d9cfbe >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:32:58 up 58 min,  2 users,  load average: 0.08, 0.02, 0.01 b3d3907bdf9867e221bcc4
a57a5a00b4d58dbc
15c2d44 solve the "si" command and try to run again
36ab905 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:28:52 up 53 min,  2 users,  load average: 0.08, 0.02, 0.01 d1d3506227795bb6c706941ada
445778958950a0
c563712 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:28:52 up 53 min,  2 users,  load average: 0.08, 0.02, 0.01 81f5f34538ea44641a4170
35e35b376d156f8b3
cf5ff12 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:26:08 up 51 min,  2 users,  load average: 0.00, 0.00, 0.00 8c48cb4ba8e035d8af0eaa28ab
dc803f7c56ade
0cb7747 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:26:04 up 51 min,  2 users,  load average: 0.00, 0.00, 0.00 5d6f06a67d95992bf36322812e
2bc4096845e224
f7514b7 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:25:02 up 50 min,  2 users,  load average: 0.00, 0.00, 0.00 ff0cfe7270ab3715c9edd3cd2c
b0d953648f3f5
e70260e >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:24:04 up 49 min,  2 users,  load average: 0.00, 0.00, 0.00 5c5a1e751b177d57371e8a29b5
e03066d2351841
1e24c2e >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:23:21 up 48 min,  2 users,  load average: 0.00, 0.00, 0.00 f4b8c8d3e271faa544a1bd6509
00fac8ba381d5
f9a12f6 "si" run wrongly and other si command run correctly
ed166b6 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:12:20 up 37 min,  2 users,  load average: 0.00, 0.00, 0.00 7c131becc101789f831151f7d7
c2d9362968d4bf
:
```



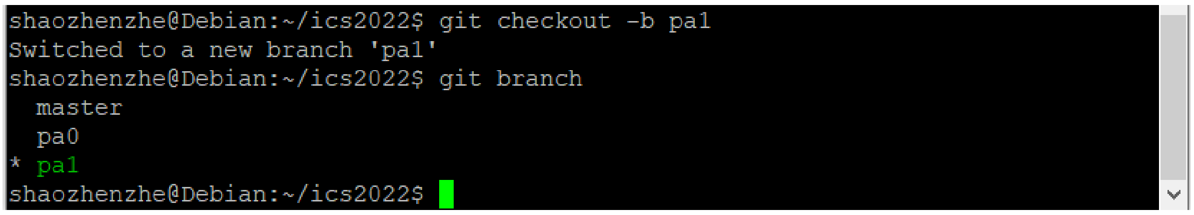
shaozhenzhe@Debian: ~/ics2022/nemu

```
ed166b6 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:12:20 up 37 min,  2 users,  load average: 0.00, 0.00, 0.00 7c131becc101789f831151f7d7
c2d9362968d4bf
a302b9a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:12:04 up 37 min,  2 users,  load average: 0.00, 0.00, 0.00 23a644beed4ebba46d90da8df9
f226b7f09c69b3
7df51bb >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:12:04 up 37 min,  2 users,  load average: 0.00, 0.00, 0.00 64f98752d19f6dc95a11bb
56f09f134cc08b8d5f
771758a >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:11:06 up 36 min,  2 users,  load average: 0.00, 0.00, 0.00 1241e3b5cd4a503d6410049375
058592c2ec033
b98c72c >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:08:18 up 33 min,  2 users,  load average: 0.00, 0.00, 0.00 db9cb0ab70890a9f8fa29961d9
270021af598e7b
1dc0d7b >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:08:18 up 33 min,  2 users,  load average: 0.00, 0.00, 0.00 501e3ce83b832dafa6d76a
20e9b23680a800d8b5
b9fcbe6 add si command and try to run
7d98c11 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  10:42:23 up 7 min,  2 users,  load average: 0.00, 0.00, 0.00 1bb3c7eb337dcc1360422ee7601
489bfe8020b3
1b56528 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  20:44:07 up  3:06,  2 users,  load average: 0.08, 0.02, 0.01 ed83479695a66a205cb34a4c49d
43ffd109d6476
564720e >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  20:44:07 up  3:06,  2 users,  load average: 0.08, 0.02, 0.01 8adc11384036d4a94244e53
6cba1b3919d031bda
b8aa2cd task 1 run successfully
843cc98 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:56:58 up  2:19,  2 users,  load average: 0.04, 0.01, 0.00 4a55f497bafc138d959b8918fec
8daeb6bb61af
c0b5524 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:56:19 up  2:18,  2 users,  load average: 0.08, 0.02, 0.01 1a55c671e993394e4ca7a5c7d8f
41f5ea2f94a7
879976e >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  19:56:19 up  2:18,  2 users,  load average: 0.08, 0.02, 0.01 b0d4d3fd8d15ace577e32f3
d2f9df1be139dfe3b
28f7b89 change again and try to run again
7826d5d change the reg.h's struct and try run
:
```



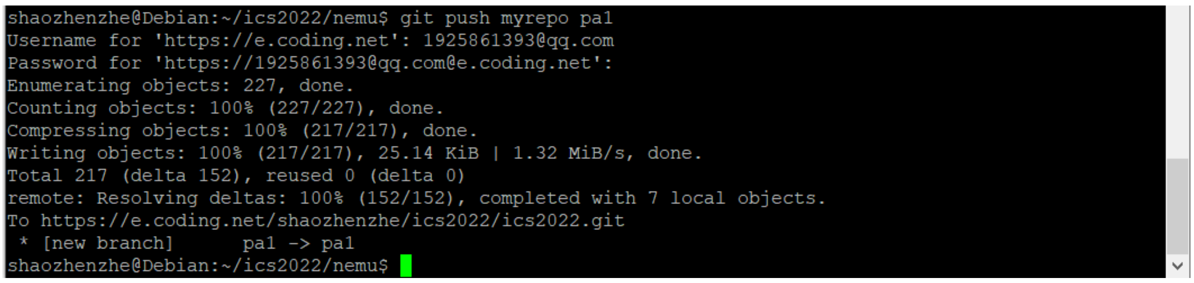
```
shaozhenzhe@Debian: ~/ics2022/nemu
686 GNU/Linux  11:11:06 up 36 min,  2 users,  load average: 0.00, 0.00, 0.00 1241e3b5cd4a503d6410049375
058592c2ec033
b98c72c >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  11:08:18 up 33 min,  2 users,  load average: 0.00, 0.00, 0.00 db9cb0ab70890a9f8fa29961d9
270021af598e7b
1dc0d7b >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  11:08:18 up 33 min,  2 users,  load average: 0.00, 0.00, 0.00 501e3ce83b832dafa6d76a
20e9b23680a800d8b5
b9fcbe6 add si command and try to run
7d98c11 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  10:42:23 up 7 min,  2 users,  load average: 0.00, 0.00, 0.00 1bb3c7eb337dcc1360422ee7601
489bfe8020b3
1b56528 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  20:44:07 up  3:06,  2 users,  load average: 0.08, 0.02, 0.01 ed83479695a66a205cb34a4c49d
43ffd109d6476
564720e >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  20:44:07 up  3:06,  2 users,  load average: 0.08, 0.02, 0.01 8adc11384036d4a94244e53
6cba1b3919d031bda
b8aa2cd task 1 run successfully
843cc98 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:56:58 up  2:19,  2 users,  load average: 0.04, 0.01, 0.00 4a55f497bafc138d959b8918fec
8daeb6bb61af
c0b5524 >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:56:19 up  2:18,  2 users,  load average: 0.08, 0.02, 0.01 1a55c671e993394e4ca7a5c7d8f
41f5ea2f94a7
879976e >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  19:56:19 up  2:18,  2 users,  load average: 0.08, 0.02, 0.01 b0d4d3fd8d15ace577e32f3
d2f9df1be139dfe3b
28f7b89 change again and try to run again
7826d5d change the reg.h's struct and try run
ff2551f >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-29) i
686 GNU/Linux  19:10:49 up  1:33,  2 users,  load average: 0.08, 0.02, 0.01 6bf9f953145fed4aa2c9ab23770
244cf5c1fbf1f
94533d4 >  compile 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (2021-09-2
9) i686 GNU/Linux  19:10:49 up  1:33,  2 users,  load average: 0.08, 0.02, 0.01 908b2430b84761f65bc3752
577ac90016995c2fc
c4be220 (pa0) before starting pa1
c648502 (myrepo/pa0) >  run 162020130 shaozhenzhe Linux Debian 4.19.0-18-686 #1 SMP Debian 4.19.208-1 (
2021-09-29) i686 GNU/Linux  09:55:41 up 28 min,  2 users,  load average: 0.08, 0.02, 0.00 205dc4c18781c
cdc7230f59b3ca1a4d4e36c1ea4
:
```

9.Git Branch截图， 使用命令 git branch 并截图，尽量完整。

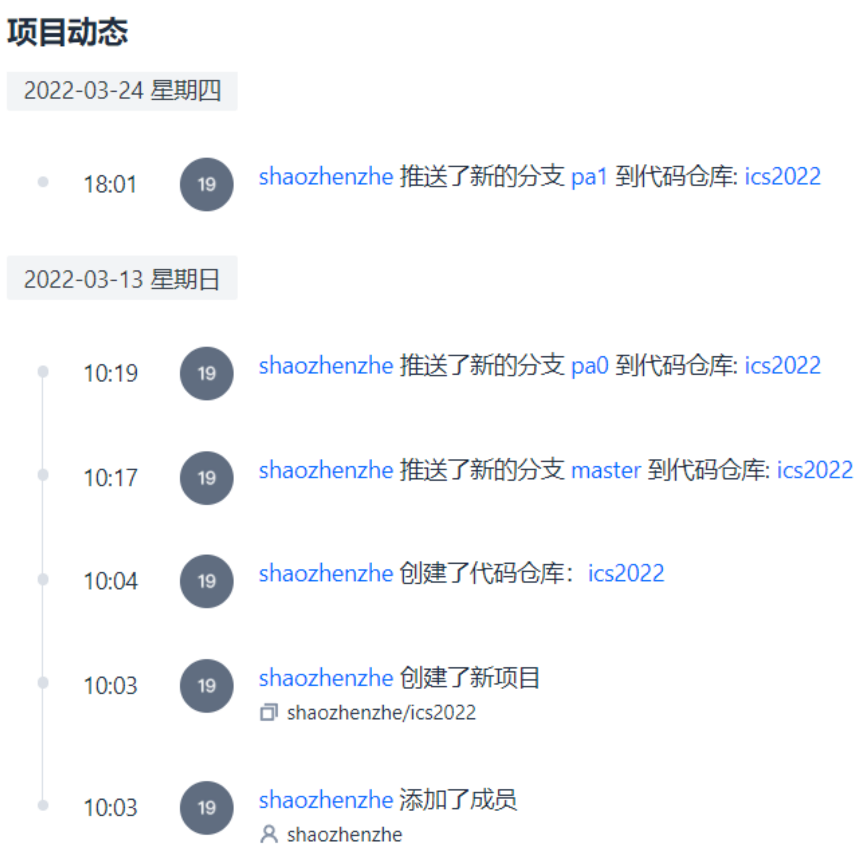


```
shaozhenzhe@Debian:~/ics2022$ git checkout -b pa1
Switched to a new branch 'pa1'
shaozhenzhe@Debian:~/ics2022$ git branch
  master
  pa0
* pa1
shaozhenzhe@Debian:~/ics2022$
```

10.远程git仓库提交截图， 做完后，把此次代码提交到 github (或国内基于 git 的仓库)上，并截图



```
shaozhenzhe@Debian:~/ics2022/nemu$ git push myrepo pa1
Username for 'https://e.coding.net': 1925861393@qq.com
Password for 'https://1925861393@qq.com@e.coding.net':
Enumerating objects: 227, done.
Counting objects: 100% (227/227), done.
Compressing objects: 100% (217/217), done.
Writing objects: 100% (217/217), 25.14 KiB | 1.32 MiB/s, done.
Total 217 (delta 152), reused 0 (delta 0)
remote: Resolving deltas: 100% (152/152), completed with 7 local objects.
To https://e.coding.net/shaozhenzhe/ics2022/ics2022.git
 * [new branch]      pa1 -> pa1
shaozhenzhe@Debian:~/ics2022/nemu$
```


项目动态

2022-03-24 星期四

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

2022-03-13 星期日

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

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

10:04 shaozhenzhe 创建了代码仓库： ics2022

10:03 shaozhenzhe 创建了新项目
shaozhenzhe/ics2022

10:03 shaozhenzhe 添加了成员
shaozhenzhe

实验内容

实现寄存器结构体

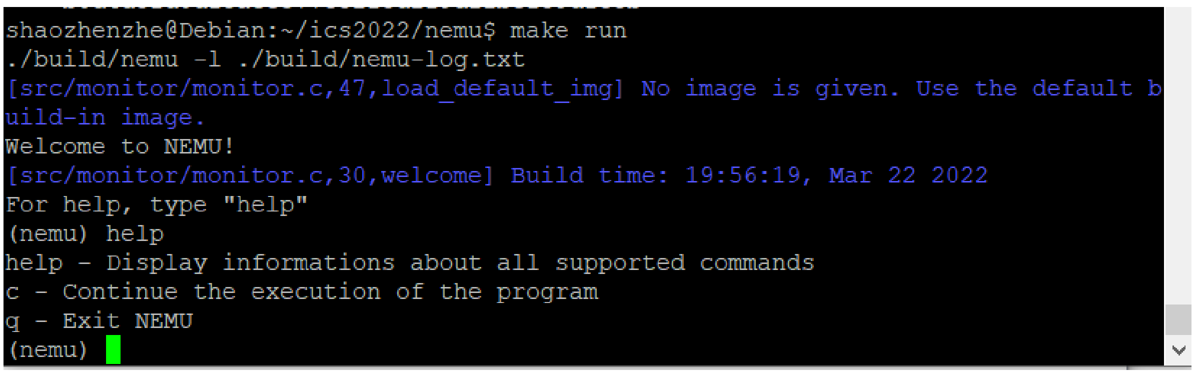
先用`union`声明8个32位寄存器，每个寄存器都共用内存，能够满足我们的需要

由于寄存器变量是单独定义的，为了和上面申请的空间一一对应，用`struct`构建变量，再把`struct`和之前的`union`再用一个`union`联合，这样`struct`的变量就和寄存器空间共用内存且一一对应，最后再用`struct`把单独的`eip`包含到一起

```c
typedef struct {
    union{
        union{
            uint32_t _32;
            uint16_t _16;
            uint8_t _8[2];
        }gpr[8];

  /* Do NOT change the order of the GPRs' definitions. */

  /* In NEMU, rtlreg_t is exactly uint32_t. This makes RTL instructions
   * in PA2 able to directly access these registers.
   */
        struct{
            rtlreg_t eax, ecx, edx, ebx, esp, ebp, esi, edi;
        };
    };
    vaddr_t eip;
} CPU_state;
```

构建完成后编译运行成功进入NEMU界面

```
make run
```



```
shaozhenzhe@Debian:~/ics2022/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 19:56:19, Mar 22 2022
For help, type "help"
(nemu) help
help - Display informations about all supported commands
c - Continue the execution of the program
q - Exit NEMU
(nemu)
```

实现单步执行

先把`si`命令加入到`cmd_table[]`

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

然后构建`cmd_si()`函数，可以模仿`cmd_help()`函数

首先用`char *arg = strtok(NULL, " ");`分离出`si`后面的步数，用`sscanf(arg, "%d", &count);`从字符串读取格式化输入数字

这里要分情况，`si`命令后缺省就默认执行一步，其他情况执行对应步数即可（-1等同于`c`，也就是最大）

根据`cmd_c()`函数，可知是调用了`cpu_exec()`函数执行步数

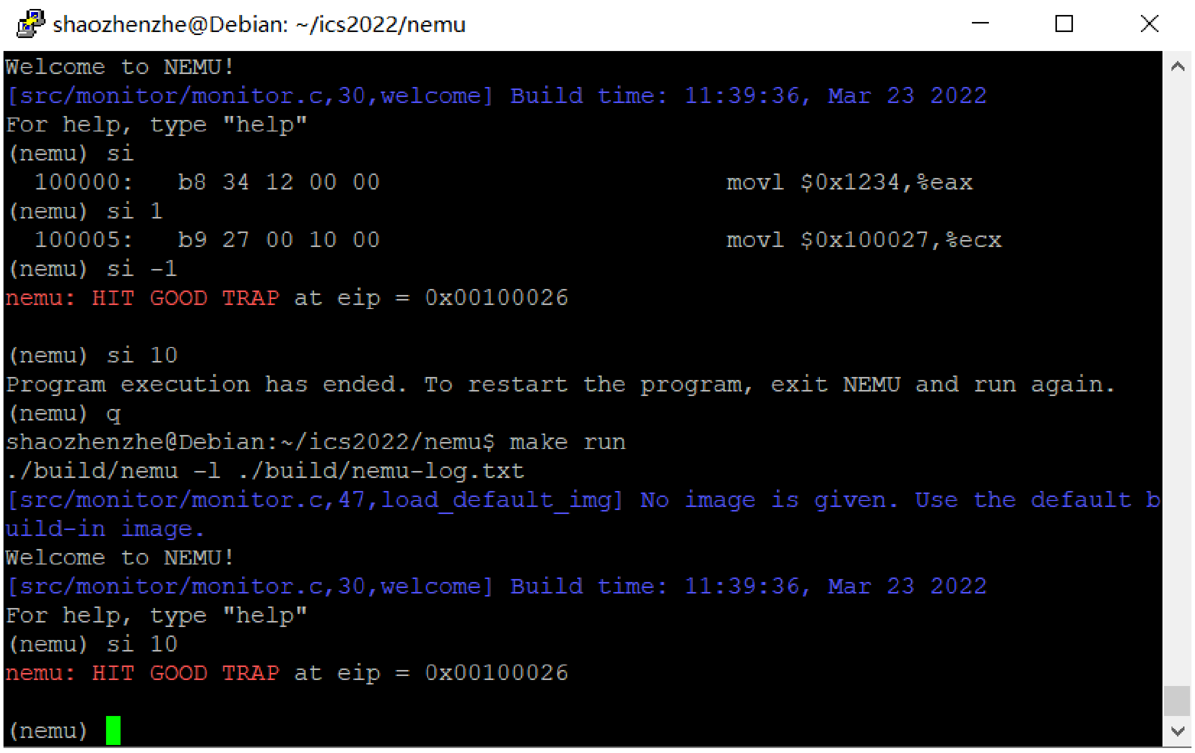
```
static int cmd_si(char *args) {
    char *arg = strtok(NULL, " ");
    int count;
    if(arg == NULL) {   /*no numbers so only run once*/
        cpu_exec(1);
        return 0;
    }
    sscanf(arg, "%d", &count);/*read how many steps*/
    cpu_exec(count);
    return 0;
}
```

构建完成后编译运行

```
make clean
make run
```

测试用例 `si 1` (运行1步)， `si` (运行1步)， `si -1` (等同于 c)， `si 10` (运行10 步)

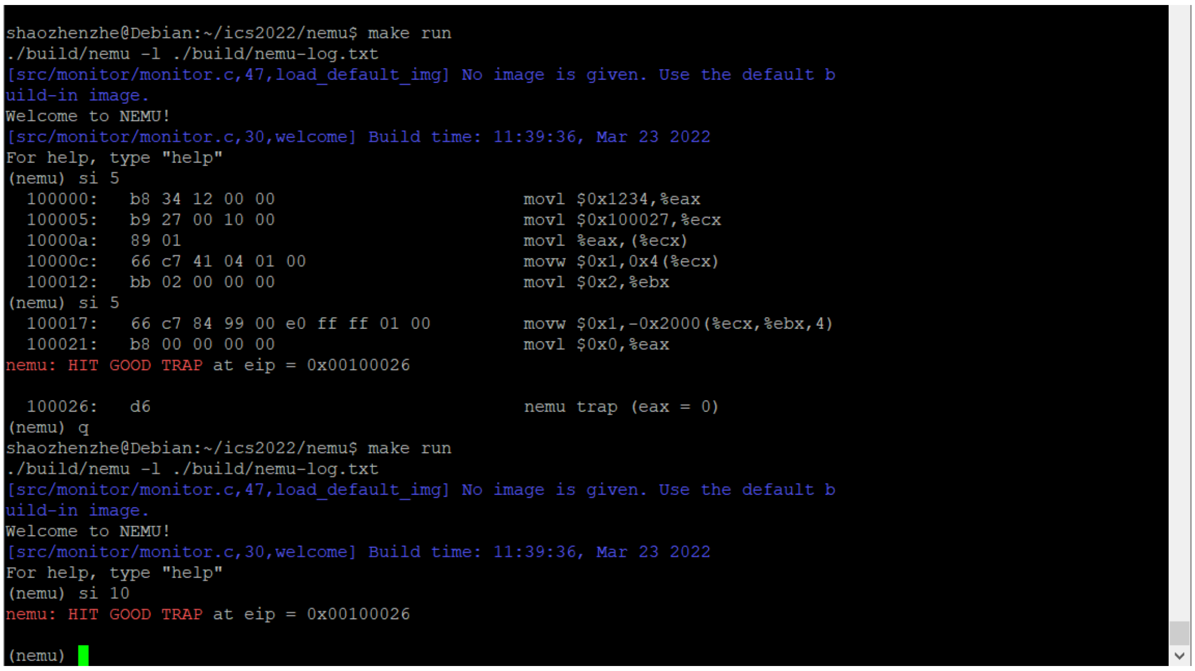
运行 `si 10` 之前程序已经结束，所以需要先退出重新进入后再执行



```
shaozhenzhe@Debian: ~/ics2022/nemu
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 11:39:36, Mar 23 2022
For help, type "help"
(nemu) si
  100000:   b8 34 12 00 00                          movl $0x1234,%eax
(nemu) si 1
  100005:   b9 27 00 10 00                          movl $0x100027,%ecx
(nemu) si -1
nemu: HIT GOOD TRAP at eip = 0x00100026

(nemu) si 10
Program execution has ended. To restart the program, exit NEMU and run again.
(nemu) q
shaozhenzhe@Debian:~/ics2022/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 11:39:36, Mar 23 2022
For help, type "help"
(nemu) si 10
nemu: HIT GOOD TRAP at eip = 0x00100026

(nemu) 
```

修改一次打印步数上限

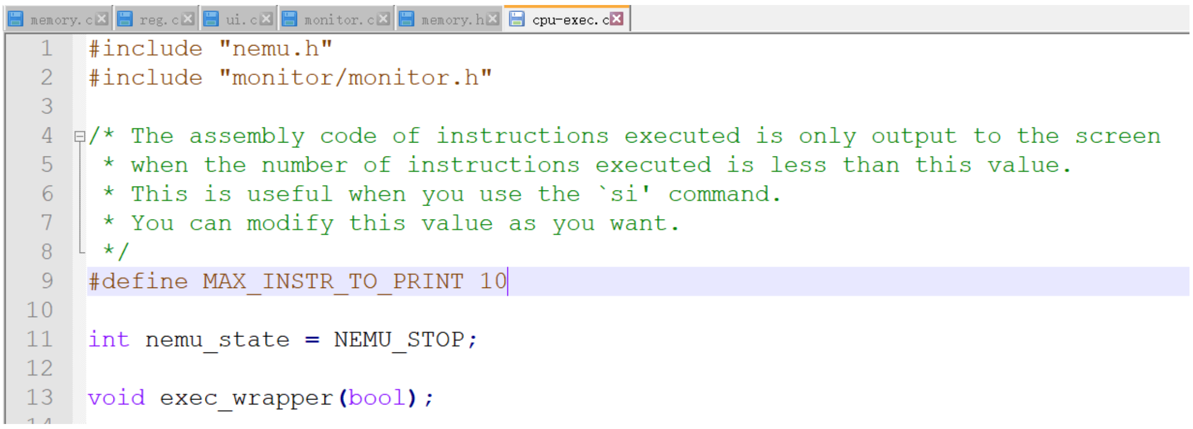
运行两次 `si 5` 命令和运行一次 `si 10` 命令



```
shaozhenzhe@Debian:~/ics2022/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 11:39:36, Mar 23 2022
For help, type "help"
(nemu) si 5
  100000:   b8 34 12 00 00                          movl $0x1234,%eax
  100005:   b9 27 00 10 00                          movl $0x100027,%ecx
  10000a:   89 01                                   movl %eax,(%ecx)
  10000c:   66 c7 41 04 01 00                       movw $0x1,0x4(%ecx)
  100012:   bb 02 00 00 00                          movl $0x2,%ebx
(nemu) si 5
  100017:   66 c7 84 99 00 e0 ff ff 01 00           movw $0x1,-0x2000(%ecx,%ebx,4)
  100021:   b8 00 00 00 00                          movl $0x0,%eax
nemu: HIT GOOD TRAP at eip = 0x00100026

  100026:   d6                                      nemu trap (eax = 0)
(nemu) q
shaozhenzhe@Debian:~/ics2022/nemu$ make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default b
uild-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 11:39:36, Mar 23 2022
For help, type "help"
(nemu) si 10
nemu: HIT GOOD TRAP at eip = 0x00100026

(nemu) 
```

可以看到控制台的输出不一样

原因是在 `/nemu/src/monitor/cpu-exec.c` 中的宏 `MAX_INSTR_TO_PRINT` 设置为10，只有步数小于10才会有输出



```c
#include "nemu.h"
#include "monitor/monitor.h"

/* The assembly code of instructions executed is only output to the screen
 * when the number of instructions executed is less than this value.
 * This is useful when you use the `si' command.
 * You can modify this value as you want.
 */
#define MAX_INSTR_TO_PRINT 10

int nemu_state = NEMU_STOP;

void exec_wrapper(bool);
```

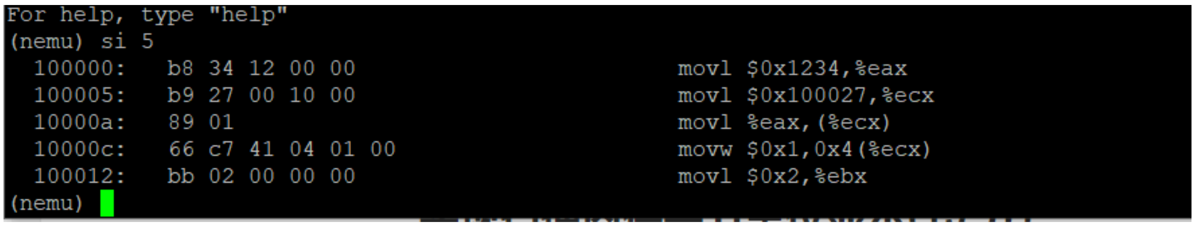
因此改变`MAX_INSTR_TO_PRINT`即可

```
#define MAX_INSTR_TO_PRINT 1000001
```

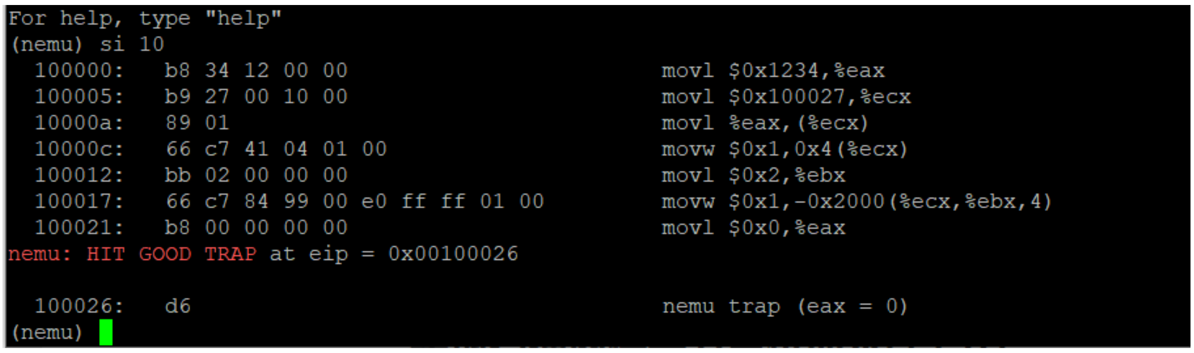
修改后编译运行

```
make clean
make run
```

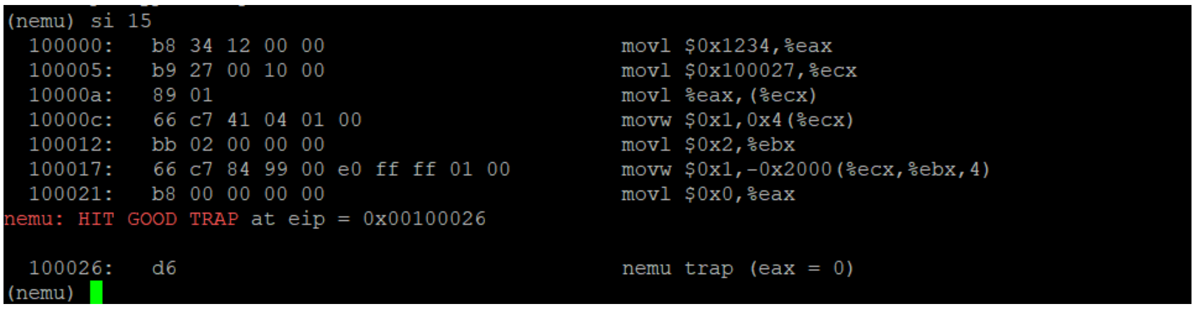
测试用例`si 5`，`si 10`，`si 15`，`si 1000000`



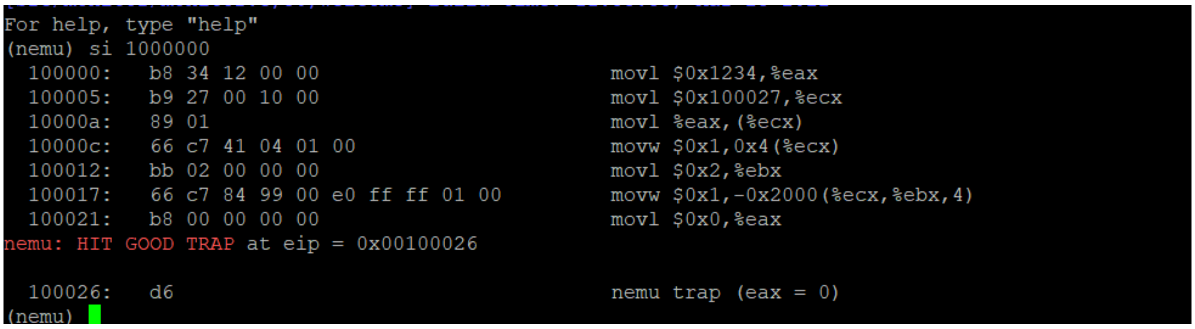
```
For help, type "help"
(nemu) si 5
  100000:    b8 34 12 00 00                              movl $0x1234,%eax
  100005:    b9 27 00 10 00                              movl $0x100027,%ecx
  10000a:    89 01                                       movl %eax,(%ecx)
  10000c:    66 c7 41 04 01 00                           movw $0x1,0x4(%ecx)
  100012:    bb 02 00 00 00                              movl $0x2,%ebx
(nemu)
```



```
For help, type "help"
(nemu) si 10
  100000:    b8 34 12 00 00                              movl $0x1234,%eax
  100005:    b9 27 00 10 00                              movl $0x100027,%ecx
  10000a:    89 01                                       movl %eax,(%ecx)
  10000c:    66 c7 41 04 01 00                           movw $0x1,0x4(%ecx)
  100012:    bb 02 00 00 00                              movl $0x2,%ebx
  100017:    66 c7 84 99 00 e0 ff ff 01 00               movw $0x1,-0x2000(%ecx,%ebx,4)
  100021:    b8 00 00 00 00                              movl $0x0,%eax
nemu: HIT GOOD TRAP at eip = 0x00100026

  100026:    d6                                          nemu trap (eax = 0)
(nemu)
```



```
(nemu) si 15
  100000:    b8 34 12 00 00                              movl $0x1234,%eax
  100005:    b9 27 00 10 00                              movl $0x100027,%ecx
  10000a:    89 01                                       movl %eax,(%ecx)
  10000c:    66 c7 41 04 01 00                           movw $0x1,0x4(%ecx)
  100012:    bb 02 00 00 00                              movl $0x2,%ebx
  100017:    66 c7 84 99 00 e0 ff ff 01 00               movw $0x1,-0x2000(%ecx,%ebx,4)
  100021:    b8 00 00 00 00                              movl $0x0,%eax
nemu: HIT GOOD TRAP at eip = 0x00100026

  100026:    d6                                          nemu trap (eax = 0)
(nemu)
```



```
For help, type "help"
(nemu) si 1000000
  100000:   b8 34 12 00 00                          movl $0x1234,%eax
  100005:   b9 27 00 10 00                          movl $0x100027,%ecx
  10000a:   89 01                                   movl %eax,(%ecx)
  10000c:   66 c7 41 04 01 00                       movw $0x1,0x4(%ecx)
  100012:   bb 02 00 00 00                          movl $0x2,%ebx
  100017:   66 c7 84 99 00 e0 ff ff 01 00           movw $0x1,-0x2000(%ecx,%ebx,4)
  100021:   b8 00 00 00 00                          movl $0x0,%eax
nemu: HIT GOOD TRAP at eip = 0x00100026

  100026:   d6                                      nemu trap (eax = 0)
(nemu)
```

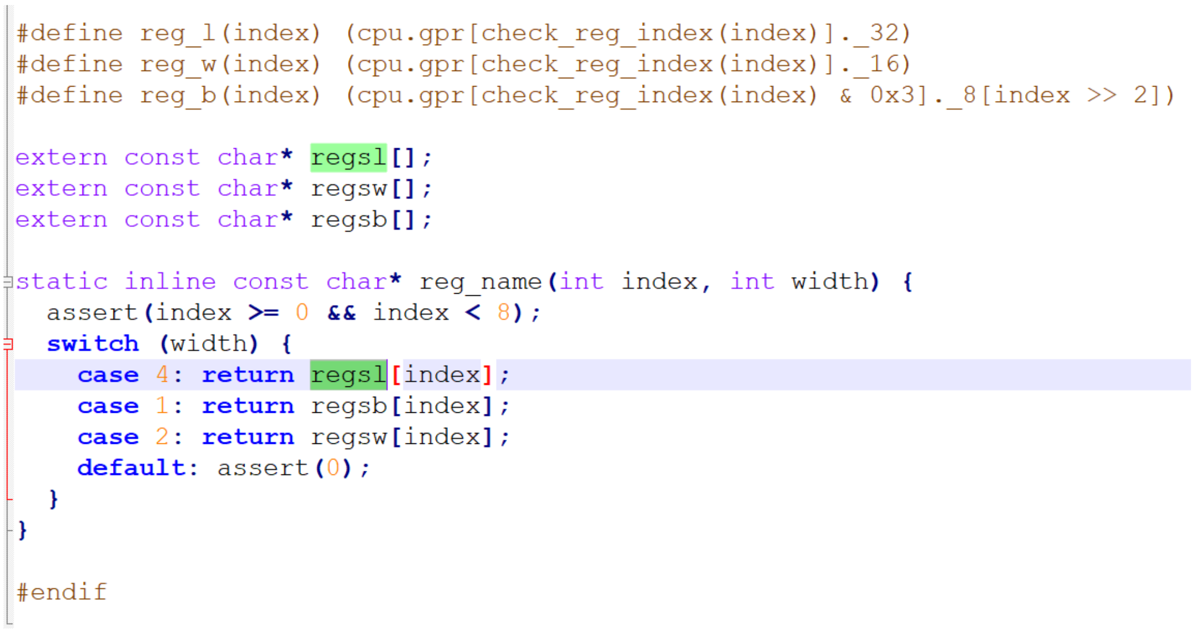
实现打印寄存器功能

先把`info`命令加入到`cmd_table[]`

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

`char *arg = strtok(NULL, " ");`分离`info`命令后的内容，如果是`r`，就打印信息，`w`待添加

根据`/nemu/include/cpu/reg.h`的结构体定义，可知`regsl[]`里存放的是32位寄存器的名字，`cpu.gpr[]._32`，也就是`reg_l()`，是寄存器的值，分别用`%08x`和`%d`的方式输出8位补零十六进制和十进制

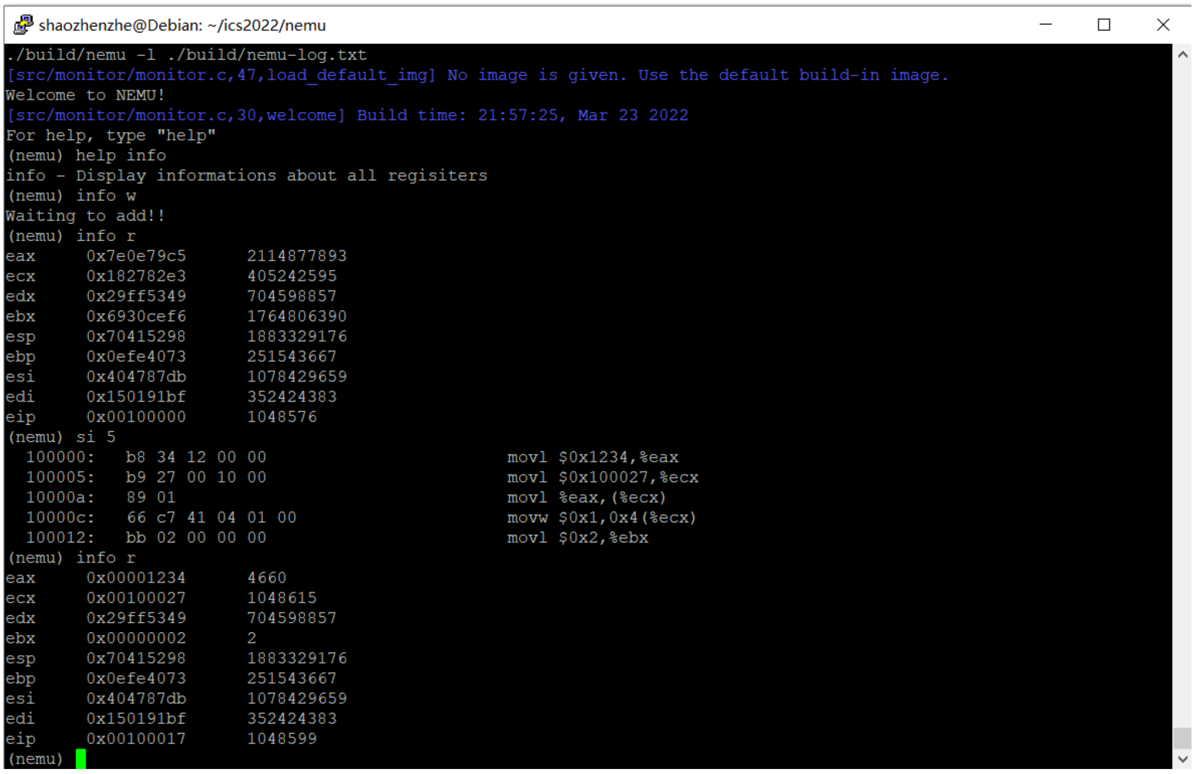


```
#define reg_l(index) (cpu.gpr[check_reg_index(index)]._32)
#define reg_w(index) (cpu.gpr[check_reg_index(index)]._16)
#define reg_b(index) (cpu.gpr[check_reg_index(index) & 0x3]._8[index >> 2])

extern const char* regsl[];
extern const char* regsw[];
extern const char* regsb[];

static inline const char* reg_name(int index, int width) {
  assert(index >= 0 && index < 8);
  switch (width) {
    case 4: return regsl[index];
    case 1: return regsb[index];
    case 2: return regsw[index];
    default: assert(0);
  }
}

#endif
```

用for循环输出8个寄存器信息，最后加上`eip`的信息即可

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

测试先执行命令`info r`，`si 5`后再次执行`info r`，均正常执行


shaozhenzhe@Debian: ~/ics2022/nemu

```
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 21:57:25, Mar 23 2022
For help, type "help"
(nemu) help info
info - Display informations about all regisiters
(nemu) info w
Waiting to add!!
(nemu) info r
eax     0x7e0e79c5      2114877893
ecx     0x182782e3      405242595
edx     0x29ff5349      704598857
ebx     0x6930cef6      1764806390
esp     0x70415298      1883329176
ebp     0x0efe4073      251543667
esi     0x404787db      1078429659
edi     0x150191bf      352424383
eip     0x00100000      1048576
(nemu) si 5
  100000:    b8 34 12 00 00                              movl $0x1234,%eax
  100005:    b9 27 00 10 00                              movl $0x100027,%ecx
  10000a:    89 01                                       movl %eax,(%ecx)
  10000c:    66 c7 41 04 01 00                           movw $0x1,0x4(%ecx)
  100012:    bb 02 00 00 00                              movl $0x2,%ebx
(nemu) info r
eax     0x00001234      4660
ecx     0x00100027      1048615
edx     0x29ff5349      704598857
ebx     0x00000002      2
esp     0x70415298      1883329176
ebp     0x0efe4073      251543667
esi     0x404787db      1078429659
edi     0x150191bf      352424383
eip     0x00100017      1048599
(nemu) 
```

实现扫描内存功能

先把`x`命令加入到`cmd_table[]`

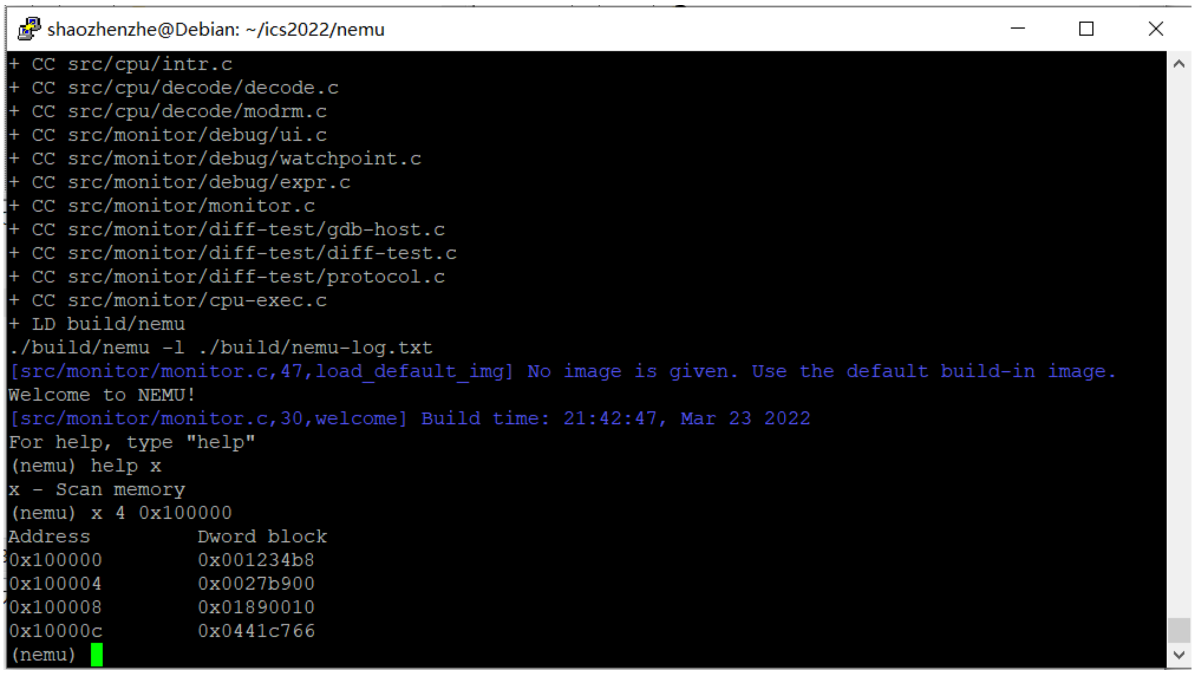
```c
static struct {
  char *name;
  char *description;
  int (*handler) (char *);
} cmd_table [] = {
  { "help", "Display informations about all supported commands", cmd_help },
  { "c", "Continue the execution of the program", cmd_c },
  { "q", "Exit NEMU", cmd_q },
  { "si", "One Step", cmd_si },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

利用两个`strtok()`和`sscanf()`函数分离出扫描内存的长度和地址

利用`vaddr_read()`函数（在`/nemu/src/memory/memory.c`）读出地址上4个字节的内容，输出也为4个字节一行

```c
static int cmd_x(char *args){
    char *arg = strtok(NULL, " ");
    char *arg_1 = strtok(NULL, " ");
    int count;
    vaddr_t address;
    sscanf(arg, "%d", &count);
    sscanf(arg_1, "%x", &address);
    printf("Address\t\tDword block\n");
    for(int i=0; i<count;i++){
        printf("0x%08x\t0x%08x\t", address, vaddr_read(address, 4));
        printf("\n");
        address += 4; /*address add 4 bytes to the next Dword block*/
    }
    return 0;
}
```

测试执行命令`x 4 0x100000`，运行成功



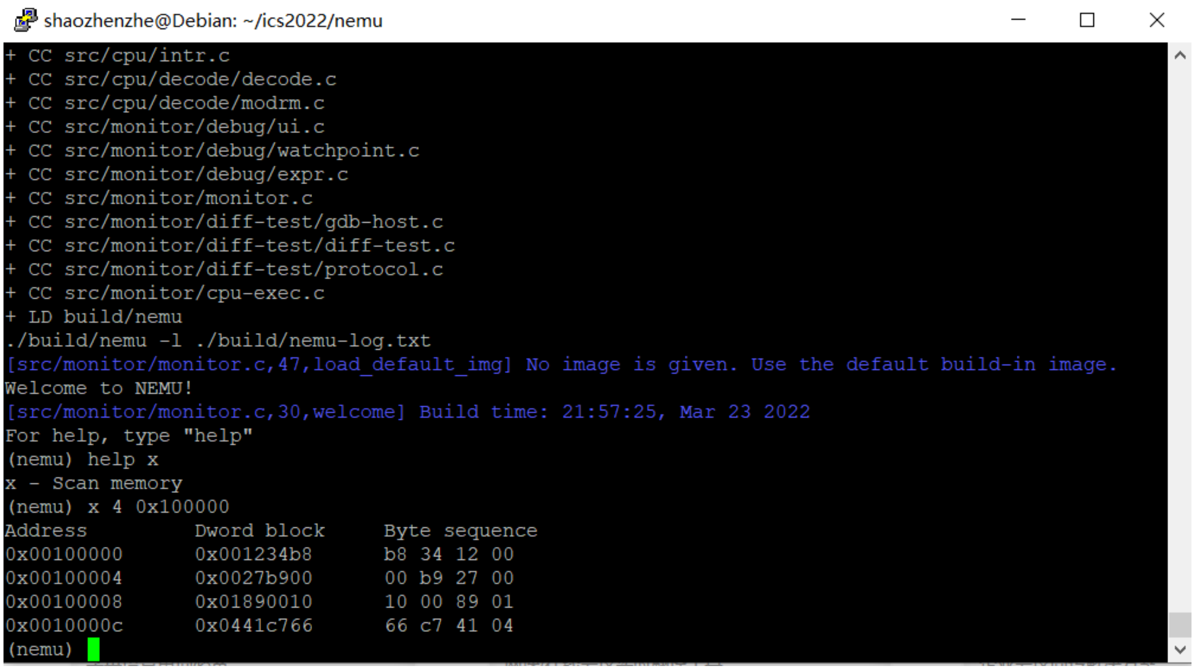
```
+ CC src/cpu/intr.c
+ CC src/cpu/decode/decode.c
+ CC src/cpu/decode/modrm.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/monitor.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/cpu-exec.c
+ LD build/nemu
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 21:42:47, Mar 23 2022
For help, type "help"
(nemu) help x
x - Scan memory
(nemu) x 4 0x100000
Address         Dword block
0x100000        0x001234b8
0x100004        0x0027b900
0x100008        0x01890010
0x10000c        0x0441c766
(nemu)
```

实现扫描内存字节单位显示

和上面的操作类似，只需要在`vaddr_read()`函数中传入参数1，即读取一个字节

```c
static int cmd_x(char *args){
    char *arg = strtok(NULL, " ");
    char *arg_1 = strtok(NULL, " ");
    int count;
    vaddr_t address;
    sscanf(arg, "%d", &count);
    sscanf(arg_1, "%x", &address);
    printf("Address\t\tDword block\tByte sequence\n");
    for(int i=0; i<count;i++){
        printf("0x%08x\t0x%08x\t", address, vaddr_read(address, 4));
        for(int j=0;j<4;j++) {
            printf("%02x ", vaddr_read(address+j, 1));/*read 1 byte once*/
        }
        printf("\n");
        address += 4; /*address add 4 bytes to the next Dword block*/
    }
    return 0;
}
```

测试执行命令`x 4 0x100000`，运行成功



shaozhenzhe@Debian: ~/ics2022/nemu

```
+ CC src/cpu/intr.c
+ CC src/cpu/decode/decode.c
+ CC src/cpu/decode/modrm.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/monitor.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/cpu-exec.c
+ LD build/nemu
./build/nemu -l ./build/nemu-log.txt
[src/monitor/monitor.c,47,load_default_img] No image is given. Use the default build-in image.
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 21:57:25, Mar 23 2022
For help, type "help"
(nemu) help x
x - Scan memory
(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
(nemu)
```

遇到的问题及解决办法

1.遇到问题：在添加`si`命令时，编译无问题，并且运行`si 1`也没有问题，但是运行`si`命令时报错`Segmentation fault`

  解决方案：上网搜索后发现可能是访问了不存在的内存，于是仔细查看我写的代码，我在进行`if(arg == NULL)`判断之前，执行了`sscanf(arg, "%d", &count);`，如果`arg`确实为`NULL`，那么`sscanf`对空指针读取就会报错，因此我把`sscanf`写在`if`判断后，问题就解决了，可以正常执行`si`命令

2.遇到问题：添加`info`命令时，`cmd_info()`函数编译报错`no return statement in function returning non-void`

  解决方案：在函数最后加上`return 0;`即可。这里也是一个需要注意的点，以前在`windows`的`devcpp`软件里编译经常不写`return 0;`但是在`linux`里编译比较严格

3.遇到问题：添加`si`命令时，`cmd_si()`函数编译报错`passing argument 1 of 'sscanf' make pointer from integer without a cast`

  解决方案：仔细查看了`sscanf`的用法，发现第一个参数传入的应该是字符串指针，而我定义了`char *arg`，但是使用时却`sscanf(*arg, "%d", &count);`，改为`sscanf(arg, "%d", &count);`成功编译

实验心得

了解了寄存器的结构和实现方法，`union`和`struct`用得好可以构造出神奇的结构。尝试构建了几个命令，在实现的同时了解了这些命令是如何被读取并执行的，这也让我对平时`windows cmd`命令和`linux bash`命令的实现有了一点了解。锻炼了阅读大规模代码的能力，可以做到在数量众多的文件和函数中找到目标函数，并且能够做到正确调用。这次PA1.1收获很大。

其他备注

无