南京航空航天大学 计算机科学与技术系学院 计算机组成原理 课程实验

学号：161630220

姓名：赵维康

PA2- 简单复杂的机器： 冯诺依曼计算机系统

程序， 运行时环境与 AM
现代指令系统
思考题：什么是 AM?
答：抽象机器。
思考题：堆和栈在哪里?
程序运行时刻用到的堆和栈又是怎么来的?
答：通过存储器映像方式，映射到虚拟存储空间上。
在实现具体相关指令的执行函数之前，先进入 nemu/src/cpu/exec/all-instr.h 文件，对要实现的每个指令的执行函数进行声明，以免后面运行 dummy 时出现函数未声明的错误，以下是具体实现

```
#include "cpu/exec.h"

make_EHelper(mov);

make_EHelper(movzx);

make_EHelper(movsx);

make_EHelper(cltd);

make_EHelper(call);

make_EHelper(ret);

make_EHelper(push);

make_EHelper(pop);

make_EHelper(leave);

make_EHelper(sub);

make_EHelper(sbb);
```

```
make_EHelper(add);
make_EHelper(div);
make_EHelper(idiv);
make_EHelper(adc);
make_EHelper(inc);
make_EHelper(dec);
make_EHelper(neg);
make_EHelper(mul);
make_EHelper(imul1);
make_EHelper(imul2);
make_EHelper(imul3);

make_EHelper(xor);
make_EHelper(and);
```

```
make_EHelper(or);
make_EHelper(shl);
make_EHelper(shr);
make_EHelper(sar);
make_EHelper(rol);
make_EHelper(not);

make_EHelper(cmp);

make_EHelper(lea);

make_EHelper(jmp);
make_EHelper(jcc);

make_EHelper(test);
```

```
make_EHelper(nop);

make_EHelper(setcc);


make_EHelper(operand_size);


make_EHelper(inv);

make_EHelper(nemu_trap);
```

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
[sudo] zhaoweikang 的密码：
+ CC src/cpu/decode/modrm.c
+ CC src/cpu/decode/decode.c
+ CC src/cpu/exec/cc.c
+ CC src/cpu/exec/arith.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/exec/exec.c
+ CC src/cpu/exec/special.c
+ CC src/cpu/intr.c
+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make run
./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: 22:30:21, Apr 18 2018
For help, type "help"
(nemu)
```

说明实现没有错误

下面进入 nemu/src/exec/data-mov.c 文件，实现 leave 指令、cltd 指令以及 lea 指令的执行函数，以下是具体实现

```
make_EHelper(leave) {
  rtl_mv(&cpu.esp,&cpu.ebp);
  rtl_pop(&cpu.ebp);

  print_asm("leave");
}
```

leave：此指令实现的功能为 movl　%ebp,%esp; pop　%ebp;调用使用 RTL 实现的 rtl_mv(),以及 rtl_pop()即可实现所述功能，调用 print_asm()打印反汇编指令 leave

```
make_EHelper(cltd) {
  if (decoding.is_operand_size_16) {
   rtl_lr(&t0,R_AX,2);
          if((int32_t)(int16_t)(uint16_t)t0 < 0) {
                 rtl_addi(&t1,&tzero,0xffff);
                 rtl_sr(R_DX,&t1);
          }
          else {
                 rtl_sr(R_DX,&tzero);
          }
  }
  else {
    rtl_lr(&t0,R_EAX,4);
                if((int32_t)t0 < 0) {
                       rtl_addi(&t1,&tzero,0xffffffff);
                       rtl_sr(R_EDX,4,&t1);
                }
                else {
                         rtl_sr(R_EDX,4,&tzero);
  }

  print_asm(decoding.is_operand_size_16 ? "cwtl" : "cltd");
}
```

cltd：查 i386 手册知，cltd 指令的操作为：若操作数大小为 16 位，则使用 rtl_lr()读取带宽度寄存器 AX 中的内容，若 AX<0，则使用 rtl_sr()将带宽度的寄存器 DX 设置为 0xffff，否则设置为 0；32 位操作数同理，只不过，将 AX->EAX,DX->EDX,0xffff->0xffffffff。

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
+ CC src/cpu/exec/data-mov.c

+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo 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: 10:42:15, Apr 15 2018
For help, type "help"
(nemu)
```

说明实现没有错误

下面进入 nemu/stc/exec/arith.c 文件，完成对 add 指令、cmp 指令、inc 指令、dec 指令、neg 指令的执行函数的编写，下面是具体实现

```
make_EHelper(add) {
  rtl_add(&t2, &id_dest->val, &id_src->val);

  operand_write(id_dest, &t2);

  rtl_update_ZFSF(&t2, id_dest->width);

  rtl_sltu(&t0, &t2, &id_dest->val);

  rtl_set_CF(&t0);

  rtl_xor(&t0, &id_dest->val, &id_src->val);

  rtl_not(&t0);

  rtl_xor(&t1, &id_dest->val, &t2);

  rtl_and(&t0, &t0, &t1);

  rtl_msb(&t0, &t0, id_dest->width);

  rtl_set_OF(&t0);


  print_asm_template2(add);
}
```

add:add 指令的操作为两个操作数相加，结果放在目的操作数所在寄存器，然后写回内存，同时更新 **ZF** 与 **SF** 位，然后根据无符号数和带符号数分别设置 **CF** 位与 **OF** 位。

```
make_EHelper(cmp) {
  rtl_sub(&t2,&id_dest->val,&id_src->val);
  rtl_update_ZFSF(&t2,id_dest->width);

  rtl_sltu(&t0,&id_dest->val,&t2);
  rtl_set_CF(&t0);

  rtl_xor(&t0,&id_dest->val,&id_src->val);
        rtl_xor(&t1,&id_dest->val,&t2);
  rtl_and(&t0,&t0,&t1);
  rtl_msb(&t0,&t0,id_dest->width);
  rtl_set_OF(&t0);

  print_asm_template2(cmp);
}
```

cmp:cmp 指令的操作为利用 **sub** 指令求两个操作数的差，结果不写会目的操作数寄存器，而是来设置 **FLAG** 位。同时更新 **ZF** 与 **SF** 位，然后根据无符号数和带符号数分别设置 **CF** 位与 **OF** 位。

```
make_EHelper(inc) {
  t0 = 1;
  rtl_add(&t2,&id_dest->val,&t0);
  operand_write(id_dest,&t2);

  rtl_set_ZFSF(&t2,id_dest->width);

  rtl_xor(&t0,&id_dest->val,&id_src->val);
  rtl_not(&t0);
        rtl_xor(&t1,&id_dest->val,&t2);
  rtl_and(&t0,&t0,&t1);
  rtl_msb(&t0,&t0,id_dest->width);
  rtl_set_OF(&t0);

  print_asm_template1(inc);
}
```

inc:此指令用于目的操作数的自增运算，调用 rtl_add()加 1，然后写回，此指令不会改变 CF 标志，然后更新 ZF、SF 以及 OF 标志。

```
make_EHelper(dec) {
  t0 = 1;
  rtl_sub(&t2,&id_dest->val,&t0);
  operand_write(id_dest,&t2);

  rtl_update_ZFSF(&t2,id_dest->width);

  rtl_xor(&t0,&id_dest->val,&id_src->val);
        rtl_xor(&t1,&id_dest->val,&t2);
  rtl_and(&t0,&t0,&t1);
  rtl_msb(&t0,&t0,id_dest->width);

  print_asm_template1(dec);
}
```

dec: 此指令用于目的操作数的自减运算，调用 rtl_sub()减 1，然后写回，此指令不会改变 CF 标志，然后更新 ZF、SF 以及 OF 标志。

```
make_EHelper(neg) {
  if(!id_dest->val) {
        rtl_set_CF(&tzero);
  }
  else {
        rtl_addi(&t0,&tzero,1);
        rtl_set_CF(&t0);
  }
  rtl_add(&t0,&tzero,&id_dest->val);
  t0 = -t0;
  operand_write(id_dest,&t0);

  rtl_update_ZFSF(&t2,id_dest->width);

  rtl_xor(&t0,&id_dest->val,&id_src->val);
  rtl_xor(&t1,&id_dest->val,&t2);
  rtl_and(&t0,&t0,&t1);
  rtl_msb(&t0,&t0,id_dest->width);
  rtl_set_OF(&t0);

  print_asm_template1(neg);
}
```

neg:此指令是取负指令(也叫取补指令)，查手册知，若目的操作数==0，则 CF 为 0，否则 CF 为 1。然后将操作是取负，更新相应的 ZF、SF、以及 OF 标志。

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
+ CC src/cpu/exec/arith.c

+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo 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: 10:42:15, Apr 15 2018
For help, type "help"
(nemu) q
```

说明实现没有错误

下面进入 nemu/src/exec/logic.c 文件，进行 not、and、or、xor、sal(shl)、shr、sar、test 指令执行函数的实现，下面是实现代码

```
make_EHelper(and) {
  rtl_and(&t2,&id_dest->val,&id_src->val);
  operand_write(id_dest,&t2);

  rtl_update_ZFSF(&t2,id_dest->width);
  rtl_set_CF(&tzero);
  rtl_set_OF(&tzero);
  print_asm_template2(and);
}
```

and：调用 rtl_and()实现两个操作数的按位与运算，然后写回。由手册知，此指令的 CF、OF 标志位都为 0，然后更新 ZF、SF。

```
make_EHelper(or) {
  rtl_or(&t2, &id_dest->val, &id_src->val);

        operand_write(id_dest, &t2);

        rtl_set_OF(&tzero);

        rtl_set_CF(&tzero);

        rtl_update_ZFSF(&t2, id_dest->width);

  print_asm_template2(or);
}
```

or:同 and 操作。

```
make_EHelper(shl) {
  rtl_shl(&t2, &id_dest->val, &id_src->val);

        operand_write(id_dest, &t2);

        rtl_update_ZFSF(&t2, id_dest->width);

  // unnecessary to update CF and OF in NEMU

  print_asm_template2(shl);
}
```

shl:由于不需要设置 **CF、OF** 标志位，所以调用 **rtl_shl()**实现逻辑左移，写回，更新 **ZF、SF** 即可。

```
make_EHelper(not) {
 rtl_not(&id_dest->val);

       operand_write(id_dest, &id_dest->val);

  print_asm_template1(not);
}
```

not:not 指令只有一个操作数，因此调用 **rtl_not()**按位取反，然后写回即可。

```
make_EHelper(sar) {
  if(id_dest->width == 1) {
          id_dest->val = (int8_t)id_dest->val;
  }
  else if(id_dest->width == 2) {
          id_dest->val = (int16_t)id_dest->val;
  }
  rtl_sar(&t2,&id_dest->val,&id_src->val);
  operand_write(id_dest,&t2);
  rtl_update_ZFSF(&t2,id_dest->width);
  // unnecessary to update CF and OF in NEMU

  print_asm_template2(sar);
}
```

sar:查手册知，此指令相当与带符号除，向着负无穷方向，因此，做了符号扩展，然后调用 **rtl_sar(),**然后写回，更新 **ZF、SF** 即可。

```
make_EHelper(shr) {

  rtl_shr(&t2,&id_dest->val.&id_src->val);
  operand_write(id_dest,&t2);
  rtl_update_ZFSF(&t2,id_dest->width);
  // unnecessary to update CF and OF in NEMU

  print_asm_template2(shr);
}
```

shr:此指令与 **sar** 的不同之处在于它是无符号除法，其余操作同 **sar**。
由于讲义中漏写了 **rol(**循环左移**)**，因此在本文件定义，并实现之

```
make_EHelper(rol) {
        rtl_shl(&t0, &id_dest->val, &id_src->val);
        rtl_shri(&t1, &id_dest->val, id_dest->width * 8 - id_src->val);
        rtl_or(&t2, &t1, &t0);
        operand_write(id_dest, &t2);

        print_asm_template2(rol);
}
```

rol:此指令模仿 shl、shr 即可。

```
make_EHelper(test) {
  rtl_and(&t2,&id_dest->val,&id_src->val);
  rtl_update_ZFSF(&t2,id_dest->width);
  rtl_set_CF(&tzero);
  rtl_set_OF(&tzero);
  print_asm_template2(test);
}
```

test:查手册知，此指令的操作为两个操作数想与，结果不写回，只影响标志位。CF、OF 均置为 0，同时更新 ZF、SF。

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
[sudo] zhaoweikang 的密码：
+ CC src/cpu/exec/logic.c

+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo 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: 10:42:15, Apr 15 2018
For help, type "help"
(nemu)
```

说明实现没有错误

此外，由于框架已经帮我们实现了 setcc 没有的执行函数，但是与之对应的 RTL 指令并没有完全实现，需要更新相应的 EFLAGS，下面进入 nemu/src/cpu/exec/cc.c 实现

```
void rtl_setcc(rtlreg_t* dest, uint8_t subcode) {
  bool invert = subcode & 0x1;
  enum {
    CC_O, CC_NO, CC_B,  CC_NB,
    CC_E, CC_NE, CC_BE, CC_NBE,
    CC_S, CC_NS, CC_P,  CC_NP,
    CC_L, CC_NL, CC_LE, CC_NLE
  };

  // TODO: Query EFLAGS to determine whether the condition code is satisfied.
  // dest <- ( cc is satisfied ? 1 : 0)
  switch (subcode & 0xe) {
    case CC_O:
                    *dest = cpu.eflags.OF;
                    break;
    case CC_B:
                    *dest = cpu.eflags.CF;
                    break;
    case CC_E:
                    *dest = cpu.eflags.ZF;
                    break;
    case CC_BE:
                    *dest = ((cpu.eflags.CF) || (cpu.eflags.ZF));
                    break;
    case CC_S:
                     *dest = cpu.eflags.SF;
                     break;
    case CC_L:
                     *dest = (cpu.eflags.SF != cpu.eflags.OF);
                     break;
    case CC_LE:
                     *dest = ((cpu.eflags.ZF) || (cpu.eflags.SF != cpu.eflags.OF));
                     break;
    default: panic("should not reach here");
    case CC_P: panic("n86 does not have PF");
  }

  if (invert) {
    rtl_xori(dest, dest, 0x1);
  }
}
```

setcc: 查 手 册 知 ， seto:OF=1;setb:CF=1;sete:ZF=1;setbe:CF=1 or ZF=1;sets:SF=1;setl:SF!=OF;setle:ZF=1 or SF!=OF。

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
[sudo] zhaoweikang 的密码：
+ CC src/cpu/exec/cc.c

+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo 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: 10:42:15, Apr 15 2018
For help, type "help"
(nemu)
```

说明实现没有错误

进入 nemu/src/cpu/exec/exec.c 文件 ， 下面填写 opcode_table[]表，具体实现如 mov:框架已实现

leave:

```
/* 0xc8 */  EMPTY,EX(leave), EMPTY, EMPTY,
```

cltd:

```
 /* 0x98 */  EMPTY,  EX(cltd), EMPTY, EMPTY,
```

movsx:

```
 /* 0xbc */  EMPTY, EMPTY, IDEXW(mov_E2G, movsx, 1), IDEXW(mov_E2G, movsx, 2)
```

movzx:

```
 /* 0xb4 */  EMPTY, EMPTY,  IDEXW(mov_E2G, movzx, 1), IDEXW(mov_E2G, movzx, 2),
```

add:

```
/* 0x00 */  IDEXW(G2E, add, 1), IDEX(G2E, add), IDEXW(E2G, add, 1), IDEX(E2G, add),
 /* 0x04 */  IDEXW(I2a, add, 1), IDEX(I2a, add), EMPTY, EMPTY,
```

inc:

```
 /* 0x40 */  IDEX(r, inc), IDEX(r, inc), IDEX(r, inc), IDEX(r, inc),
 /* 0x44 */  IDEX(r, inc), IDEX(r, inc), IDEX(r, inc), IDEX(r, inc),
```

dec:

```
 /* 0x48 */  IDEX(r, dec), IDEX(r, dec), IDEX(r, dec), IDEX(r, dec),
 /* 0x4c */  IDEX(r, dec), IDEX(r, dec), IDEX(r, dec), IDEX(r, dec),
```

cmp:

```
 /* 0x38 */  IDEXW(G2E, cmp, 1), IDEX(G2E, cmp), IDEXW(E2G, cmp, 1), IDEX(E2G, cmp),
 /* 0x3c */  IDEXW(I2a, cmp, 1), IDEX(I2a, cmp), EMPTY, EMPTY,

 /* 0x80 */  IDEXW(I2E, gp1, 1), IDEX(I2E, gp1), EMPTY, IDEX(SI2E, gp1),
```

neg: 框架已实现

adc:

```
 /* 0x10 */  IDEXW(G2E, adc, 1), IDEX(G2E, adc), IDEXW(E2G, adc, 1), IDEX(E2G, adc),
 /* 0x14 */  IDEXW(I2a, adc, 1), IDEX(I2a, adc), EMPTY, EMPTY,
```

sbb:

```
/* 0x18 */  IDEXW(G2E, sbb, 1), IDEX(G2E, sbb), IDEXW(E2G, sbb, 1), IDEX(E2G, sbb),
/* 0x1c */  IDEXW(I2a, sbb, 1), IDEX(I2a, sbb), EMPTY, EMPTY,
```

mul:框架已实现

imul:

```
/* 0xf4 */  EMPTY, EMPTY, IDEXW(E, gp3, 1), IDEX(E, gp3),

/* 0xac */  EMPTY, EMPTY, EMPTY, IDEX(E2G, imul2),

/* 0x68 */  EMPTY, EMPTY, EMPTY, IDEX(I_E2G, imul3),
```

div:框架已实现

idiv: 框架已实现

not: 框架已实现

and:

```
/* 0x20 */  IDEXW(G2E, and, 1), IDEX(G2E, and), IDEXW(E2G, and, 1), IDEX(E2G, and),
/* 0x24 */  IDEXW(I2a, and, 1), IDEX(I2a, and), EMPTY, EMPTY,

 /* 0x80 */  IDEXW(I2E, gp1, 1), IDEX(I2E, gp1), EMPTY, IDEX(SI2E, gp1),
```

or:

```
 /* 0x08 */  IDEXW(G2E, or, 1), IDEX(G2E, or), IDEXW(E2G, or, 1), IDEX(E2G, or),
 /* 0x0c */  IDEXW(I2a, or, 1), IDEX(I2a, or), EMPTY, EX(2byte_esc),

 /* 0x80 */  IDEXW(I2E, gp1, 1), IDEX(I2E, gp1), EMPTY, IDEX(SI2E, gp1),
```

shl：框架已实现

shr: 框架已实现

sar: 框架已实现

setcc:

```
  /* 0x90 */  IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E,
setcc, 1),

  /* 0x94 */  IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E,
setcc, 1),

  /* 0x98 */  IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E,
setcc, 1),

  /* 0x9c */  IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E, setcc, 1), IDEXW(E,
setcc, 1),
```

test:

```
 /* 0x84 */  IDEXW(G2E, test, 1), IDEX(G2E, test), EMPTY, EMPTY,

  /* 0xa8 */  IDEXW(I2a, test, 1), IDEX(I2a, test), EMPTY, EMPTY,

 /* 0xf4 */  EMPTY, EMPTY, IDEXW(E, gp3, 1), IDEX(E, gp3),
```

jmp:

```
 /* 0xe8 */  IDEX(J, call), IDEXW(J, jmp, 4), EMPTY, IDEXW(J, jmp, 1),
```

jcc:

```
 /* 0x80 */  IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc),

 /* 0x84 */  IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc),

 /* 0x88 */  IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc),

 /* 0x8c */  IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc), IDEX(J, jcc),
```

lea:

```
 /* 0x8c */  EMPTY,  IDEX(lea_M2G, lea), EMPTY, EMPTY,
```

nop:

```
  /* 0x90 */  EX(nop), EMPTY, EMPTY, EMPTY,
```

然后 **make** 以及 **make run** 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
+ CC src/cpu/exec/exec.c

+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo 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: 10:42:15, Apr 15 2018
For help, type "help"
(nemu)
```

说明实现没有错误

运行时环境与 **AM**

运行更多的程序

首先进入 nexus-am/Makefile.check 文件，做一些修改，如图

```
ifneq ($(MAKECMDGOALS),clean) # ignore check for make clean

ifeq ($(AM_HOME),) # AM_HOME must exist
$(error Environment variable AM_HOME must be defined.)
endif
#ARCH ?= native
ARCH ?= x86-nemu
ARCHS = $(shell ls $(AM_HOME)/am/arch/)

ifeq ($(filter $(ARCHS), $(ARCH)), ) # ARCH must be valid
$(error Invalid ARCH. Supported: $(ARCHS))
endif

endif
```

同样，进入 nexus-am/am/arch/x86-nemu/img/run 文件，做一些修改，以便可以使用 GDB 调试

```
#!/bin/bash

#make -C $NEMU_HOME run ARGS="-l `dirname $1`/nemu-log.txt $1.bin"
make -C $NEMU_HOME gdb ARGS="-l `dirname $1`/nemu-log.txt $1.bin"
```

先测试 add-longlong.c,如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=add-
longlong run
Building add-longlong [x86-nemu]
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/add-longlong-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 add.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=add
run
Building add [x86-nemu]
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/add-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 bit.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=bit
run
Building bit [x86-nemu]
+ CC tests/bit.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/bit-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 bubble-sort.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=bubb
le-sort run
Building bubble-sort [x86-nemu]
+ CC tests/bubble-sort.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/bubble-sort-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 fact.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=fact
 run
Building fact [x86-nemu]
+ CC tests/fact.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/fact-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 fib.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=fib
run
Building fib [x86-nemu]
+ CC tests/fib.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/fib-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 goldbach.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=gold
bach run
Building goldbach [x86-nemu]
+ CC tests/goldbach.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/goldbach-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 if-else.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=if-e
lse run
Building if-else [x86-nemu]
+ CC tests/if-else.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/if-else-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 leap-year.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=leap
-year run
Building leap-year [x86-nemu]
+ CC tests/leap-year.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/leap-year-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 load-store.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=load
-store run
Building load-store [x86-nemu]
+ CC tests/load-store.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/load-store-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 matrix-mul.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=matr
ix-mul run
Building matrix-mul [x86-nemu]
+ CC tests/matrix-mul.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/matrix-mul-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 max.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=max
run
Building max [x86-nemu]
+ CC tests/max.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 min3.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=min3
 run
Building min3 [x86-nemu]
+ CC tests/min3.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 mov-c.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=mov-
c run
Building mov-c [x86-nemu]
+ CC tests/mov-c.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 movsx.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=movs
x run
Building movsx [x86-nemu]
+ CC tests/movsx.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 mul-longlong.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=mul-
longlong run
Building mul-longlong [x86-nemu]
+ CC tests/mul-longlong.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 pascal.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=pasc
al run
Building pascal [x86-nemu]
+ CC tests/pascal.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 prime.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=prim
e run
Building prime [x86-nemu]
+ CC tests/prime.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 quick-sort.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=quic
k-sort run
Building quick-sort [x86-nemu]
+ CC tests/quick-sort.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 recursion.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=recu
rsion run
Building recursion [x86-nemu]
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 select-sort.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=sele
ct-sort run
Building select-sort [x86-nemu]
+ CC tests/select-sort.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 **shift.c**，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=shif
t run
Building shift [x86-nemu]
+ CC tests/shift.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 **shuixianhua.c**，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=shui
xianhua run
Building shuixianhua [x86-nemu]
+ CC tests/shuixianhua.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 **sub-longlong.c**，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=sub-
longlong run
Building sub-longlong [x86-nemu]
+ CC tests/sub-longlong.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 sum.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=sum
run
Building sum [x86-nemu]
+ CC tests/sum.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 switch.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=swit
ch run
Building switch [x86-nemu]
+ CC tests/switch.c
Building am [x86-nemu]

[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 to-lower-case.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=to-l
ower-case run
Building to-lower-case [x86-nemu]
+ CC tests/to-lower-case.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 unalign.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=unal
ign run
Building unalign [x86-nemu]
+ CC tests/unalign.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 wanshu.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=wans
hu run
Building wanshu [x86-nemu]
+ CC tests/wanshu.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 string.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=stri
ng run
Building string [x86-nemu]
+ CC tests/string.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

测试 hello-str.c，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/cputest# make ALL=hell
o-str run
Building hello-str [x86-nemu]
+ CC tests/hello-str.c
Building am [x86-nemu]
```

```
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/cputest/build/max-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:30:21, Apr 18 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100027

(nemu)
```

说明实现成功

Differential Testing

Differential Testing 的引入：在 nemu/include/common.h 文件中定义宏 DIFF_TEST，如图

```
#ifndef __COMMON_H__
#define __COMMON_H__

#define DEBUG
#define DIFF_TEST
```

重新编译 NEMU 后运行,NEMU 多输出了 Connect to QEMU successfully 的信息，如图

```
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[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: 11:00:53, Apr 30 2018
For help, type "help"
(nemu)
```

下面进入 nemu/src/monitor/diff-test/diff-test.c 文件，完成 difftest_step()函数，以便实现 Differential Testing,如图

```
  gdb_si();
  gdb_getregs(&r);
  regcpy_from_nemu(mine);
  // TODO: Check the registers state with QEMU.
  // Set `diff` as `true` if they are not the same.
  if(r.eax != mine.eax || r.ecx != mine.ecx || r.edx != mine.edx || r.ebx != mine.ebx || r.esp !=
mine.esp || r.ebp != mine.ebp || r.esi != mine.esi || r.edi != mine.edi || r.eip != mine.eip) {
     diff = true;
     printf("qemus eax:0x%08x,mine eax:0x%08x,#eip:0x%08x\n",r.eax,mine.eax,mine.eip);
     printf("qemus ecx:0x%08x,mine ecx:0x%08x,#eip:0x%08x\n",r.ecx,mine.ecx,mine.eip);
     printf("qemus edx:0x%08x,mine edx:0x%08x,#eip:0x%08x\n",r.edx,mine.edx,mine.eip);
     printf("qemus ebx:0x%08x,mine ebx:0x%08x,#eip:0x%08x\n",r.ebx,mine.ebx,mine.eip);
     printf("qemus esp:0x%08x,mine esp:0x%08x,#eip:0x%08x\n",r.esp,mine.esp,mine.eip);
     printf("qemus ebp:0x%08x,mine ebp:0x%08x,#eip:0x%08x\n",r.ebp,mine.ebp,mine.eip);
     printf("qemus esi:0x%08x,mine esi:0x%08x,#eip:0x%08x\n",r.esi,mine.esi,mine.eip);
     printf("qemus edi:0x%08x,mine edi:0x%08x,#eip:0x%08x\n",r.edi,mine.edi,mine.eip);
     printf("qemus eip:0x%08x,mine eip:0x%08x,#eip:0x%08x\n",r.eip,mine.eip,mine.eip);
  }
  if (diff) {
    nemu_state = NEMU_END;
  }
}
```

然后 **make** 以及 **make run** 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make run
./build/nemu -l ./build/nemu-log.txt
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[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: 11:00:53, Apr 30 2018
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100026

(nemu)
```

没有错误，说明正确实现了 **Differential Testing**。

一键回归测试

运行 **bash runall.sh** 命令，结果如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nemu# bash runall.sh
NEMU compile OK
compiling testcases...
testcases compile OK
[  add-longlong] PASS!
[           add] PASS!
[           bit] PASS!
[   bubble-sort] PASS!
[         dummy] PASS!
[          fact] PASS!
[           fib] PASS!
[      goldbach] PASS!
[     hello-str] PASS!
[       if-else] PASS!
[     leap-year] PASS!
[    load-store] PASS!
[    matrix-mul] PASS!
[           max] PASS!
[          min3] PASS!
[         mov-c] PASS!
[         movsx] PASS!
[  mul-longlong] PASS!
[        pascal] PASS!
[         prime] PASS!
[    quick-sort] PASS!
[     recursion] PASS!
[   select-sort] PASS!
[         shift] PASS!
[   shuixianhua] PASS!
[        string] PASS!
[  sub-longlong] PASS!
[           sum] PASS!
[        switch] PASS!
[ to-lower-case] PASS!
[       unalign] PASS!
[        wanshu] PASS!
```

全部 pass,说明所有指令均实现正确

思考题：NEMU 的本质

把思绪回归到 PA 中,通用程序的性质告诉我们,NEMU 的潜力是无穷的。为了创造出一个缤纷多彩的世界,你觉得 NEMU 还缺少些什么呢?

答：我觉得 NEMU 还缺少对缓存（Cache）相关操作的功能。

git log 记录

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git status
位于分支 master
要提交的变更：
  （使用 "git reset HEAD <文件>..." 以取消暂存）

        修改：     include/common.h
        修改：     include/cpu/rtl.h
        修改：     src/cpu/decode/decode.c
        修改：     src/cpu/exec/all-instr.h
        修改：     src/cpu/exec/arith.c
        修改：     src/cpu/exec/cc.c
        修改：     src/cpu/exec/control.c
        修改：     src/cpu/exec/data-mov.c
        修改：     src/cpu/exec/exec.c
        修改：     src/cpu/exec/logic.c
        修改：     src/monitor/diff-test/diff-test.c

zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git add .
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git commit --allow-empty
[master 9c1f823] fix bug for pa2.2
 11 files changed, 316 insertions(+), 88 deletions(-)
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git log
commit 9c1f823f8235a814fd302195da9824170fac759f

Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Mon Apr 30 17:04:44 2018 +0800

    fix bug for pa2.2

commit ad2bf1ce5a92514156820ffda60e4a4023295814
Author: tracer-ics2017 <tracer@njuics.org>
Date:   Mon Apr 30 16:58:18 2018 +0800

    >  run
    161630220
    root
    Linux zhaoweikang 4.9.0-6-686-pae #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02)
 i686 GNU/Linux
     16:58:18 up  3:25,  1 user,  load average: 0.00, 0.00, 0.00
    66caacd06ee082c5d981cc48824847c0a32f1b5c

commit 5e22e09951a1d3d445cc5aafdde523a9b62814fb
Author: tracer-ics2017 <tracer@njuics.org>
Date:   Mon Apr 30 16:41:07 2018 +0800

    >  run
```

输入输出

思考题：理解 volatile 关键字

（这段程序我是在 ubuntu 中写的）去掉 volatile 关键字前的反汇编代码，如图

```
root@zhaoweikang-virtual-machine:/home/zhaoweikang/桌面# objdump -d exefile

exefile:     文件格式 elf64-x86-64


Disassembly of section .init:

00000000004003c8 <_init>:
  4003c8:       48 83 ec 08             sub    $0x8,%rsp
  4003cc:       48 8b 05 25 0c 20 00    mov    0x200c25(%rip),%rax        # 600ff8 <_DYNAMIC+0x1d0>
  4003d3:       48 85 c0                test   %rax,%rax
  4003d6:       74 05                   je     4003dd <_init+0x15>
  4003d8:       e8 43 00 00 00          callq  400420 <__libc_start_main@plt+0x10>
  4003dd:       48 83 c4 08             add    $0x8,%rsp
  4003e1:       c3                      retq

Disassembly of section .plt:

00000000004003f0 <puts@plt-0x10>:
  4003f0:       ff 35 12 0c 20 00       pushq  0x200c12(%rip)        # 601008 <_GLOBAL_OFFSET_TABLE_+0x
8>
  4003f6:       ff 25 14 0c 20 00       jmpq   *0x200c14(%rip)        # 601010 <_GLOBAL_OFFSET_TABLE_+0
x10>
  4003fc:       0f 1f 40 00             nopl   0x0(%rax)

0000000000400400 <puts@plt>:
  400400:       ff 25 12 0c 20 00       jmpq   *0x200c12(%rip)        # 601018 <_GLOBAL_OFFSET_TABLE_+0
x18>
  400406:       68 00 00 00 00          pushq  $0x0
  40040b:       e9 e0 ff ff ff          jmpq   4003f0 <_init+0x28>

0000000000400410 <__libc_start_main@plt>:
  400410:       ff 25 0a 0c 20 00       jmpq   *0x200c0a(%rip)        # 601020 <_GLOBAL_OFFSET_TABLE_+0
x20>
  400416:       68 01 00 00 00          pushq  $0x1
  40041b:       e9 d0 ff ff ff          jmpq   4003f0 <_init+0x28>

Disassembly of section .plt.got:

0000000000400420 <.plt.got>:
```

```
0000000000400420 <.plt.got>:
  400420:       ff 25 d2 0b 20 00       jmpq   *0x200bd2(%rip)        # 600ff8 <_DYNAMIC+0x1d0>
  400426:       66 90                   xchg   %ax,%ax

Disassembly of section .text:

0000000000400430 <_start>:
  400430:       31 ed                   xor    %ebp,%ebp
  400432:       49 89 d1                mov    %rdx,%r9
  400435:       5e                      pop    %rsi
  400436:       48 89 e2                mov    %rsp,%rdx
  400439:       48 83 e4 f0             and    $0xfffffffffffffff0,%rsp
  40043d:       50                      push   %rax
  40043e:       54                      push   %rsp
  40043f:       49 c7 c0 b0 05 40 00    mov    $0x4005b0,%r8
  400446:       48 c7 c1 40 05 40 00    mov    $0x400540,%rcx
  40044d:       48 c7 c7 26 05 40 00    mov    $0x400526,%rdi
  400454:       e8 b7 ff ff ff          callq  400410 <__libc_start_main@plt>
  400459:       f4                      hlt
  40045a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

0000000000400460 <deregister_tm_clones>:
  400460:       b8 3f 10 60 00          mov    $0x60103f,%eax
  400465:       55                      push   %rbp
  400466:       48 2d 38 10 60 00       sub    $0x601038,%rax
  40046c:       48 83 f8 0e             cmp    $0xe,%rax
  400470:       48 89 e5                mov    %rsp,%rbp
  400473:       76 1b                   jbe    400490 <deregister_tm_clones+0x30>
  400475:       b8 00 00 00 00          mov    $0x0,%eax
  40047a:       48 85 c0                test   %rax,%rax
  40047d:       74 11                   je     400490 <deregister_tm_clones+0x30>
  40047f:       5d                      pop    %rbp
  400480:       bf 38 10 60 00          mov    $0x601038,%edi
  400485:       ff e0                   jmpq   *%rax
  400487:       66 0f 1f 84 00 00 00    nopw   0x0(%rax,%rax,1)
  40048e:       00 00
  400490:       5d                      pop    %rbp
  400491:       c3                      retq
  400492:       0f 1f 40 00             nopl   0x0(%rax)
  400496:       66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
  40049d:       00 00 00
```

```
  4004a6:       48 81 ee 38 10 60 00    sub    $0x601038,%rsi
  4004ad:       48 c1 fe 03             sar    $0x3,%rsi
  4004b1:       48 89 e5                mov    %rsp,%rbp
  4004b4:       48 89 f0                mov    %rsi,%rax
  4004b7:       48 c1 e8 3f             shr    $0x3f,%rax
  4004bb:       48 01 c6                add    %rax,%rsi
  4004be:       48 d1 fe                sar    %rsi
  4004c1:       74 15                   je     4004d8 <register_tm_clones+0x38>
  4004c3:       b8 00 00 00 00          mov    $0x0,%eax
  4004c8:       48 85 c0                test   %rax,%rax
  4004cb:       74 0b                   je     4004d8 <register_tm_clones+0x38>
  4004cd:       5d                      pop    %rbp
  4004ce:       bf 38 10 60 00          mov    $0x601038,%edi
  4004d3:       ff e0                   jmpq   *%rax
  4004d5:       0f 1f 00                nopl   (%rax)
  4004d8:       5d                      pop    %rbp
  4004d9:       c3                      retq
  4004da:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

LibreOffice Calc
_do_global_dtors_aux>:
  4004e0:       80 3d 51 0b 20 00 00    cmpb   $0x0,0x200b51(%rip)        # 601038 <__TMC_END__>
  4004e7:       75 11                   jne    4004fa <__do_global_dtors_aux+0x1a>
  4004e9:       55                      push   %rbp
  4004ea:       48 89 e5                mov    %rsp,%rbp
  4004ed:       e8 6e ff ff ff          callq  400460 <deregister_tm_clones>
  4004f2:       5d                      pop    %rbp
  4004f3:       c6 05 3e 0b 20 00 01    movb   $0x1,0x200b3e(%rip)        # 601038 <__TMC_END__>
  4004fa:       f3 c3                   repz retq
  4004fc:       0f 1f 40 00             nopl   0x0(%rax)

0000000000400500 <frame_dummy>:
  400500:       bf 20 0e 60 00          mov    $0x600e20,%edi
  400505:       48 83 3f 00             cmpq   $0x0,(%rdi)
  400509:       75 05                   jne    400510 <frame_dummy+0x10>
  40050b:       eb 93                   jmp    4004a0 <register_tm_clones>
  40050d:       0f 1f 00                nopl   (%rax)
  400510:       b8 00 00 00 00          mov    $0x0,%eax
  400515:       48 85 c0                test   %rax,%rax
  400518:       74 f1                   je     40050b <frame_dummy+0xb>
  40051a:       55                      push   %rbp
  40051b:       48 89 e5                mov    %rsp,%rbp
```

```
  40055a:       53                      push   %rbx
  40055b:       49 89 f6                mov    %rsi,%r14
  40055e:       49 89 d5                mov    %rdx,%r13
  400561:       4c 29 e5                sub    %r12,%rbp
  400564:       48 83 ec 08             sub    $0x8,%rsp
  400568:       48 c1 fd 03             sar    $0x3,%rbp
  40056c:       e8 57 fe ff ff          callq  4003c8 <_init>
  400571:       48 85 ed                test   %rbp,%rbp
  400574:       74 20                   je     400596 <__libc_csu_init+0x56>
  400576:       31 db                   xor    %ebx,%ebx
  400578:       0f 1f 84 00 00 00 00    nopl   0x0(%rax,%rax,1)
  40057f:       00
  400580:       4c 89 ea                mov    %r13,%rdx
  400583:       4c 89 f6                mov    %r14,%rsi
LibreOffice Writer
                ff                    mov    %r15d,%edi
                14 dc                 callq  *(%r12,%rbx,8)
  40058d:       48 83 c3 01             add    $0x1,%rbx
  400591:       48 39 eb                cmp    %rbp,%rbx
  400594:       75 ea                   jne    400580 <__libc_csu_init+0x40>
  400596:       48 83 c4 08             add    $0x8,%rsp
  40059a:       5b                      pop    %rbx
  40059b:       5d                      pop    %rbp
  40059c:       41 5c                   pop    %r12
  40059e:       41 5d                   pop    %r13
  4005a0:       41 5e                   pop    %r14
  4005a2:       41 5f                   pop    %r15
  4005a4:       c3                      retq
  4005a5:       90                      nop
  4005a6:       66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
  4005ad:       00 00 00

00000000004005b0 <__libc_csu_fini>:
  4005b0:       f3 c3                   repz retq

Disassembly of section .fini:

00000000004005b4 <_fini>:
  4005b4:       48 83 ec 08             sub    $0x8,%rsp
  4005b8:       48 83 c4 08             add    $0x8,%rsp
  4005bc:       c3                      retq
```

去掉 **volatile** 关键字反汇编结果省略。

你或许会感到疑惑,代码优化不是一件好事情吗?为什么会有 volatil 这种奇葩的存在？思考一下,如果代码中的地址 0x8049000 最终被映射到一个设备寄存器,去掉 volatile可能会带来什么问题?

答：一个定义为 volatile 的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份，例如，状态寄存器。

加入 IOE

下面进入 nemu/include/common.c 文件，定义宏 HAS_IOE（其实只需要去掉注释即可），如图

```
/* You will define this macro in PA2 */
#define HAS_IOE
```

重新编译后,运行 NEMU，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
+ CC src/device/serial.c
+ CC src/device/vga.c
+ CC src/device/device.c
+ CC src/device/keyboard.c
+ CC src/device/timer.c
+ CC src/device/io/mmio.c
+ CC src/device/io/port-io.c
+ CC src/memory/memory.c
+ CC src/monitor/cpu-exec.c
+ CC src/monitor/monitor.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/cpu/decode/modrm.c
+ CC src/cpu/decode/decode.c
+ CC src/cpu/reg.c
```

```
+ CC src/cpu/exec/cc.c
+ CC src/cpu/exec/arith.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/exec/exec.c
+ CC src/cpu/exec/special.c
+ CC src/cpu/intr.c
fatal: ..: '..' 在仓库之外
Makefile:41: recipe for target 'build/nemu' failed
make: [build/nemu] Error 128 (ignored)
+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make run
```

```
./build/nemu -l ./build/nemu-log.txt
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[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: 21:34:55, Apr 30 2018
For help, type "help"
(nemu)
```

会弹出一个新窗口,用于显示 VGA 的输出，如图

串口

下面实现 in、out 指令，先进入 nemu/src/cpu/exec/all-instr.h 文件对 in、out 的执行函数进行声明，如图

```
make_EHelper(in);

make_EHelper(out);
```

进入 nemu/src/cpu/exec/system.c 文件，编写 in、out 执行函数，如图

```
make_EHelper(in) {
  reg_l(R_EAX) = pio_read(reg_w(R_EDX), id_dest->width);

  print_asm_template2(in);

#ifdef DIFF_TEST
  diff_test_skip_qemu();
#endif
}
```

in:查手册知，此指令从源操作数读取端口数据，写入目的操作数所在寄存器。

```
make_EHelper(out) {
   pio_write(reg_w(R_EDX), id_dest->width, reg_l(R_EAX));

  print_asm_template2(out);

#ifdef DIFF_TEST
  diff_test_skip_qemu();
#endif
}
```

out:与 in 指令正好相反。

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
[sudo] zhaoweikang 的密码：
+ CC src/cpu/exec/system.c
+ CC src/cpu/exec/exec.c
```

```
./build/nemu -l ./build/nemu-log.txt
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[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: 21:34:55, Apr 30 2018
For help, type "help"
(nemu)
```

说明实现没有错误

进入 nemu/src/cpu/exec/exec.c 文件，填写译码表，如图

```
/* 0xec */    EXW(in, 1), EX(in), EXW(out, 1), EX(out),
```

然后 make 以及 make run 一下，如图

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
+ CC src/cpu/exec/exec.c
+ LD build/nemu
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make run
```

```
./build/nemu -l ./build/nemu-log.txt
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[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: 21:34:55, Apr 30 2018
For help, type "help"
(nemu)
```

说明实现没有错误

在 nexus-am/am/arch/x86-nemu/src/trm.c中定义宏 HAS_SERIAL（去掉注释即可），如图

```
// Define this macro after serial has been implemented
#define HAS_SERIAL
```

在 nexus-am/apps/hello 目录下键入 make run，如图

```
./build/nemu -l /home/zhaoweikang/ics2017/nexus-am/apps/hello/build/nemu-log.txt
 /home/zhaoweikang/ics2017/nexus-am/apps/hello/build/hello-x86-nemu.bin
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/apps/hello/build/hello-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 21:34:55, Apr 30 2018
For help, type "help"
(nemu) c
 Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
nemu: HIT GOOD TRAP at eip = 0x0010006e

(nemu)
```

时钟

实现 IOE

进入 nexus-am/am/arch/x86-nemu/src/ioe.c 文件，实现_uptime()函数，如图

```
unsigned long _uptime() {
  return inl(0x48) - boot_time;
}
```

讲义上说的很清楚，初始化时将会注册 0x48 处的端口作为 RTC 寄存器,CPU 可以通过 I/O 指令访问这一寄存器,获得当前时间(单位是 ms)。

在 NEMU 中运行 timetest 程序(在 nexus-am/tests/timetest 目录下)，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/timetest# make run
Building timetest [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
```

```
./build/nemu -l /home/zhaoweikang/ics2017/nexus-am/tests/timetest/build/nemu-log
.txt /home/zhaoweikang/ics2017/nexus-am/tests/timetest/build/timetest-x86-nemu.b
in
[src/monitor/diff-test/diff-test.c,96,init_difftest] Connect to QEMU successfull
y
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/tests/timetest/build/timetest-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 21:34:55, Apr 30 2018
For help, type "help"
(nemu) c
 1 second.
2 seconds.
3 seconds.
4 seconds.
5 seconds.
6 seconds.
7 seconds.
8 seconds.
9 seconds.
^Cqemu-system-i386: terminating on signal 2
```

看看 NEMU 跑多快

我们先进入 nemu/include /common.h 文件，将宏 DEBUG 和 DIFF_TEST 注释掉，如图

```
//#define DEBUG
//#define DIFF_TEST
```

进入 nexus-am/apps/ dhrystone 目录，运行 dhrystone 项目，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/apps/dhrystone# make run
Building dhrystone [x86-nemu]
+ CC dry.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
```

```
make[1]: Entering directory '/home/zhaoweikang/ics2017/nemu'
+ CC src/device/serial.c
+ CC src/device/vga.c
+ CC src/device/device.c
+ CC src/device/keyboard.c
+ CC src/device/timer.c
+ CC src/device/io/mmio.c
+ CC src/device/io/port-io.c
+ CC src/memory/memory.c
+ CC src/monitor/cpu-exec.c
+ CC src/monitor/monitor.c
+ CC src/monitor/debug/expr.c
+ CC src/monitor/debug/watchpoint.c
+ CC src/monitor/debug/ui.c
+ CC src/monitor/diff-test/diff-test.c
+ CC src/monitor/diff-test/gdb-host.c
+ CC src/monitor/diff-test/protocol.c
+ CC src/cpu/decode/modrm.c
+ CC src/cpu/decode/decode.c
+ CC src/cpu/reg.c
+ CC src/cpu/exec/cc.c
+ CC src/cpu/exec/arith.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/exec/exec.c
+ CC src/cpu/exec/special.c
+ CC src/cpu/intr.c
./build/nemu -l /home/zhaoweikang/ics2017/nexus-am/apps/dhrystone/build/nemu-log
.txt /home/zhaoweikang/ics2017/nexus-am/apps/dhrystone/build/dhrystone-x86-nemu.
bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus
-am/apps/dhrystone/build/dhrystone-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:37:21, May  1 2018
For help, type "help"
(nemu) c
 Dhrystone Benchmark, Version C, Version 2.2
Trying 500000 runs through Dhrystone.
Finished in 23247 ms
------------------------------------------------
Dhrystone PASS         44 Marks
                   vs. 100000 Marks (i7-6700 @ 3.40GHz)
nemu: HIT GOOD TRAP at eip = 0x0010006e
```

进入 **nexus-am/apps/coremark** 目录，运行 **coremark** 项目，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/apps/coremark# make run
Building coremark [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
```

```
./build/nemu -l /home/zhaoweikang/ics2017/nexus-am/apps/coremark/build/nemu-log
.txt /home/zhaoweikang/ics2017/nexus-am/apps/coremark/build/coremark-x86-nemu.b
in
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexu
s-am/apps/coremark/build/coremark-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:37:21, May  1 2018
For help, type "help"
(nemu) c
 Running CoreMark for 1000 iterations
2K performance run parameters for coremark.
CoreMark Size    : 666
Total time (ms)  : 26755
Iterations       : 1000
Compiler version : GCC6.3.0 20170516
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xd340
Finised in 26755 ms.
==================================================
CoreMark PASS       167 Marks
               vs. 100000 Marks (i7-6700 @ 3.40GHz)

nemu: HIT GOOD TRAP at eip = 0x0010006e
```

进入 nexus-am/apps/microbench 目录，运行 microbench 项目，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/apps/microbench# make run
Building microbench [x86-nemu]
+ CC src/sieve/sieve.c
+ CC src/md5/md5.c
+ CC src/qsort/qsort.c
+ CC src/queen/queen.c
+ CC src/fib/fib.c
+ CXX src/ssort/ssort.cpp
+ CXX src/dinic/dinic.cpp
+ CXX src/15pz/15pz.cpp
+ CC src/bench.c
+ CC src/bf/bf.c
+ CC src/lzip/lzip.c
+ CC src/lzip/quicklz.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
```

```
./build/nemu -l /home/zhaoweikang/ics2017/nexus-am/apps/microbench/build/nemu-log.txt /home/zhaoweikang/ics2017/nexus-am/apps/microbench/build/microbench-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexus-am/apps/microbench/build/microbench-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:37:21, May  1 2018
For help, type "help"
(nemu) c
 [qsort] Quick sort: * Passed.
  min time: 1811 ms [304]
[queen] Queen placement: * Passed.
  min time: 2962 ms [174]
[bf] Brainf**k interpreter: * Passed.
  min time: 16208 ms [161]
[fib] Fibonacci number: * Passed.
  min time: 32633 ms [87]
[sieve] Eratosthenes sieve: * Passed.
  min time: 27373 ms [154]
[15pz] A* 15-puzzle search: * Passed.
  min time: 5586 ms [103]
[dinic] Dinic's maxflow algorithm: * Passed.
  min time: 5029 ms [269]
[lzip] Lzip compression: * Passed.
  min time: 13720 ms [192]
[ssort] Suffix sort: * Passed.
  min time: 2785 ms [212]
[md5] MD5 digest: * Passed.
  min time: 25817 ms [75]
=================================================
MicroBench PASS        173 Marks
                  vs. 100000 Marks (i7-6700 @ 3.40GHz)
nemu: HIT GOOD TRAP at eip = 0x00100032
```

键盘

思考题：如何检测多个键同时被按下

答：如果一个按键被按下，检查其他按键的状态，即它们对应的状态寄存器是否为 1。

实现 IOE(2)

进入 nexus-am/am/arch/x86-nemu/src/ioe.c 文件，实现_read_key ()函数，如图

```
#define I8042_DATA_PORT 0x60

#define I8042_STATUS_PORT 0x64
```

宏定义 i8042 初始化时 0x60 处的数据寄存器，0x64 处的状态寄存器

```
int _read_key() {
uint8_t impresskey = inb(I8042_STATUS_PORT);

  if (impresskey) {

          return inl(I8042_DATA_PORT);

  } else {
  return _KEY_NONE;
 }

}
```

从状态寄存器读取键盘状态，有键盘按下，则从数据寄存器读取键盘码，否则，返回

_KEY_NONE。

进入 nexus-am/tests/keytest 目录，运行 keytest 程序，如图

```
root@zhaoweikang:/home/zhaoweikang/ics2017/nexus-am/tests/keytest# make run
Building keytest [x86-nemu]
+ CXX main.cpp
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
+ CC arch/x86-nemu/src/ioe.c
+ AR /home/zhaoweikang/ics2017/nexus-am/am/build/am-x86-nemu.a
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
./build/nemu -l /home/zhaoweikang/ics2017/nexus-am/tests/keytest/build/nemu-log
.txt /home/zhaoweikang/ics2017/nexus-am/tests/keytest/build/keytest-x86-nemu.bi
n
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nexu
s-am/tests/keytest/build/keytest-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:37:21, May  1 2018
For help, type "help"
(nemu) c
 Get key: 62 M down
Get key: 62 M up
Get key: 48 H down
Get key: 48 H up
Get key: 34 Y down
Get key: 34 Y up
Get key: 35 U down
Get key: 35 U up
Get key: 32 R down
Get key: 32 R up
Get key: 59 V down
Get key: 59 V up
Get key: 60 B down
Get key: 60 B up
Get key: 61 N down
Get key: 61 N up
```

VGA

思考题：神奇的调色板

在一些 90 年代的游戏中,很多渐出渐入效果都是通过调色板实现的,聪明的你知道其中的玄机吗?

答：调色板只有图片的颜色小于等于 256 色的时候才有,16 位高彩和 24 位 32 位真彩是没有调色板的。调色板的存在的意义只是在当初 486 以前为了节省空间的一种采用索引的压缩算法,现在没有人用这种东西。调色板是为了节约空间所用的，相当于一个索引。只有 16 位以下的才用调色板，真彩色不用调色板。

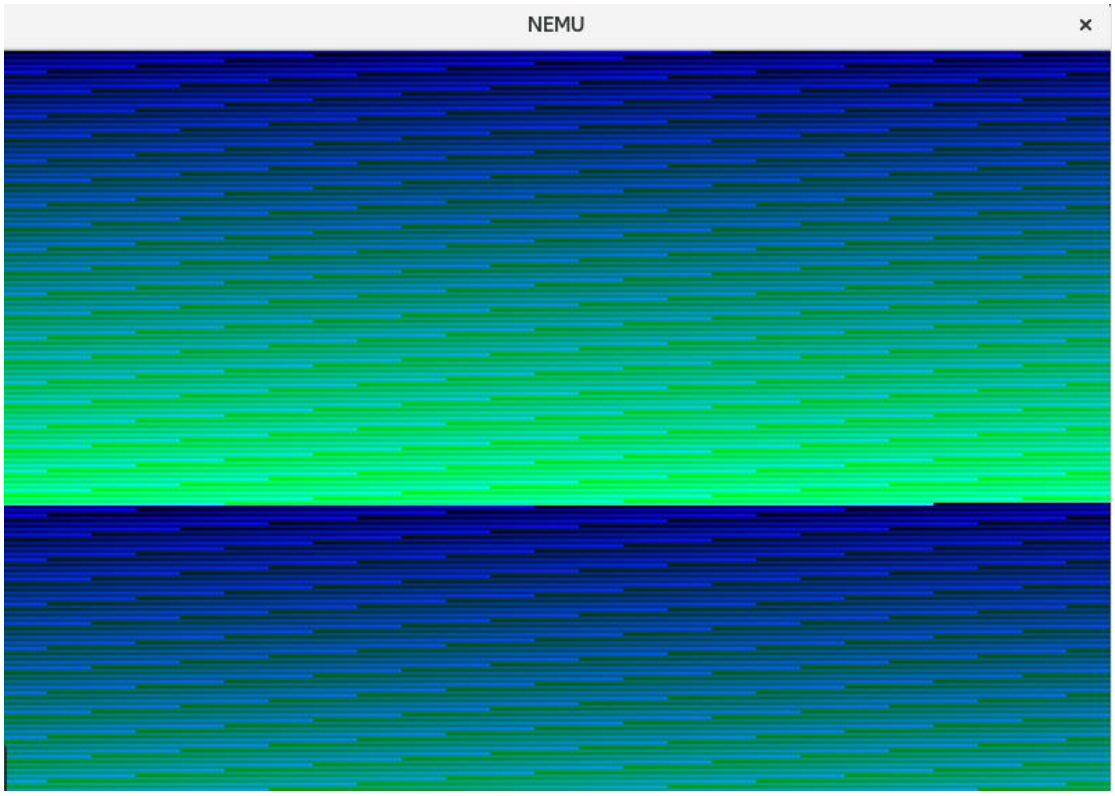
添加内存映射 I/O

进入 nemu/src/memery/memery.c 文件，在 paddr_read()和 paddr_write() 中加入对内存映射 I/O 的判断，如图

```c
/* Memory accessing interfaces */

uint32_t paddr_read(paddr_t addr, int len) {
  int i =is_mmio(addr);
  if (i == -1)
  return pmem_rw(addr, uint32_t) & (~0u >> ((4 - len) << 3));
  return mmio_read(addr,len,i)& (~0u >> ((4 - len) << 3));
}

void paddr_write(paddr_t addr, int len, uint32_t data) {
  int i = is_mmio(addr);
  if(i == -1)
  memcpy(guest_to_host(addr), &data, len);
  else
  mmio_write(addr,len,data,i);
}
```

通过 is_mmio()函数判断 一个物理地址是否被映射到 I/O 空间,如果是,is_mmio()会返回映射号,否则返回-1。

在 NEMU 中运行 videotest 程序(在 nexus-am/tests/videotest 目录下)，新窗口中输出了一些颜色信息，如图

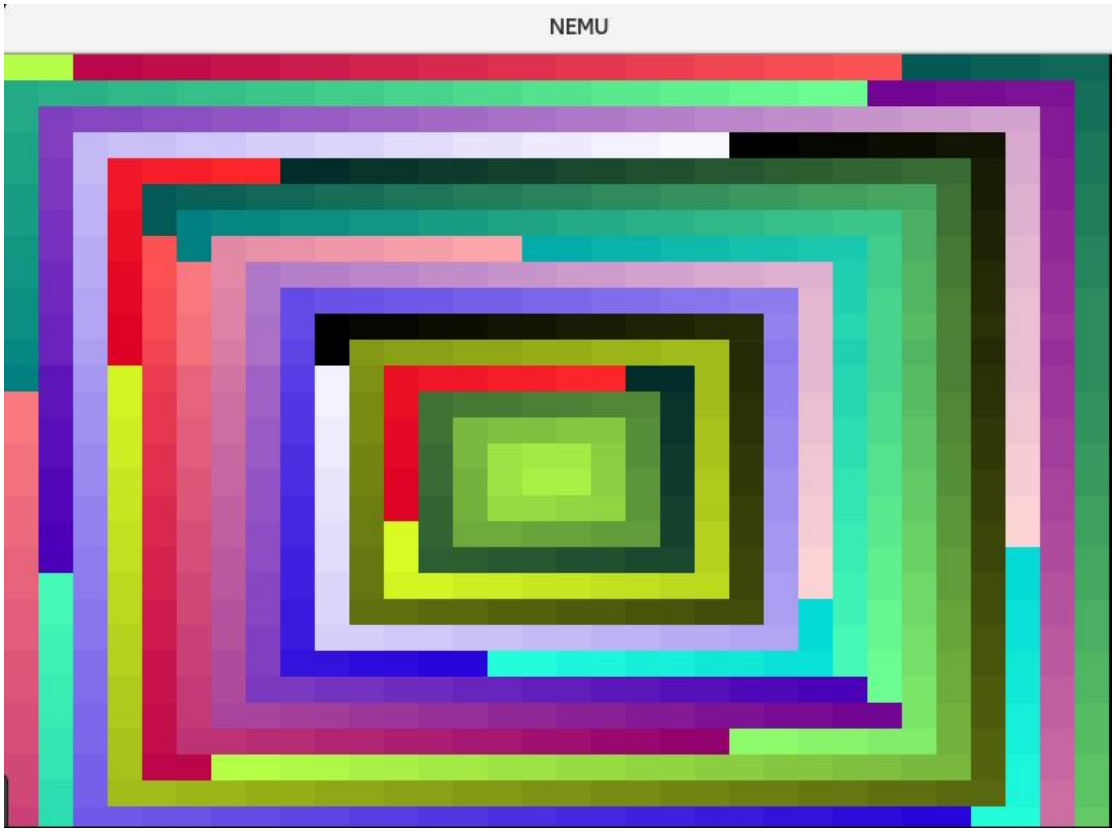


进入 nexus-am/am/arch/x86-nemu/src/ioe.c 文件，实现_draw_rect()函数，如图

，　　如　　图

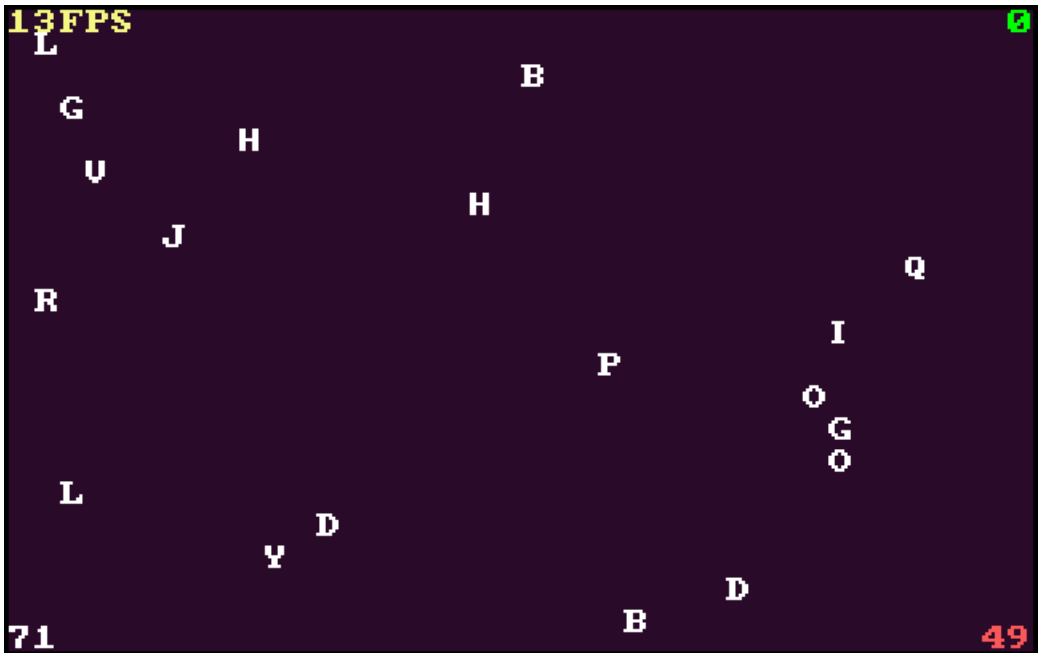
```
void _draw_rect(const uint32_t *pixels, int x, int y, int w, int h) {
   int cp_bytes = sizeof(uint32_t) * (w < (_screen.width - x) ? w : (_screen.width - x));

  for (int j = 0; j < h && y + j < _screen.height; j ++) {

    memcpy(&fb[(y + j) * _screen.width + x], pixels, cp_bytes);

    pixels += w;

  }
}
```

在 NEMU 中重新运行 videotest 程序(在 nexus-am/tests/videotest 目录下)，如图



运行打字小游戏

在 nexus-am/apps/typing 目录下，键入 make ARCH=x86-nemu run，如图



git log 记录

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git status
[sudo] zhaoweikang 的密码：
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git checkout -- <文件>..." 丢弃工作区的改动）

        修改：      include/common.h
        修改：      src/cpu/exec/all-instr.h
        修改：      src/cpu/exec/data-mov.c
        修改：      src/cpu/exec/exec.c
        修改：      src/cpu/exec/system.c
        修改：      src/memory/memory.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git add .a
fatal: 路径规格 '.a' 未匹配任何文件
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git add .
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git commit --allow-empty
[master 5c08a64] fix bug for pa2.3
 6 files changed, 28 insertions(+), 10 deletions(-)
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo git log
commit 5c08a6477bcd2aa6e5a6965cda6931493c81e7cc
```

```
commit 5c08a6477bcd2aa6e5a6965cda6931493c81e7cc
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Wed May 2 20:45:40 2018 +0800

    fix bug for pa2.3

commit 4cbb05f6da0cb4750bb0803826af536bcd9cab0d
Author: tracer-ics2017 <tracer@njuics.org>
Date:   Tue May 1 17:09:27 2018 +0800

    >  run
    161630220
    root
    Linux zhaoweikang 4.9.0-6-686-pae #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02)
 i686 GNU/Linux
     17:09:27 up  4:05,  1 user,  load average: 0.35, 0.28, 0.62
    29d0f9e495435985c97b5666a68b048ea393a3a

commit 0cff4508bcbff8cb576fe7cea4eb4fa92bc35187
Author: tracer-ics2017 <tracer@njuics.org>
Date:   Tue May 1 17:01:17 2018 +0800

    >  run
zhaoweikang@zhaoweikang:~/ics2017/nexus-am$ sudo git status
位于分支 pa2
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git checkout -- <文件>..." 丢弃工作区的改动）

        修改：      ../nemu/include/common.h
        修改：      ../nemu/include/cpu/reg.h
        修改：      ../nemu/include/cpu/rtl.h
        修改：      ../nemu/src/cpu/decode/decode.c
        修改：      ../nemu/src/cpu/exec/all-instr.h
        修改：      ../nemu/src/cpu/exec/arith.c
        修改：      ../nemu/src/cpu/exec/cc.c
        修改：      ../nemu/src/cpu/exec/control.c
        修改：      ../nemu/src/cpu/exec/data-mov.c
        修改：      ../nemu/src/cpu/exec/exec.c
        修改：      ../nemu/src/cpu/exec/logic.c
        修改：      ../nemu/src/cpu/exec/system.c
        修改：      ../nemu/src/memory/memory.c
        修改：      ../nemu/src/monitor/diff-test/diff-test.c
        修改：      ../nemu/src/monitor/monitor.c
        修改：      Makefile.check
        修改：      am/arch/x86-nemu/img/run
```

```
        修改：      am/arch/x86-nemu/src/ioe.c
        修改：      am/arch/x86-nemu/src/trm.c

未跟踪的文件:
  （使用 "git add <文件>..." 以包含要提交的内容）

        tests/cputest/dummy-x86-nemu.txt

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
zhaoweikang@zhaoweikang:~/ics2017/nexus-am$ sudo git add .
zhaoweikang@zhaoweikang:~/ics2017/nexus-am$ sudo git commit --allow-empty
[pa2 ea21a2e] fix bug for pa2.3
 5 files changed, 26 insertions(+), 7 deletions(-)
 create mode 100644 nexus-am/tests/cputest/dummy-x86-nemu.txt
zhaoweikang@zhaoweikang:~/ics2017/nexus-am$ sudo git log
commit ea21a2efaa92091bb5d631d93587313c95e8c450
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Wed May 2 20:50:40 2018 +0800

    fix bug for pa2.3
```

```
commit cc11412df2700fdc325252df017d0fbed7965cbb
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Fri Apr 6 16:48:19 2018 +0800

    before starting pa2

commit f3aebdcc5fa2113d66c1322d3269ddad1e5dd2f2
Author: tracer-ics2017 <tracer@njuics.org>
Date:   Tue Mar 27 17:03:34 2018 +0800

    >  run
    161630220
    root
    Linux zhaoweikang 4.9.0-6-686-pae #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02)
 i686 GNU/Linux
     17:03:34 up  2:31,  1 user,  load average: 0.03, 0.05, 0.00
    fc6d17507c3c02ad4f6202c2b8ebc4cb9edab98
```

必答题：

编译与链接　在 nemu/include/cpu/rtl.h 中,你会看到由 static inline 开头定义的各种 RTL 指令函数。选择其中一个函数,分别尝试去掉 static,去掉 inline 或去掉两者,然后重新进行编译,你会看到发生错误。请分别解释为什么会发生这些错误?你有办法证 明你的想法吗?

答：去掉 static，然后编译，我的并没有发生错误，去掉 inline，提示“-Werror=unused-function”，我认为报错是因为在此处只声明了此函数，并没有定义，所以报错；两者都去掉，提示“xxx 多重定义”，说明此处定义的函数与后面所使用的函数不一致，有二义性，所以报错。

编译与链接

1.在 nemu/include/common.h 中添加一行volatile static int dummy;然后重新编译 NEMU.请问重新编译后的 NEMU 含有多少个 dummy 变量的实体?你是如何得到这个结果的?

答：含有 29 个实体，因为编译了 29 个相关文件。

2.添加上题中的代码后,再在 nemu/include/debug.h中添加一行 volatile static int dummy; 然后重新编译 NEMU.请问此时的 NEMU 含有多少个 dummy 变量的实体?与上题中 dummy 变量实体数目进行比较,并解释本题的结果。

答：相等，都是 29 个。

3.修改添加的代码,为两处 dummy 变量进行初始化:volatile static int dummy = 0; 然后重新编译 NEMU.你发现了什么问题? 为什么之前没有出现这样的问题?(回答完本题后可以删除添加的代码.)

答：如图所示，看报错，这两个文件都与 serial.c 文件，编译生成文件 serial.o 文件时，会分别从这两个文件搜索 dummy 变量，发现重定义，因此报错。

```
zhaoweikang@zhaoweikang:~/ics2017/nemu$ sudo make
+ CC src/device/serial.c
In file included from src/device/serial.c:1:0:
./include/common.h:16:21: error: redefinition of ‘dummy’
 volatile static int dummy =0;
                     ^~~~~
In file included from ./include/common.h:10:0,
                 from src/device/serial.c:1:
./include/debug.h:6:21: note: previous definition of ‘dummy’ was here
 volatile static int dummy =0;
                     ^~~~~
Makefile:23: recipe for target 'build/obj/device/serial.o' failed
make: *** [build/obj/device/serial.o] Error 1
```

了解 Makefile

请描述你在 nemu 目录下敲入 make 后，make 程序如何组织.c 和.h 文件，最终生成可执行文件 nemu/build/nemu.(这个问题包括两个方面:Makefile 的工作方式和编译链接的过程.)

答：在命令行键入 sudo man make，找到-n 选项，如图，输入 make 命令之后，编译器会对你刚才修改过的.c 文件以及与之相关联的.c 与.h 文件进行联编，然后打印出要执行的命令。

```
   -n, --just-print, --dry-run, --recon
        Print the commands that would be executed, but do not execute them
        (except in certain circumstances).
```