

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：PA1.1
- 完成日期：2021.3.22
- 本次实验，我完成了所有内容。

目录

南京航空航天大学《计算机组成原理II课程设计》报告 目录

思考题

- 一、存放的是什么？
- 二、贵圈真乱
- 三、虚拟机和模拟器的区别
- 四、从哪开始阅读代码呢
- 五、究竟要执行多久
- 六、谁来指示程序的结束？
- 七、为什么会这样？
- 八、Git Log截图
- 九、Git Branch截图
- 十、远程git仓库提交截图

实验内容

- PA1.1.1 实现寄存器结构体
- PA1.1.2 实现单步执行
- PA1.1.3 修改一次打印步数上限
- PA1.1.4 实现打印寄存器功能
- PA1.1.5 实现扫描内存功能
- PA1.1.6 实现扫描内存字节单位显示

遇到的问题及解决办法

实验心得

其他备注

思考题

一、存放的是什么？

为什么是存放指令的存放地址而不是指令本身呢？

存放指令的地址可以通过当前指令地址和当前指令长度得到下一条指令地址，从而一直执行下去。否则不知道下一条指令是哪一条。

二、贵圈真乱

```
+-----+
| "Hello world" program |
+-----+
| Abstract Machine      |
+-----+
| simulated x86 hardware |
+-----+
| NEMU                  |
+-----+
| GNU/Linux            |
+-----+
| VirtualBox (Simulated Hardware) |
+-----+
| Host Operating System |
+-----+
| Computer hardware     |
+-----+
```

三、虚拟机和模拟器的区别

模拟器其实也是虚拟机的一种，和虚拟机的差别在于，模拟器使用固化的操作系统，不能随意更改操作系统，且结构简单，不能随意添加虚拟硬件设备。

四、从哪开始阅读代码呢

从 `main` 函数开始执行。

应该从 `nemu/src/main.c` 开始阅读。

五、究竟要执行多久

`void cpu_exec(uint64_t n)` 的参数是 64 位无符号数。当传入的参数为 -1 时，它将被解释为 64 位的无符号整数，64 位无符号整数的最大值。

同时，该函数中让 `cpu` 不断执行指令的是一个 `for` 循环，`for (; n>0 ;n--)`，`n` 非常大时可以保证 `cpu` 持续不断地执行指令，直到执行完所有的指令，而避免指令还没执行完就提前停止执行。

六、谁来指示程序的结束？

`CRT`（显示器）在收到 `main` 函数返回值后，将这个值传递了 `win32API ExitProcess`，当程序执行这个 `API` 时，操作系统将终止应用程序并释放整个进程资源，然后进程死亡。

七、为什么会这样？

数据的排列顺序分为大端法和小端法。大端法将最高有效字节数据存储在低地址单元中，最低有效字节存放在高地址单元中；小端法相反。

Linux 采用的是小端法，而将字节进行拆分操作时，是先从低地址位进行获取，所以 4 字节为单位打印和以 1 字节为单位打印时相比，顺序会不一样。

八、Git Log截图

在 pa1 分支下使用命令 `git log --oneline` 并截图

```
ipa > git log --oneline
60efecb > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:48:26 up 1 day, 7:22, 1 user, load average: 0
.08, 0.03, 0.01 80bb655a3f5eba22738e7b19bf48e7297d7469d1
ef39b5d > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:43:32 up 1 day, 7:17, 1 user, load average: 0.03,
0.03, 0.00 8b21315f9ec8937879557f4038aaf80a920dec1c
e922082 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:43:32 up 1 day, 7:17, 1 user, load average: 0
.03, 0.03, 0.00 37b0d57a39e9fea8b789d5e3cc5b7f3e54739fe8
6b28e54 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:41:14 up 1 day, 7:15, 1 user, load average: 0.06,
0.03, 0.01 f82969e06f84b5d1cf47f38fa9167eaf7040b6f
ad41134 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:40:47 up 1 day, 7:14, 1 user, load average: 0.09,
0.04, 0.01 498e2099d858878dfe8edf5cd320405ec4830f69
0b2f08d > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:40:46 up 1 day, 7:14, 1 user, load average: 0
.09, 0.04, 0.01 5a7f0ee96d214701ce8a6eb843cfb7a08c6c1caf
b2c4d50 Finish Single step execution
ad05f7c > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:10:24 up 1 day, 6:44, 1 user, load average: 0.00,
0.00, 0.00 6ddc1f05a3356d5d8f0d4a9075e88e5f2e93
59d5939 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:10:24 up 1 day, 6:44, 1 user, load average: 0
.00, 0.00, 0.00 322843837c28bba9ffff8b8d6ac8872eb062fc
f5f444f > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:09:07 up 1 day, 6:42, 1 user, load average: 0.00,
0.00, 0.00 4bbaa158c583eb77eb1cd02b1220c6f1be7f424
c6f1be4 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:09:07 up 1 day, 6:42, 1 user, load average: 0
.00, 0.00, 0.00 b90fb0c112e39f0b169345e6675d6a9f20c8248
54bce67 添加注释
aa3e00e > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:52:23 up 3:09, 1 user, load average: 0.08, 0.02,
0.01 908975e3c2abce8932d2d4380806b4f4949de64
79343a5 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:52:23 up 3:09, 1 user, load average: 0.08, 0.0,
0.02, 0.01 beee8d01e443cbb552cdd4d3d598ce0e53d212f
9f874d0 change reg.h
06aae24 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:42:31 up 2:59, 1 user, load average: 0.08, 0.02,
0.01 64b7b3813d050e06902d4765b421e1dd95abf731
6879d7 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:42:31 up 2:59, 1 user, load average: 0.08, 0.
02, 0.01 3db3a6d4710e8d673e889d1a53471941a254e42
6ce6800 before starting pa1
```

```
ipa > git log --oneline
.03, 0.02, 0.00 e84b1054467780938dbab2d786ed040b78163104
930ff8b > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:09:23 up 1 day, 10:43, 1 user, load average: 0.00,
0.00, 0.00 71bc8017ec56f0077ba0ecf1b6addf3a8d89410
c0a6479 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:09:23 up 1 day, 10:43, 1 user, load average: 0
.00, 0.00, 0.00 c6a422c45b86565b11f62fc58b98eacdb0228ba
42cf1da > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 23:30:23 up 1 day, 10:04, 1 user, load average: 0.00,
0.00, 0.00 fe95a16f26291e1ea0b06a7e76f7ff76a0b
eae9710 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 23:30:23 up 1 day, 10:04, 1 user, load average: 0
.00, 0.00, 0.00 7fa09abb19612e3a22cd7fd2efdc5ca9f08450
1c4e938 Add cmd si Limit Step
0bf3e24 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 23:13:10 up 1 day, 9:46, 1 user, load average: 0.00,
0.00, 0.00 618d4a713a6cfb799c17934bf1320db3925b16
08f4b81 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 23:13:10 up 1 day, 9:46, 1 user, load average: 0
.00, 0.00, 0.00 18654b02be2865069b398b9fbdedef28edddaf
6ca3ac4 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:34:53 up 1 day, 9:08, 1 user, load average: 0.00,
0.00, 0.00 5ffa30aab9ae50d740f2d8654354626d2d43ca39
eb98f4c > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:34:53 up 1 day, 9:08, 1 user, load average: 0
.00, 0.00, 0.00 f0ac848b2a87f2d9ae73bdebfde055d2e93e7f
10daa8d > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:24:12 up 1 day, 8:58, 1 user, load average: 0.00,
0.00, 0.00 e8c1afdbcb131dbf539ee2edd23c4dae58e5f407
62ef22a > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 22:21:48 up 1 day, 8:55, 1 user, load average: 0.00,
0.00, 0.00 127958bad19d5dc8410b00b6aab1496e51e8333
0ff0a16 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 21:04:57 up 1 day, 7:38, 1 user, load average: 0.00,
0.00, 0.00 c611ea358a2dc9fcdad2eab0b72e55032e6f5444
1ea0194 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 21:02:24 up 1 day, 7:36, 1 user, load average: 0.00,
0.00, 0.00 4a7b278da595ee93cfffad4c7056d597371009
4475d2f > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 21:02:24 up 1 day, 7:36, 1 user, load average: 0
.00, 0.00, 0.00 ee82d31e6d0506048a7672967ac393f09798430d9
e2279a2 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:52:11 up 1 day, 7:26, 1 user, load average: 0.00,
0.01, 0.00 4a20dbb35c450b0d3c0e5e9ef7b1d5ec770
d815a2d > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:48:27 up 1 day, 7:22, 1 user, load average: 0.08,
0.03, 0.01 8f59a53c9cc4e8c4c6c788d1f6ffda1e459ba193
```

```
ipa > git log --oneline
.02, 0.01 f8226069a6a42d5f2bcb46b3cd4f2fd7879dbdc
c8e9074 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:37:56 up 39 min, 1 user, load average: 0.05, 0.01,
0.00 ffbcc57a3adb7466f211231b799dfa71435b5409c
6306318 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:37:56 up 39 min, 1 user, load average: 0.05, 0
.01, 0.00 cf936fd225ce1154af88052f6ce935e248d7alab
41718cf > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:34:21 up 35 min, 1 user, load average: 0.00, 0.00,
0.00 67ed130dbb6e39e04ae3112e525ae56f0bd0e41
1045fe5 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:34:20 up 35 min, 1 user, load average: 0.00, 0
.00, 0.00 2ee56f5dd717ecd56ee4dc41c8d5bc5173cd777
9e56eab > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:32:27 up 33 min, 1 user, load average: 0.00, 0.00,
0.00 58fd2ed64ac2fad022d38f54122b944d5c5f4a2
5d33935 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:32:27 up 33 min, 1 user, load average: 0.00, 0
.00, 0.00 1f696c9bb6df7d67ecb700cee0493313c65055
ceef722 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:29:51 up 31 min, 1 user, load average: 0.00, 0.00,
0.00 20cdaaf7b06900e9d8ac6d7c00458f0b0
3346c51 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:29:50 up 31 min, 1 user, load average: 0.00, 0
.00, 0.00 c287199e1d0fc01e7081584e123a2ab28c8f68f1
ec7af76 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:33:31 up 1 day, 11:07, 1 user, load average: 0.02,
0.01, 0.00 c26da3d4153c9f18f9db6fcd641681bc7242ac7
9a0adeb > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:33:31 up 1 day, 11:07, 1 user, load average: 0
.02, 0.01, 0.00 ddac386e2ee5985796c91293794eac0b15da2993
6609c23 Renew cmd _function
c5e0f1e > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:27:50 up 1 day, 11:01, 1 user, load average: 0.00,
0.00, 0.00 b96a261b81892e38b687b34fa6139b27fab0d14
0ca06ad > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:27:50 up 1 day, 11:01, 1 user, load average: 0
.00, 0.00, 0.00 7693f2cc8ec04f49c76d430d965c4c6d66fcf31
7dc05de > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:23:12 up 1 day, 10:57, 1 user, load average: 0.01,
0.02, 0.00 dad624415643c8a9d244b5f4c065f30f9a80ca9
2e89237 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:22:34 up 1 day, 10:56, 1 user, load average: 0.03,
0.02, 0.00 f8a2dbb35c450b0d3c0e5e9ef7b1d5ec770
d8010e9 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 00:22:34 up 1 day, 10:56, 1 user, load average: 0
.03, 0.02, 0.00 e84b1054467780938dbab2d786ed040b78163104
```

```
1 ps +
27c11f8 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:54:31 up 56 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 4a83c0f2c23586d5d878bbf33f1fcdca3674669b
845abca > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:54:31 up 56 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 9689cd79f624a8b4d2eb0c3a8be9956c6bec08d
70dc556 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:50:53 up 52 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 1e5c5c28444d44de58b68de8990cd7e3ff6f9ac
c7179ec > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:50:53 up 52 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 5b9eab357e4dc6124a84457c50fe8d74ee16a791
ef26132 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:49:58 up 51 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 e2c855501878d183f9310e8ee040bb6ae05c9e1
c2a14b2 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:49:58 up 51 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 7115c0ec15e7b7d34270d8203da47e4b4ddcc
10dafbd > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:49:11 up 50 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 cb6e8f4912f221ced931ef1fab07f272a3b7159c2
4b5270 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:49:10 up 50 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 4bd4528baff7859027efc540cc4f39a8717902f
bc59e7 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:47:20 up 48 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 fa5b83184ccc8066aaecf7a1aa2da1319a84d39
1d714b6 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:47:20 up 48 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 f974e6bad98b377424aca798e61bd0d2df2ac4a88
3bb7296 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:44:15 up 45 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 b24f2e1d7a621ced931ef1fab07f272a3b7159c2
b3deaf5 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:44:15 up 45 min, 1 user, load average: 0.00, 0.00, 0.00
0.00 4a61c62190a9db0ddeb44e4a55e12e2d472a7ed
be0a43b > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:40:37 up 42 min, 1 user, load average: 0.03, 0.02, 0.00
0.00 946bd42310844ebbb6d5e5fa4ee82f7bd1f6945d
3c585c3 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:40:37 up 42 min, 1 user, load average: 0.03, 0.02, 0.00
0.00 7fae7c7dd8dd7daa81e2cbfe913b14dcb8a472f
1e22abc > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:39:23 up 40 min, 1 user, load average: 0.09, 0.02, 0.01
0.01 55abc6b11cfab06d5e5fa4ee82f7bd1f6945d
ae010e9 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 12:39:23 up 40 min, 1 user, load average: 0.09, 0.02, 0.01
0.01 f0226069a6a42d5f2bcb46b3cd4f2fd7879dbdc
```

```
1 ps +
0.00 7022b3868f1165eb7a4e296e67edb6a1affe19b7
598a2c8 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:12:23 up 8:13, 1 user, load average: 0.01, 0.00, 0.00
0.00 e6d83340f0b844fcb347e2afdd349ba11dd87abf
b7424ed > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:11:34 up 8:13, 1 user, load average: 0.02, 0.01, 0.00
0.00 c3edd9d3a61eb0cd381ab58e40fd2a828393d2a5c
8433027 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:11:33 up 8:13, 1 user, load average: 0.02, 0.01, 0.00
0.01 a1d194612fb9504c93157785addc65df5be6805
d39c533 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 18:55:27 up 6:56, 1 user, load average: 0.00, 0.00, 0.00
0.00 c99c260b3f6076b0c079987791a68fec8e311f8
03208cc > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 18:55:27 up 6:56, 1 user, load average: 0.00, 0.00, 0.00
0.00 5db4c4f0c9c966a2c6a266eff1573ae392a319
01572d3 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 18:46:36 up 6:48, 1 user, load average: 0.00, 0.00, 0.00
0.00 dac8e59d9f8d0dd954d7e931cd0b10d983025c
c636035 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 18:46:36 up 6:48, 1 user, load average: 0.00, 0.00, 0.00
0.00 ee4496282498a5ec9531cd085e50a23b9c0a477
6e032af > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 16:35:35 up 4:37, 1 user, load average: 0.00, 0.00, 0.00
0.00 eff5e954edc0e7359af29253f30fead434a0330
332ee95 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 16:35:35 up 4:37, 1 user, load average: 0.00, 0.00, 0.00
0.00 3d8360ff04c630270720c49fe23d3a8f3cca6e7
f94e69a Finish Print Register
a184530 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 15:54:19 up 3:55, 1 user, load average: 0.08, 0.02, 0.01
0.01 alf5c758deb9b7cb2ddb3bcd6a5dece81ce5c4
1484e29 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 15:54:19 up 3:55, 1 user, load average: 0.08, 0.02, 0.01
0.01 dcb6678fd9836cae369dd394b2dc6a4b144c0e37
38eaf57 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 14:37:41 up 2:39, 1 user, load average: 0.04, 0.02, 0.00
0.00 e6bbbe512d1fc3bf37aad844f10fd8aa6c1a3852
1fe9814 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 14:37:41 up 2:39, 1 user, load average: 0.04, 0.02, 0.00
0.00 207709b0f0f411ce4c592b735589f903ceff0156
4af677d > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 13:07:07 up 1:08, 1 user, load average: 0.10, 0.04, 0.01
0.01 1330b3f0dcba424455510b06e45e52fb18eff9b99
61fcaf7 > compile 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 13:07:06 up 1:08, 1 user, load average: 0.10, 0.04, 0.01
0.01 dd59799ec4bef520cd63f915ff734bd1b82fee3
```

```
06153c7 (HEAD -> pal) Finish Scanning memory and display with Byte Sequence
963fce2 > run 161930131 marui Linux debian 4.19.0-14-686 #1 SMP Debian 4.19.171-2 (2021-01-30) i686 GNU/Linux 20:12:23 up 8:13, 1 user, load average: 0.01, 0.00, 0.00
0.00 7022b3868f1165eb7a4e296e67edb6a1affe19b7
```

九、Git Branch截图

```
marui@debian:~/ics2021/nemu/src/monitor/debug$ git branch
* master
  pa0
  pal
```

十、远程git仓库提交截图

```
Enumerating objects: 319, done.
Counting objects: 100% (319/319), done.
Delta compression using up to 2 threads
Compressing objects: 100% (308/308), done.
Writing objects: 100% (308/308), 34.40 KiB | 366.00 KiB/s, done.
Total 308 (delta 209), reused 0 (delta 0)
remote: Resolving deltas: 100% (209/209), completed with 7 local objects.
remote: Powered by GITEE.COM [GNK-5.0]
remote: Create a pull request for 'pal' on Gitee by visiting:
remote: https://gitee.com/Leslie-Chung/ics2021/pull/new/Leslie-Chung:pal...Leslie-Chung:master
To https://gitee.com/Leslie-Chung/ics2021.git
* [new branch] pal -> pal
```

实验内容

PA1.1.1 实现寄存器结构体

实现在 `nemu/include/cpu/reg.h` 中的结构体 `CPU_state`：

```
typedef struct {
    union {
        union {
            unsigned int _32;
            unsigned short _16;
            unsigned char _8[2];
        } gpr[8];
        struct {
            rtlreg_t eax, ecx, edx, ebx, esp, ebp, esi, edi;
        }; // 这些元素各自使用各自的空间, 且每个元素与相应的 gpr[i] 共用一个地址
    };

    /* 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.
     */
    union {
        vaddr_t eip;
        unsigned short ip;
    };
    union {
        unsigned int eflags;
        unsigned short flags;
    };
} CPU_state;
```

对于第一个 union 来说, `gpr[i]` 中的元素共用一个空间, `gpr[i]` 与 `gpr[j]` 之间使用不同空间 ($0 \leq i, j < 8$) ; 其次, `struct` 中的元素占用不同的空间, 而且从左往右来看, 每个元素又与其对应的 `gpr[i]` 共用同一个空间。

剩余的两个 union 同理。

测试样例：

执行 `make run`

```
marui@debian:~/ics2021/nemu$ make run
+ CC src/monitor/debug/ui.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: 20:39:54, Mar 21 2021
For help, type "help"
(nemu) █
```

PA1.1.2 实现单步执行

声明并定义函数

```
static int cmd_si(char *args){
    char *arg = strtok(NULL, " "); //因为si指令已经被捕获，所以只需读出数字即可
    int n;
    if (arg){
        n = atoi(arg);
        if (n < -1) {
            printf("Input Digit Error!\n");
            return 0;
        }
    }
    else n = 1; //缺省
    cpu_exec(n);
    return 0;
}
```

将 `si` 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", "Single-step execution", cmd_si},
    /* TODO: Add more commands */
};
```

测试样例：

`si 1`, `si`, `si -1`, `si 10`

```
(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 10
nemu: HIT GOOD TRAP at eip = 0x00100026
```

```
(nemu) si -1
nemu: HIT GOOD TRAP at eip = 0x00100026
```

PA1.1.3 修改一次打印步数上限

观察 `cpu_exec()` 函数（`cpu_exec()` 模拟了 CPU 的工作方式：不断执行指令。）

```
for (; n > 0; n --) {
    exec_wrapper(print_flag);
}
```


`exec_wrapper()` 函数让 CPU 执行当前 `%eip` 指向的一条指令，然后更新 `%eip`。参数 `print_flag` 的初始化为 `bool print_flag = n < MAX_INSTR_TO_PRINT;`。

```
if (print_flag) {
    puts(decoding.asm_buf);
}
```

由此可见，如果该值为真，则会执行指令。

所以修改 `MAX_INSTR_TO_PRINT` 为无限大，则输入所有合法值 (`n >= -1`) 后，就会根据 `n` 一直执行程序，直到 `n == 0`。

而 `n` 的类型为 `unsigned int`，对应的无限大也就是 `-1`。

在 `nemu\src\monitor\cpu-exec.c` 中修改：

```
#define MAX_INSTR_TO_PRINT -1
```

然后执行命令，`si 5`，`si 10`，`si 15`，`si 1000000`

测试样例：

```
(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 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) 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) 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)

```

PA1.1.4 实现打印寄存器功能

代码实现:

```

void reg_display(){
    int i;
    for (i = R_EAX; i <= R_EBX; i++){
        printf("%s:\t0x%-8x\t%u\n", regsl[i], reg_l(i), reg_l(i));
        printf("%s:\t0x%-8x\t%u\n", regsw[i], reg_w(i), reg_w(i));
        printf("%s:\t0x%-8x\t%u\n", regsb[i], reg_b(i), reg_b(i));
        printf("%s:\t0x%-8x\t%u\n", regsb[i + 4], reg_b((i + 4)), reg_b((i + 4)));
        printf("\n");
    }
    printf("\n");
    for (; i <= R_EDI; ++i) {
        printf("%s:\t0x%-8x\t%u\n", regsl[i], reg_l(i), reg_l(i));
        printf("%s:\t0x%-8x\t%u\n", regsw[i], reg_w(i), reg_w(i));
        printf("\n");
    }
    printf("\n");
    printf("eip:\t0x%-8x\t%u\n", cpu.eip, cpu.eip);
    printf("ip:\t0x%-8x\t%u\n", cpu.ip, cpu.ip);
    printf("\n");
    printf("eflags:\t0x%-8x\t%u\n", cpu.eflags, cpu.eflags);
    printf("flags:\t0x%-8x\t%u\n", cpu.flags, cpu.flags);
}

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) {
    }
    else{
        printf("Unknown command '%s'\n", arg);
    }

    return 0;
}

```


将 cmd_info 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", "Single-step execution", cmd_si},
    { "info", "Show information about registers with argument 'r'", cmd_info},
    /* TODO: Add more commands */
};
```

执行命令 info r, si 5 后再次执行 info r。

```
(nemu) info r
eax:  0x718fd76      119078262
ax:    0xfd76        64886
al:    0x76          118
ah:    0xfd          253

ecx:   0x9c08638     163612216
cx:    0x8638        34360
cl:    0x38          56
ch:    0x86          134

edx:   0x118f213     18412051
dx:    0xf213        61971
dl:    0x13          19
dh:    0xf2          242

ebx:   0xa6457ed     174348269
bx:    0x57ed        22509
bl:    0xed          237
bh:    0x57          87

esp:   0x78382c04    2016947204
sp:    0x2c04        11268

ebp:   0x63e9be52    1676262994
bp:    0xbe52        48722

esi:   0x6acddc7d    1791876221
si:    0xdc7d        56445
```

```
edi:   0x7f232c27    2133011495
di:    0x2c27        11303

eip:   0x100000      1048576
ip:    0x0            0

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

```

(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: 0x1234          4660
ax: 0x1234           4660
al: 0x34             52
ah: 0x12             18

ecx: 0x100027        1048615
cx: 0x27             39
cl: 0x27             39
ch: 0x0              0

edx: 0x118f213       18412051
dx: 0xf213           61971
dl: 0x13             19
dh: 0xf2             242

ebx: 0x2              2
bx: 0x2              2
bl: 0x2              2
bh: 0x0              0

```

```

esp: 0x78382c04      2016947204
sp: 0x2c04           11268

ebp: 0x63e9be52      1676262994
bp: 0xbe52           48722

esi: 0x6acddc7d      1791876221
si: 0xdc7d           56445

edi: 0x7f232c27      2133011495
di: 0x2c27           11303

eip: 0x100017        1048599
ip: 0x17             23

eflags: 0x0          0
flags: 0x0           0

```

PA1.1.5 实现扫描内存功能

代码实现：

```

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
uint32_t addr = 0;
sscanf(arg2, "%x", &addr); //读取起始位置
if (n < 1){
    printf("Invalid arguments for x!\n");
    return 0;
}
int i;
uint32_t data;
printf("Address      Dword block\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\t\n", data);
}
return 0;
}
}

```

nemu 的内存通过在 `nemu/src/memory/memory.c` 中定义的大数组 `pmem` 来模拟。

`vaddr_read(vaddr_t addr, int len)` 函数用来访问模拟内存，前一个参数是要访问的虚拟地址，后一个是扫描长度。该函数会返回地址中的数据，类型为 `unsigned int`。

加入命令列表：

```

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 argumen"
        "t '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 hexa"
        "decimal form", cmd_x },
    /* TODO: Add more commands */
};

```

执行命令 `x 4 0x100000` 。

```

(nemu) x 4 0x100000
Address      Dword block
0x00100000   0x001234b8
0x00100004   0x0027b900
0x00100008   0x01890010
0x0010000c   0x0441c766

```

PA1.1.6 实现扫描内存字节单位显示

代码实现:

```
void byteSequence_dispaly(uint32_t data){
    printf("    ... ");
    uint32_t byte[4];
    byte[0] = data & 0x000000ff;
    byte[1] = (data & 0x0000ff00) >> 8;
    byte[2] = (data & 0x00ff0000) >> 16;
    byte[3] = (data & 0xff000000) >> 24;
    int i;
    for(i = 0; i < 4; i++){
        printf("%02x ",byte[i]);
    }
    printf("\n");
}

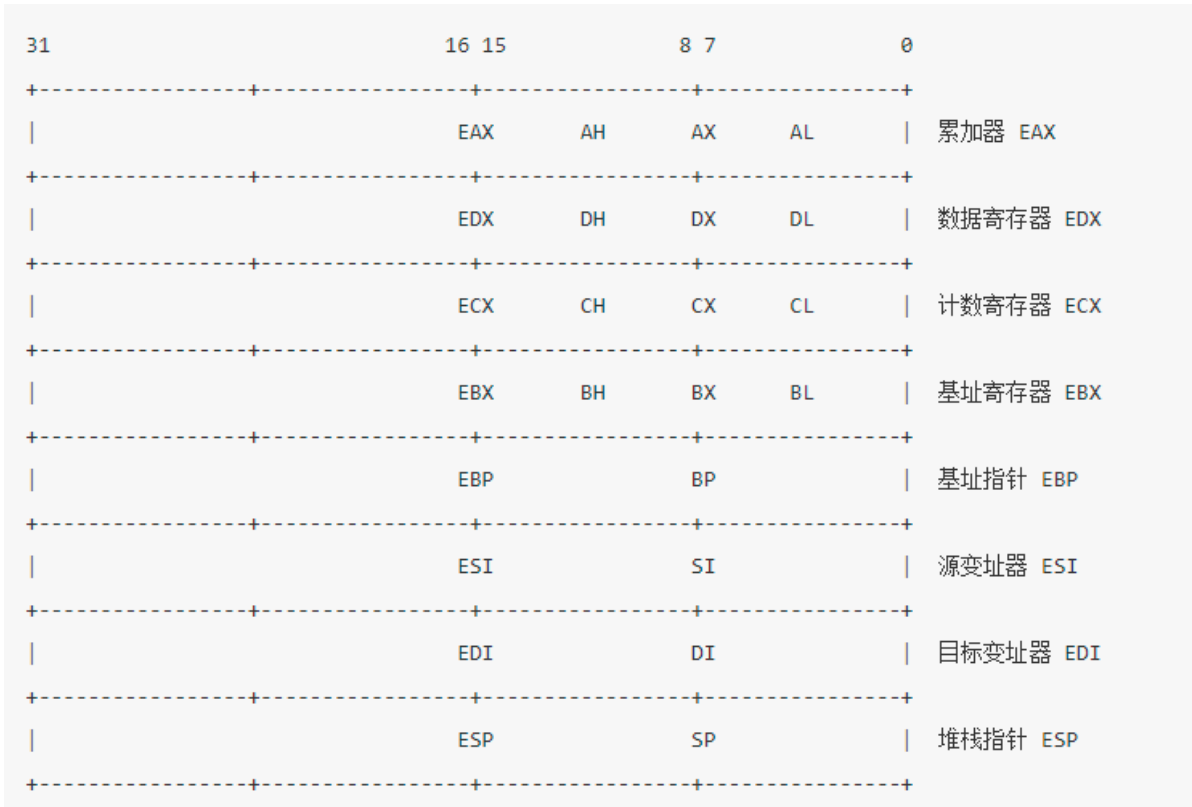
static int cmd_x(char *args){
    char *arg1 = strtok(NULL, " ");
    char *arg2 = strtok(NULL, " ");
    if (arg1 == NULL || arg2 == NULL) {
        printf("A parameter is missing!\n");
        return 0;
    }
    int n = atoi(arg1); //读取要读取的次数
    uint32_t addr = 0;
    sscanf(arg2, "%x", &addr); //读取起始位置
    if (n < 1){
        printf("Invalid arguments for x!\n");
        return 0;
    }
    int i;
    uint32_t data;
    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;
}
```

执行命令 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
```

遇到的问题及解决办法

在阅读讲义时，看到这个图非常不理解：为什么 `EAX` 在中间位置，而不是在 `31` 的地方，以为上方的数字对应其位数。



在请教学长之后知道了：

`EAX` 在中间是因为，它能表示的数据范围是0~31，它在哪个范围内就将其放置在哪个区间的中心位置；

再例如AH，它在8~15位之间，则它能表示相应的数据，也就是 `AX` 的高八位。

实验心得

- 此次实验让我大开眼界，了解到原来可以用C语言来模拟一台计算机，感觉非常神奇；
- 在阅读代码的同时也让我觉得，一个标准的C语言项目也是需要很深的功底去编写的，而且库函数也很丰富，之前自己在写C程序的时候也就是用了简单的几个库函数；
- 无符号数、位运算、匿名 `union` 与 `struct` 的搭配使用，这些都是在之前的程序设计课程上接触的较少的，这次实验也让我更加了解了它们的一些用法。

其他备注

无

