南京航空航天大学《计算机组成原理Ⅱ课程设计》报告

- 姓名：曹伟思
- 班级：1617302
- 学号：161730213
- 报告阶段：PA1.1
- 完成日期：2019.3.31
- 本次实验，我完成了所有内容。

目录

思考题

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？

如果存放指令本身那么就无法知道下一条指令在哪。而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。

贵圈真乱

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

虚拟机和模拟器的区别

模拟器最大的特点，就是对寄存器级别的硬件单元和芯片内部的时钟信号进行抽象仿真，通常是为了模拟不同指令集、不同体系架构的CPU，即异构指令集，所以多数情况要对微指令进行解释执行。

模拟器是用软件来模拟硬件操作，一般所说的虚拟机里会用模拟器来实现io设备的虚拟化（内存和CPU的虚拟化一般是由KVM或XEN这种虚拟化平台来实现的）。

而虚拟机(这里指的是VMware,Virtualbox)最大的特点就是，代码中没有一行被用作寄存器级或时钟级的建模，主要焦点都放在如何在架构相同的硬件中模拟另一个操作系统运行时的环境。也可以归类为：异构\同构操作系统和同构指令集的环境。virtualization基本都是去模拟一套相同指令集相同架构的硬件平台，因此在做好保护的前提下，很多时候可以直接利用CPU去执行目标指令。

从哪开始阅读代码呢?

从nemu/src/main.c文件开始阅读。

究竟要执行多久?

在cmd_c()函数中，调用cpu_exec()的时候传入了参数-1，你知道为什么要这么做吗，并说明理由。

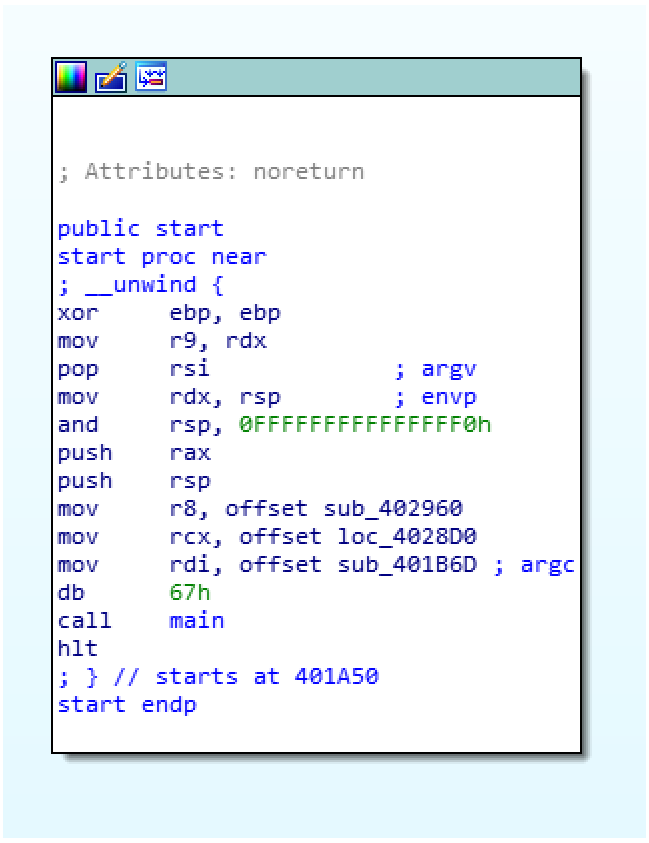
cpu_exec()的定义void cpu_exec(uint64_t n)，-1会被解释为uint64_t，也就是0xffffffffffffffff，所以就会cpu_exec()不断取指译码执行指令到程序结束。

谁来指示程序的结束?

在程序设计课上老师告诉你，当程序执行到main()函数返回处的时候,程序就退出了,你对此深信不疑。但你是否怀疑过，凭什么程序执行到main()函数的返回处就结束了？如果有人告诉你，程序设计课上老师的说法是错的，你有办法来证明/反驳吗?

一开始就不信，至少控制台不是我写的代码打开的（手动滑稽）。

以elf64为例，反汇编的结果表明操作系统在运行一个程序是会先新建一个进程并将程序加载到进程空间中，之后以elf64文件头的e_entry字段的值（一个虚拟地址）为程序入口，也就是start函数。



```
; Attributes: noreturn

public start
start proc near
; __unwind {
xor     ebp, ebp
mov     r9, rdx
pop     rsi             ; argv
mov     rdx, rsp        ; envp
and     rsp, 0FFFFFFFFFFFFFFF0h
push    rax
push    rsp
mov     r8, offset sub_402960
mov     rcx, offset loc_4028D0
mov     rdi, offset sub_401B6D ; argc
db      67h
call    main
hlt
; } // starts at 401A50
start endp
```

start函数初始化一部分参数后就调用main函数。C++中会在main函数之前先调用全局对象的构造函数，同理，结束之后调用全局对象的析构函数。

这些函数的调用取决于ELF文件中两个特殊的段。.init和.fini。

为什么会这样?

经过任务4.1和4.2，你发现本来是顺序存储的数据，为何以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?

这涉及存储对象内部的字节顺序，分为大端法（IBM和Oracle的大多数机器）和小端法（大多数Intel兼容机）。大端法即高字节数据存储在低地址中，低字节数据存放在高地址中。小端法与之相反。

实验环境下是小端法，而读取数据的代码pmem_rw(addr, uint32_t) & (~0u >> ((4 - len) << 3));是直接取uint32_t（小端法解释）返回。

实验内容

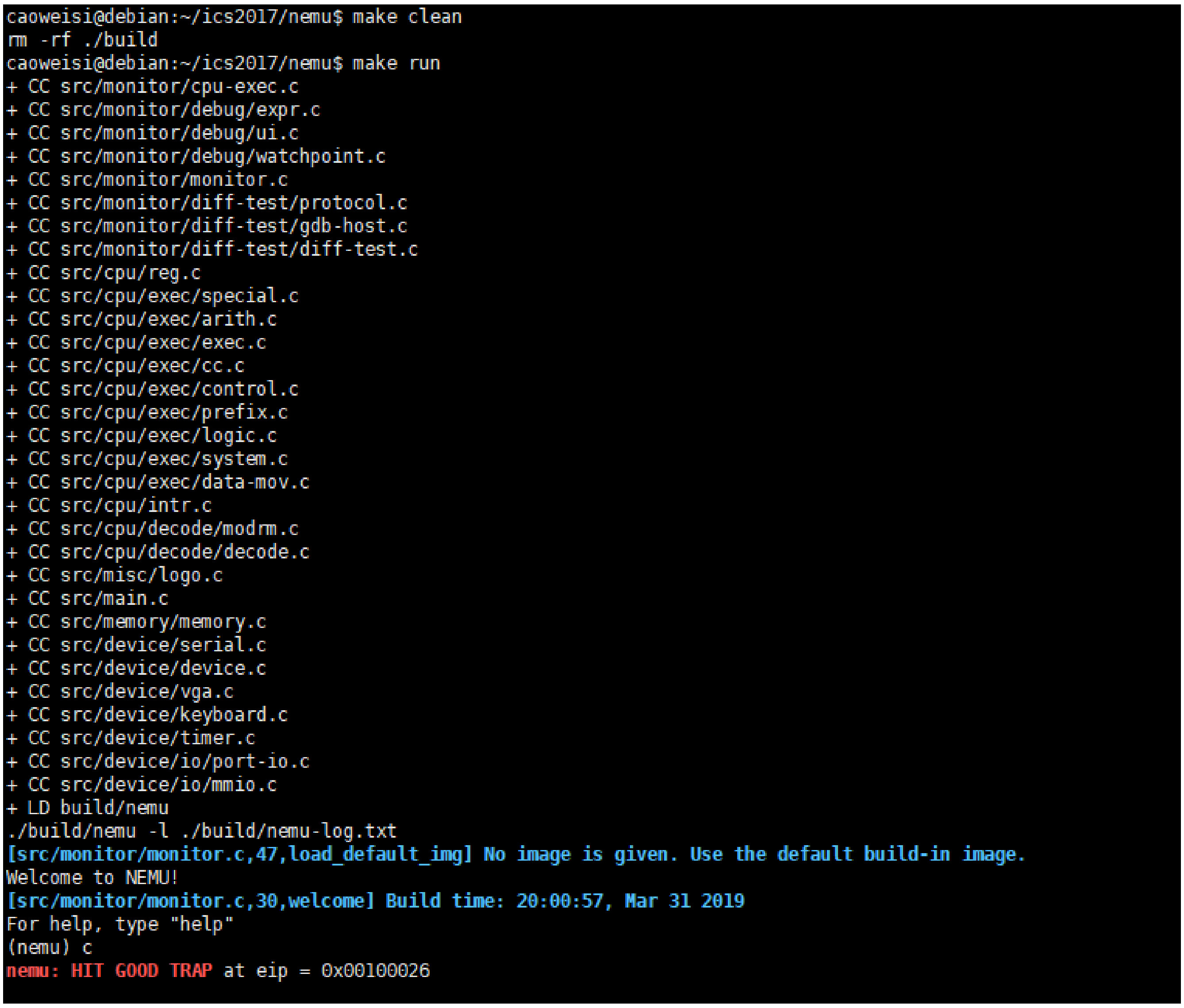
任务1：实现正确的寄存器结构体

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

```
typedef 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;
    rtlreg_t ecx;
    rtlreg_t edx;
    rtlreg_t ebx;
    rtlreg_t esp;
    rtlreg_t ebp;
    rtlreg_t esi;
    rtlreg_t edi;
    vaddr_t eip;
  };
} CPU_state;
```

通过union访问同一个内存单位的不同长度的数据。通过union和匿名struct使gpr[8]和8个通用寄存器的内存单位一一对应。

先make clean，然后make run并c。



```
caoweisi@debian:~/ics2017/nemu$ make clean
rm -rf ./build
caoweisi@debian:~/ics2017/nemu$ make run
+ CC src/monitor/cpu-exec.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/monitor.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/cpu/reg.c
+ CC src/cpu/exec/special.c
+ CC src/cpu/exec/arith.c
+ CC src/cpu/exec/exec.c
+ CC src/cpu/exec/cc.c
+ CC src/cpu/exec/control.c
+ CC src/cpu/exec/prefix.c
+ CC src/cpu/exec/logic.c
+ CC src/cpu/exec/system.c
+ CC src/cpu/exec/data-mov.c
+ CC src/cpu/intr.c
+ CC src/cpu/decode/modrm.c
+ CC src/cpu/decode/decode.c
+ CC src/misc/logo.c
+ CC src/main.c
+ CC src/memory/memory.c
+ CC src/device/serial.c
+ CC src/device/device.c
+ CC src/device/vga.c
+ CC src/device/keyboard.c
+ CC src/device/timer.c
+ CC src/device/io/port-io.c
+ CC src/device/io/mmio.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: 20:00:57, Mar 31 2019
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100026
```

任务2.1：实现单步/指定步数执行功能

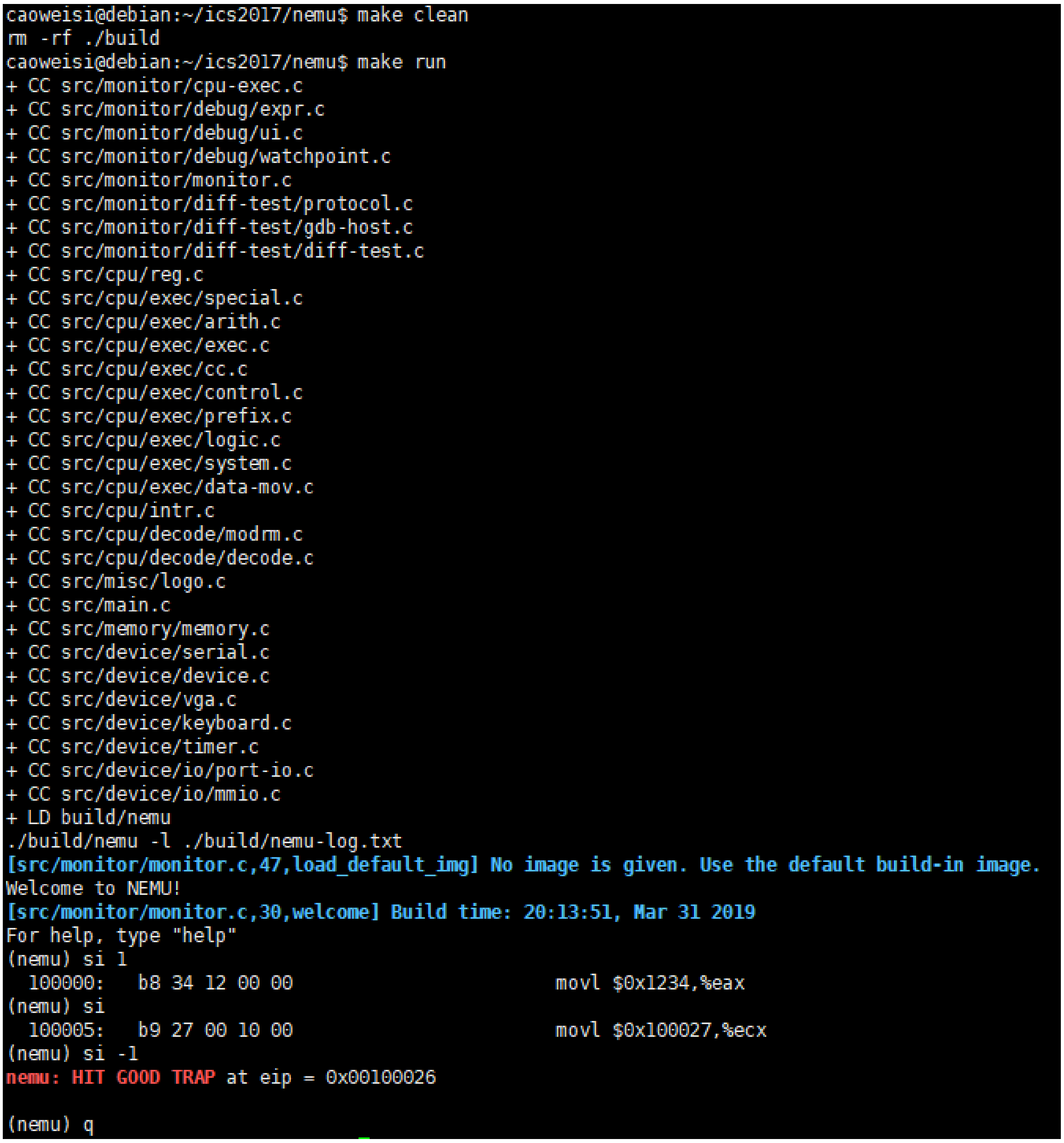
修改cmd_table（这里存储的使用来索引的指令信息或handle函数）。

```
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", "si [N]:Single-step execution N", cmd_si },
};
```

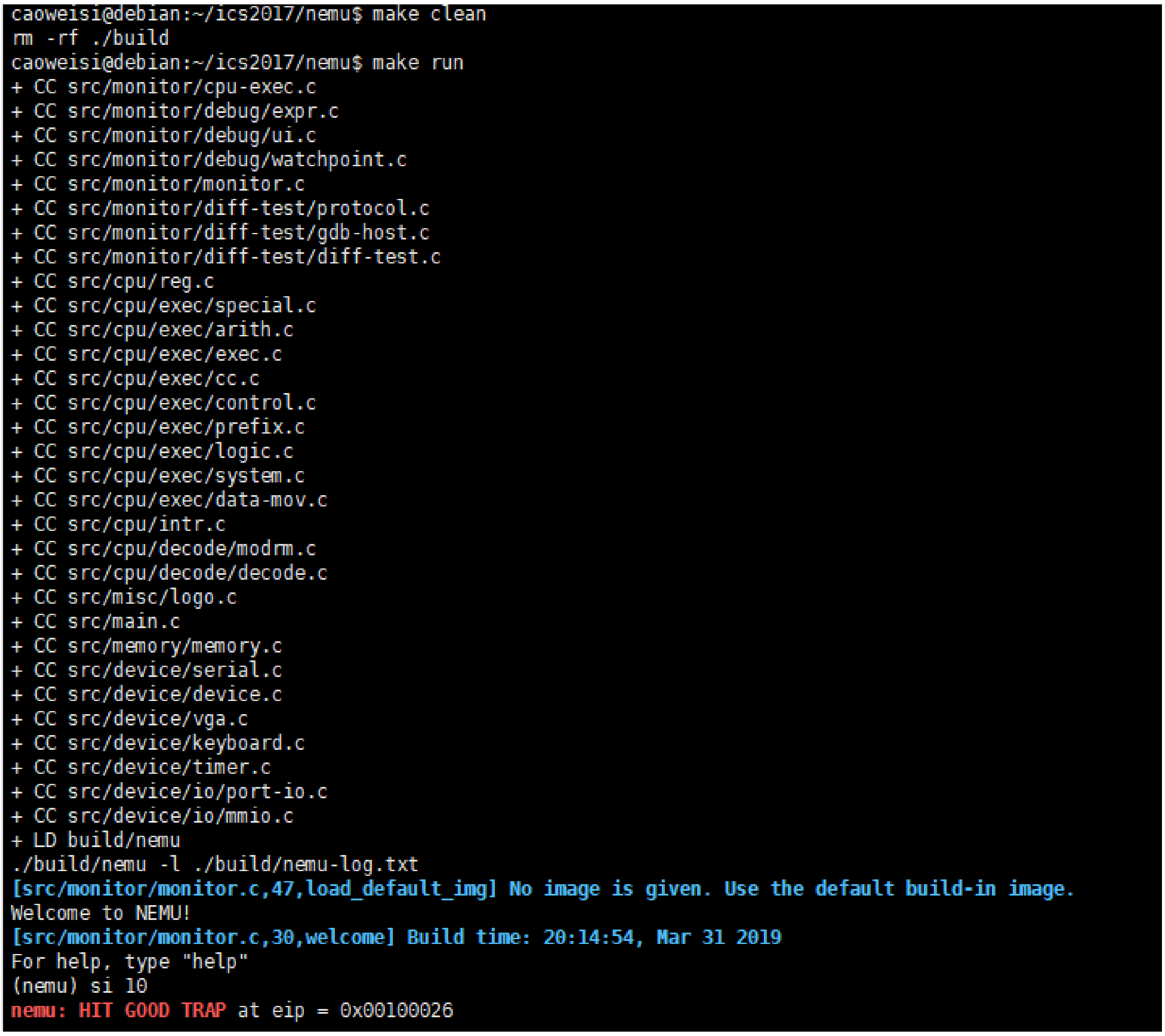
声明并定义函数cmd_si。

```
static int cmd_si(char *args);

...

static int cmd_si(char *args){
  char *arg = strtok(NULL, " ");
  int N;

  if (arg == NULL) {
    N = 1;
  }
  else {
    N = atoi(arg);
  }

  cpu_exec(N);

  return 0;
}
```

先make clean，然后make run并si 1，si，si -1，si 10。



```
caoweisi@debian:~/ics2017/nemu$ make clean
rm -rf ./build
caoweisi@debian:~/ics2017/nemu$ make run
+ CC src/monitor/cpu-exec.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/monitor.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/cpu/reg.c
+ CC src/cpu/exec/special.c
+ CC src/cpu/exec/arith.c
+ CC src/cpu/exec/exec.c
+ CC src/cpu/exec/cc.c
+ CC src/cpu/exec/control.c
+ CC src/cpu/exec/prefix.c
+ CC src/cpu/exec/logic.c
+ CC src/cpu/exec/system.c
+ CC src/cpu/exec/data-mov.c
+ CC src/cpu/intr.c
+ CC src/cpu/decode/modrm.c
+ CC src/cpu/decode/decode.c
+ CC src/misc/logo.c
+ CC src/main.c
+ CC src/memory/memory.c
+ CC src/device/serial.c
+ CC src/device/device.c
+ CC src/device/vga.c
+ CC src/device/keyboard.c
+ CC src/device/timer.c
+ CC src/device/io/port-io.c
+ CC src/device/io/mmio.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: 20:13:51, Mar 31 2019
For help, type "help"
(nemu) si 1
  100000:   b8 34 12 00 00                        movl $0x1234,%eax
(nemu) si
  100005:   b9 27 00 10 00                        movl $0x100027,%ecx
(nemu) si -1
nemu: HIT GOOD TRAP at eip = 0x00100026

(nemu) q
```



```
caoweisi@debian:~/ics2017/nemu$ make clean
rm -rf ./build
caoweisi@debian:~/ics2017/nemu$ make run
+ CC src/monitor/cpu-exec.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/monitor.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/cpu/reg.c
+ CC src/cpu/exec/special.c
+ CC src/cpu/exec/arith.c
+ CC src/cpu/exec/exec.c
+ CC src/cpu/exec/cc.c
+ CC src/cpu/exec/control.c
+ CC src/cpu/exec/prefix.c
+ CC src/cpu/exec/logic.c
+ CC src/cpu/exec/system.c
+ CC src/cpu/exec/data-mov.c
+ CC src/cpu/intr.c
+ CC src/cpu/decode/modrm.c
+ CC src/cpu/decode/decode.c
+ CC src/misc/logo.c
+ CC src/main.c
+ CC src/memory/memory.c
+ CC src/device/serial.c
+ CC src/device/device.c
+ CC src/device/vga.c
+ CC src/device/keyboard.c
+ CC src/device/timer.c
+ CC src/device/io/port-io.c
+ CC src/device/io/mmio.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: 20:14:54, Mar 31 2019
For help, type "help"
(nemu) si 10
nemu: HIT GOOD TRAP at eip = 0x00100026
```

任务2.2：修改一次打印步数上限

看cpu_exec函数，发现实际运行使用的是exec_wrapper(print_flag)。

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

参数print_flag初始化bool print_flag = n < MAX_INSTR_TO_PRINT;。

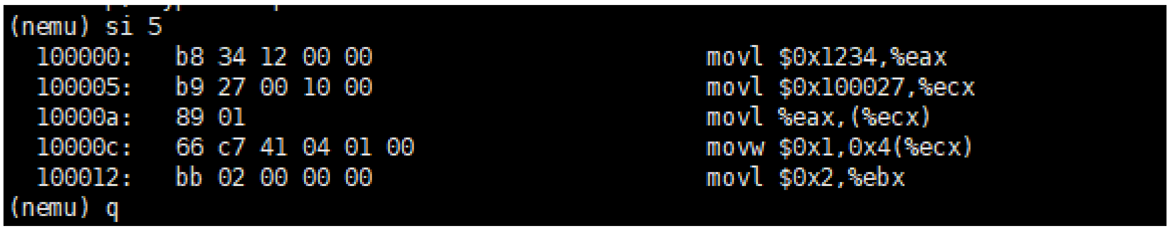
先看exec_wrapper函数定义发现打印语句。

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

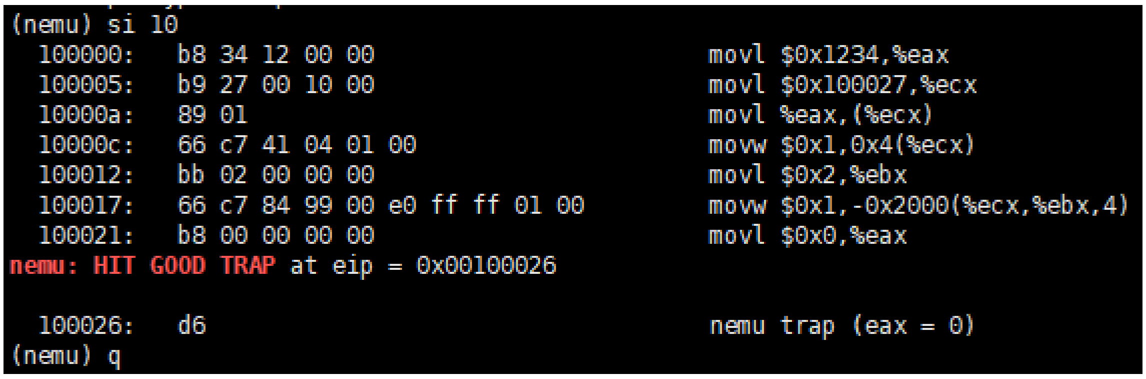
所以是否打印由参数print_flag控制，其值为n < MAX_INSTR_TO_PRINT。所以修改上限即修改MAX_INSTR_TO_PRINT。

```
#define MAX_INSTR_TO_PRINT -1
```

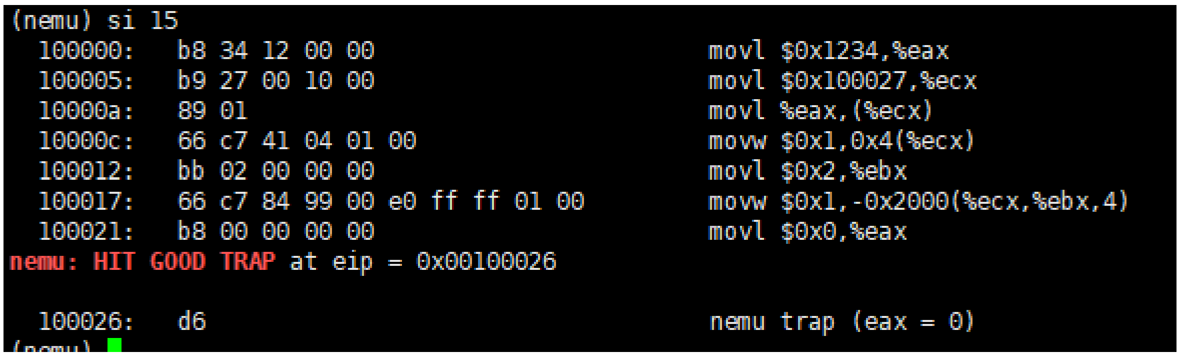
先make clean，然后make run并si 5，si 10，si 15。



```
(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) q
```



```
(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) q
```



```
(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)
```

任务3：实现打印寄存器功能

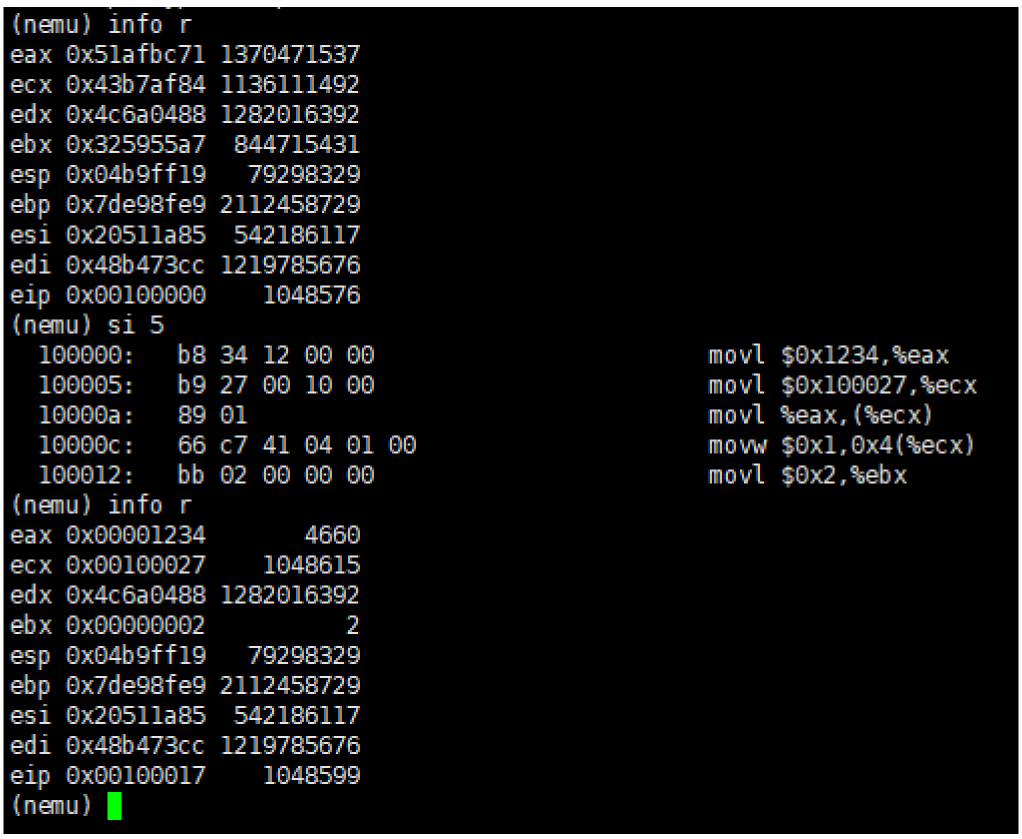
修改cmd_table。

```
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", "si [N]:Single-step execution N", cmd_si },
  { "info", "info r:print register status,info w:monitoring point information", cmd_info },
};
```

声明并定义函数cmd_info。

```
static int cmd_info(char *args);

...

static int cmd_info(char *args){
  char *arg = strtok(NULL, " ");

  if (arg == NULL) {
    printf("info r:print register status,info w:monitoring point information\n");
  }
  else if (strcmp(arg,"r") == 0) {
    printf("eax %#010x %10d\n", cpu.eax, cpu.eax);
    printf("ecx %#010x %10d\n", cpu.ecx, cpu.ecx);
    printf("edx %#010x %10d\n", cpu.edx, cpu.edx);
    printf("ebx %#010x %10d\n", cpu.ebx, cpu.ebx);
    printf("esp %#010x %10d\n", cpu.esp, cpu.esp);
    printf("ebp %#010x %10d\n", cpu.ebp, cpu.ebp);
    printf("esi %#010x %10d\n", cpu.esi, cpu.esi);
    printf("edi %#010x %10d\n", cpu.edi, cpu.edi);
    printf("eip %#010x %10d\n", cpu.eip, cpu.eip);
  }
  else if (strcmp(arg,"w") == 0) {
    printf("'%s' is not finished\n", arg);
  }
  else {
    printf("Unknown command '%s'\n", arg);
  }

  return 0;
}
```

先make clean，然后make run并info r，si 5，info r。



```
(nemu) info r
eax 0x51afbc71 1370471537
ecx 0x43b7af84 1136111492
edx 0x4c6a0488 1282016392
ebx 0x325955a7  844715431
esp 0x04b9ff19   79298329
ebp 0x7de98fe9 2112458729
esi 0x20511a85  542186117
edi 0x48b473cc 1219785676
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 0x4c6a0488 1282016392
ebx 0x00000002          2
esp 0x04b9ff19   79298329
ebp 0x7de98fe9 2112458729
esi 0x20511a85  542186117
edi 0x48b473cc 1219785676
eip 0x00100017    1048599
(nemu)
```

任务4.1：实现扫描内存功能 && 任务4.2：转换为字节显示

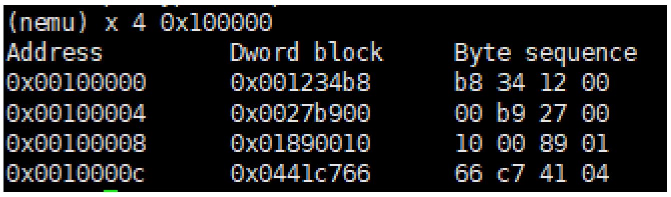
修改cmd_table。

```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", "si [N]:Single-step execution N", cmd_si },
  { "info", "info r:print register status,info w:monitoring point information", cmd_info },
  { "x", "x N EXPR:Find the value of the expression EXPR, use the result as the starting
memory address, and output N consecutive 4 bytes in hexadecimal form", cmd_x },
};
```

声明并定义函数cmd_x和用于解析表达式的函数parse_hex（未完善）。

```c
static int cmd_x(char *args);

static vaddr_t parse_hex(char *args);

...

static vaddr_t parse_hex(char *arg){
  vaddr_t i = 0;
  char * end = arg + strlen(arg);

  for (char * x = arg + 2; x < end; x++) {
    if (*x >= '0' && *x <= '9') {
      i *= 0x10;
      i += *x - 0x30;
    }
    else if (*x >= 'a' && *x <= 'f') {
      i *= 0x10;
      i += *x - 0x57;
    }
    else if (*x >= 'A' && *x <= 'F') {
      i *= 0x10;
      i += *x - 0x37;
    }
    else {
      printf("EXPR error\n");
      return i;
    }
  }

  return i;
}

static int cmd_x(char *args){
  char *arg = strtok(NULL, " ");
  int N = 0;
  vaddr_t addr = 0;

  if (arg == NULL) {
    printf("N not found\n");
    return 0;
  }
  else {
    N = atoi(arg);
  }
  arg = strtok(NULL, " ");

  if (arg == NULL) {
    printf("EXPR not found\n");
    return 0;
  }
  else {
    addr = parse_hex(arg);
    printf("Address         Dword block     Byte sequence\n");
    while (N-- > 0) {
      uint32_t value = vaddr_read(addr, 4);
      uint32_t byte[4];
      byte[0] = value & 0x000000ff;
      byte[1] = (value & 0x0000ff00) >> 8;
      byte[2] = (value & 0x00ff0000) >> 16;
      byte[3] = (value >> 24) % 0x100;
      printf("%#010x      %#010x      %02x %02x %02x %02x\n", addr, value, byte[0], byte[1],
byte[2], byte[3]);
      addr += 4;
    }
  }

  return 0;
}
```

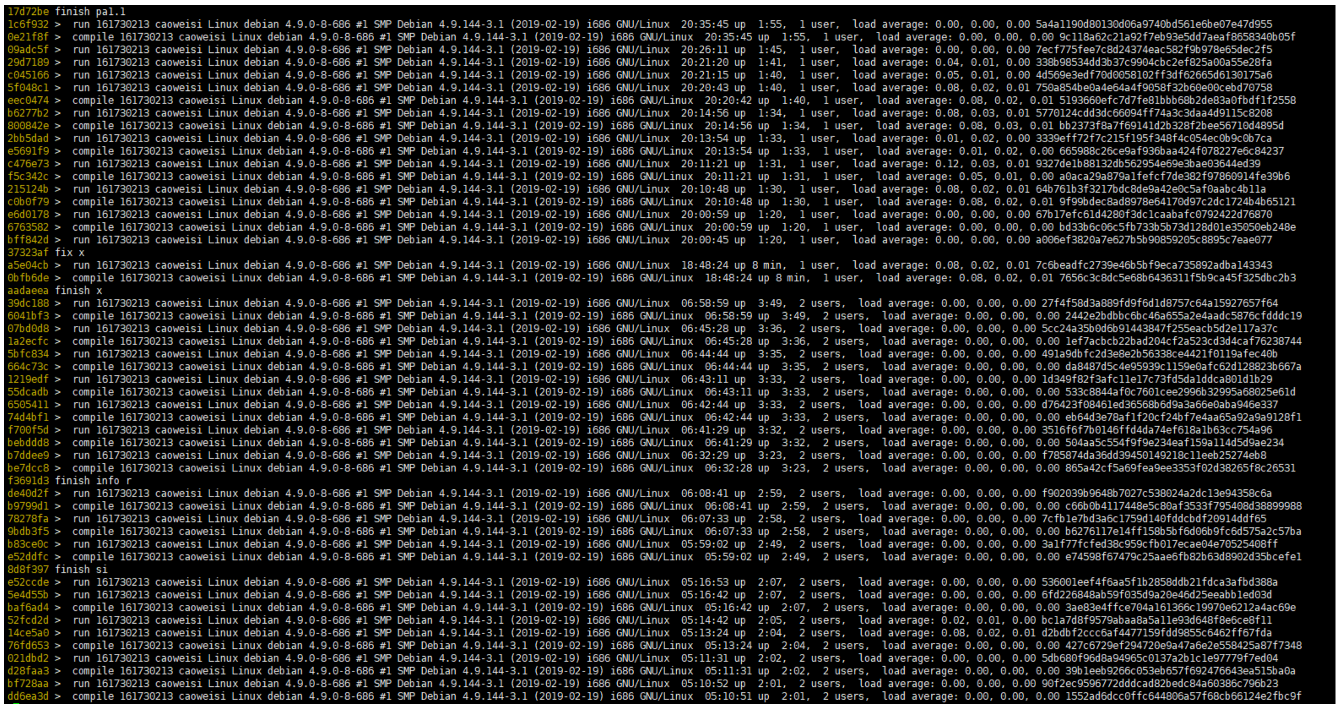
先make clean，然后make run并x 4 0x100000。



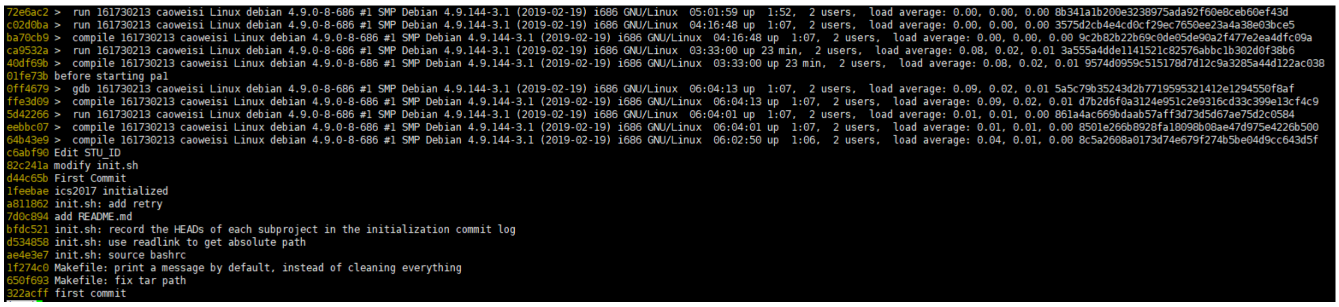
```
(nemu) x 4 0x100000
Address         Dword block     Byte sequence
0x00100000      0x001234b8      b8 34 12 00
0x00100004      0x0027b900      00 b9 27 00
0x00100008      0x01890010      10 00 89 01
0x0010000c      0x0441c766      66 c7 41 04
```

Git Log

git log --oneline截图。



```
17d72be finish pa1.1
1c6f932 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:35:45 up  1:55,  1 user,  load average: 0.00, 0.00, 0.00 5a4a1190d80130d06a9740bd561e6be07e47d955
0e21f8f >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:35:45 up  1:55,  1 user,  load average: 0.00, 0.00, 0.00 9c118a62c21a92f7eb93e5dd7aeaf8658340b05f
09adc5f >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:26:11 up  1:45,  1 user,  load average: 0.00, 0.00, 0.00 7ecf775fee7c8d24374eac582f9b978e65dec2f5
29d7189 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:21:20 up  1:41,  1 user,  load average: 0.04, 0.01, 0.00 338b98534dd3b37c9904cbc2ef825a00a55e28fa
c045166 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:21:15 up  1:40,  1 user,  load average: 0.05, 0.01, 0.00 4d569e3edf70d0058102ff3df62665d6130175a6
5f048c1 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:20:43 up  1:40,  1 user,  load average: 0.08, 0.02, 0.01 750a854be0a4e64a4f9058f32b60e00cebd70758
eec0474 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:20:42 up  1:40,  1 user,  load average: 0.08, 0.02, 0.01 5193660efc7d7fe81bbb68b2de83a0fbdf1f2558
b6277b2 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:14:56 up  1:34,  1 user,  load average: 0.08, 0.03, 0.01 5770124cdd3dc66094ff74a3c3daa4d9115c8208
800842e >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:14:56 up  1:34,  1 user,  load average: 0.08, 0.03, 0.01 bb2373f8a7f69141d2b328f2bee56710d4895d
2bb5dad >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:13:54 up  1:33,  1 user,  load average: 0.01, 0.02, 0.00 3339eff72f7c215f195f348f4c054ec0b9c0b7ca
e5691f9 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:13:54 up  1:33,  1 user,  load average: 0.01, 0.02, 0.00 665988c26ce9af936baa424f078227e6c84237
c476e73 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:11:21 up  1:31,  1 user,  load average: 0.12, 0.03, 0.01 9327de1b88132db562954e69e3bae03644ed39
f5c342c >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:11:21 up  1:31,  1 user,  load average: 0.05, 0.01, 0.00 a0aca29a879a1fefcf7de382f97860914fe39b6
215124b >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:10:48 up  1:30,  1 user,  load average: 0.08, 0.02, 0.01 64b761b3f3217bdc8de9a42e0c5af0aabc4b11a
c0b0f79 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:10:48 up  1:30,  1 user,  load average: 0.08, 0.02, 0.01 9f99bdec8ad8978e64170d97c2dc1724b4b65121
e6d0178 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:00:59 up  1:20,  1 user,  load average: 0.00, 0.00, 0.00 67b17efc61d4280f3dc1caabafc0792422d76870
6763582 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:00:59 up  1:20,  1 user,  load average: 0.00, 0.00, 0.00 bd33b6c06c5fb733b5b73d128d01e35050eb248e
bff842d >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  20:00:45 up  1:20,  1 user,  load average: 0.00, 0.00, 0.00 a006ef3820a7e627b5b90859205c8895c7eae077
37323af fix x
a5e04cb >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  18:48:24 up 8 min,  1 user,  load average: 0.08, 0.02, 0.01 7c6beadfc2739e46b5bf9eca735892adba143343
0bfb6de >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  18:48:24 up 8 min,  1 user,  load average: 0.08, 0.02, 0.01 7656c3c8dc5e68b6436311f5b9ca45f325dbc2b3
aadaeea finish x
39dc188 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:58:59 up  3:49,  2 users,  load average: 0.00, 0.00, 0.00 27f4f58d3a889fd9f6d1d8757c64a15927657f64
6041bf3 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:58:59 up  3:49,  2 users,  load average: 0.00, 0.00, 0.00 2442e2bdbbc6bc46a655a2e4aadc5876cfdddc19
07bd0d8 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:45:28 up  3:36,  2 users,  load average: 0.00, 0.00, 0.00 5cc24a35b0d6b91443847f255eacb5d2e117a37c
1a2ecfc >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:45:28 up  3:36,  2 users,  load average: 0.00, 0.00, 0.00 1ef7acbcb22bad204cf2a523cd3d4caf76238744
5bfc834 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:44:44 up  3:35,  2 users,  load average: 0.00, 0.00, 0.00 491a9dbfc2d3e8e2b56338ce4421f0119afec40b
664c73c >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:44:44 up  3:35,  2 users,  load average: 0.00, 0.00, 0.00 da8487d5c4e95939c1159e0afc62d128823b667a
1219edf >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:43:11 up  3:33,  2 users,  load average: 0.00, 0.00, 0.00 1d349f82f3afc11e17c73fd5da1ddca801d1b29
55dcadb >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:43:11 up  3:33,  2 users,  load average: 0.00, 0.00, 0.00 533c8844af0c7601cee2996b32995a68025e61d
6505411 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:42:44 up  3:33,  2 users,  load average: 0.00, 0.00, 0.00 d76423f08461ed36568b6d9a3a66e0aba946e337
74d4bf1 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:42:44 up  3:33,  2 users,  load average: 0.00, 0.00, 0.00 eb64d3e78af1f20cf24bf7e4aa65a92a9a9128f1
f700f5d >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:41:29 up  3:32,  2 users,  load average: 0.00, 0.00, 0.00 3516f6f7b0146ffd4da74ef618a1b63cc754a96
bebddd8 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:41:29 up  3:32,  2 users,  load average: 0.00, 0.00, 0.00 504aa5c554f9f9e234eaf159a114d5d9ae234
b7ddee9 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:32:29 up  3:23,  2 users,  load average: 0.00, 0.00, 0.00 f785874da36dd39450149218c11eeb25274eb8
be7dcc8 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:32:28 up  3:23,  2 users,  load average: 0.00, 0.00, 0.00 865a42cf5a69fea9ee3353f02d38265f8c26531
f3691d3 finish info r
de40d2f >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:08:41 up  2:59,  2 users,  load average: 0.00, 0.00, 0.00 f902039b9648b7027c538024a2dc13e94358c6a
b9799d1 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:08:41 up  2:59,  2 users,  load average: 0.00, 0.00, 0.00 c66b0b4117448e5c80af3533f795408d38899988
78278fa >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:07:33 up  2:58,  2 users,  load average: 0.00, 0.00, 0.00 7cfb1e7bd3a6c1759d140fddcbdf20914ddf65
9bdb3f5 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:07:33 up  2:58,  2 users,  load average: 0.00, 0.00, 0.00 b6276117e14ff158b5bf6d06b9fc6d575a2c57ba
b83ce0c >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:59:02 up  2:49,  2 users,  load average: 0.00, 0.00, 0.00 3a1f77fcfed38c959cfb017ecae04e70525408ff
e52ddfc >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:59:02 up  2:49,  2 users,  load average: 0.00, 0.00, 0.00 e74598f67479c25aae6fb82b63d8902d35bcefe1
8d8f397 finish si
e52ccde >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:16:53 up  2:07,  2 users,  load average: 0.00, 0.00, 0.00 536001eef4f6aa5f1b2858ddb21fdca3afbd388a
5e4d55b >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:16:42 up  2:07,  2 users,  load average: 0.00, 0.00, 0.00 6fd226848ab59f035d9a20e46d25eeabb1ed03d
baf6ad4 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:16:42 up  2:07,  2 users,  load average: 0.00, 0.00, 0.00 3ae83e4ffce704a161366c19970e6212a4ac69e
52fcd2d >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:14:42 up  2:05,  2 users,  load average: 0.02, 0.01, 0.00 bc1a7d8f9579abaa8a5a11e93d648f8e6ce8f11
14ce5a0 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:13:24 up  2:04,  2 users,  load average: 0.08, 0.02, 0.01 d2bdbf2ccc6af4477159fdd9855c6462ff67fda
76fd653 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:13:24 up  2:04,  2 users,  load average: 0.00, 0.00, 0.00 427c6729ef294720e9a47a6e2e558425a87f7348
021dbd2 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:11:31 up  2:02,  2 users,  load average: 0.00, 0.00, 0.00 5db680f96d8a94965c0137a2b1c1e97779f7ed04
d28faa3 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:11:31 up  2:02,  2 users,  load average: 0.00, 0.00, 0.00 39b1eeb9266c053eb657f692476643ea515ba0a
bf728aa >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:10:52 up  2:01,  2 users,  load average: 0.00, 0.00, 0.00 90f2ec9596772dddcad82bedc84a60386c796b23
dd6ea3d >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:10:51 up  2:01,  2 users,  load average: 0.00, 0.00, 0.00 1552ad6dcc0ffc644806a57f68cb66124e2fbc9f
```



```
72e6ac2 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  05:01:59 up  1:52,  2 users,  load average: 0.00, 0.00, 0.00 8b341a1b200e3238975ada92f60e8ceb60ef43d
c02d0ba >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  04:16:48 up  1:07,  2 users,  load average: 0.00, 0.00, 0.00 3575d2cb4e4cd0cf29ec7650ee23a4a38e03bce5
ba70cb9 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  04:16:48 up  1:07,  2 users,  load average: 0.00, 0.00, 0.00 9c2b82b22b69c0de05de90a2f477e2ea4dfc09a
ca9532a >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  03:33:00 up 23 min,  2 users,  load average: 0.08, 0.02, 0.01 3a555a4dde1141521c82576abbc1b302d0f38b6
40df69b >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  03:33:00 up 23 min,  2 users,  load average: 0.08, 0.02, 0.01 9574d0959c515178d7d12c9a3285a44d122ac038
01fe73b before starting pal
0ff4679 >  gdb 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:04:13 up  1:07,  2 users,  load average: 0.09, 0.02, 0.01 5a5c79b35243d2b7719595321412e1294550f8af
ffe3d09 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:04:13 up  1:07,  2 users,  load average: 0.09, 0.02, 0.01 d7b2d6f0a3124e951c2e9316cd33c399e13cf4c9
5d42266 >  run 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:04:01 up  1:07,  2 users,  load average: 0.01, 0.01, 0.00 861a4ac669bdaab57aff3d73d5d67ae75d2c0584
eebbc07 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:04:01 up  1:07,  2 users,  load average: 0.01, 0.01, 0.00 8501e266b8928fa18098b08ae47d975e4226b500
64b43e9 >  compile 161730213 caoweisi Linux debian 4.9.0-8-686 #1 SMP Debian 4.9.144-3.1 (2019-02-19) i686 GNU/Linux  06:02:50 up  1:06,  2 users,  load average: 0.04, 0.01, 0.00 8c5a2608a0173d74e679f274b5be04d9cc643d5f
c6abf90 Edit STU_ID
82c241a modify init.sh
d44c65b First Commit
1feebae ics2017 initialized
a811862 init.sh: add retry
7d0c894 add README.md
bfdc521 init.sh: record the HEADs of each subproject in the initialization commit log
d534858 init.sh: use readlink to get absolute path
ae4e3e7 init.sh: source bashrc
1f274c0 Makefile: print a message by default, instead of cleaning everything
650f693 Makefile: fix tar path
322acff first commit
```

遇到的问题及解决办法

parse_hex函数字符串处理时多处理了一位，导致程序报错，修改后解决

实验心得

深入理解计算机系统中的信息的表示和处理

其他备注

无