计算机系统设计 PA2

> 李潇逸 2111454

运行第一个 C 程序

在实验开始之前直接使用 `make ARCH=x86-nemu ALL=dummy run` 运行 `dummy.c` ，得到提示未实现指令信息，也就是在 `eip(0x0010000a)` 处指令有缺失。

```
sesame@sesame-virtual-machine:~/ics2024/nexus-am/tests/cputest$ make ARCH=x86-nemu ALL=dummy run
Building dummy [x86-nemu]
Building am [x86-nemu]
make[2]: *** 没有指明目标并且找不到 makefile。 停止。
[src/monitor/monitor.c,65,load_img] The image is /home/sesame/ics2024/nexus-am/tests/cputest/build/dummy-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 10:55:28, Mar 31 2024
For help, type "help"
(nemu) c
invalid opcode(eip = 0x0010000a): e8 01 00 00 00 90 55 89 ...

There are two cases which will trigger this unexpected exception:
1. The instruction at eip = 0x0010000a is not implemented.
2. Something is implemented incorrectly.
Find this eip(0x0010000a) in the disassembling result to distinguish which case it is.

If it is the first case, see

for more details.

If it is the second case, remember:
* The machine is always right!
* Every line of untested code is always wrong!
```

反编译结果如下：

```
/home/sesame/ics2024/nexus-am/tests/cputest/build/dummy-x86-nemu:     文件格式 elf32-i386


Disassembly of section .text:

00100000 <_start>:
  100000:   bd 00 00 00 00          mov    $0x0,%ebp
  100005:   bc 00 7c 00 00          mov    $0x7c00,%esp
  10000a:   e8 01 00 00 00          call   100010 <_trm_init>
  10000f:   90                      nop
```

查询手册

首先根据 `0x0010000a` 处的指令（call）发现该指令符号是 `e8` ，根据此符号向 `i386手册` 附录A 进行查询，发现对应的指令解释是 `Av` 。

CALL
Av

向上翻阅得知相关字母的含义：

- A： 直接寻址；指令没有modR/M字节；操作数的地址被编码在指令中；不能应用基寄存器、索引寄存器或缩放因子；例如，远跳转（EA）。
- v： 取决于操作数大小属性，字或双字。

```
A  Direct address; the instruction has no modR/M byte; the address of the
   operand is encoded in the instruction; no base register, index register,
   or scaling factor can be applied; e.g., far JMP (EA).


v  Word or double word, depending on operand size attribute.
```

其他指令查询方式同上，具体如下表。

指令	操作码	描述
call	e8	Call near, displacement relativeto next instruction
push	50+rw/rb	Push register word/dword
sub	83	Subtract sign-extended immediate byte from r/m word
xor	31	Exclusive-OR dword register to r/m word/dword
pop	58+rw/rb	Pop top of stack into word/dword register
ret	c3	Return (near) to caller

译码表

在 `nemu/src/cpu/exec/all-instr.h` 中添加指令名称

> 将这一步放在第一位是由于自身的血泪史。在一开始进行编写时只填写了 `opcode_table` ，并没有进行这一步的定义，最终导致 `debug` 了将近4小时。

```
make_EHelper(mov);
make_EHelper(call);
make_EHelper(sub);
make_EHelper(xor);
make_EHelper(push);
make_EHelper(sub);
make_EHelper(pop);
make_EHelper(ret);
make_EHelper(operand_size);
```

之后开始进行 `opcode_table` 的填写，具体如下图：

```
  /* 0x30 */    IDEXW(G2E, xor, 1), IDEX(G2E, xor), IDEXW(E2G, xor, 1), IDEX(E2G, xor),
  /* 0x34 */    IDEXW(I2a, xor, 1), IDEX(I2a, xor), EMPTY, EMPTY,
```

```
  /* 0x50 */    IDEX(r, push), IDEX(r, push), IDEX(r, push), IDEX(r, push),
  /* 0x54 */    IDEX(r, push), IDEX(r, push), IDEX(r, push), IDEX(r, push),
  /* 0x58 */    IDEX(r, pop), IDEX(r, pop), IDEX(r, pop), IDEX(r, pop),
  /* 0x5c */    IDEX(r, pop), IDEX(r, pop), IDEX(r, pop), IDEX(r, pop),
```

```
/* 0xc0 */    IDEXW(gp2_Ib2E, gp2, 1), IDEX(gp2_Ib2E, gp2), EX(ret), EX(ret),
```

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

rtl

push 和 pop

```
static inline void rtl_push(const rtlreg_t* src1) {
  // esp <- esp - 4
  rtl_subi(&cpu.esp, &cpu.esp, 4);

  // M[esp] <- src1
  rtl_sm(&cpu.esp, 4, src1);
}

static inline void rtl_pop(rtlreg_t* dest) {
  // dest <- M[esp]
  rtl_lm(dest, &cpu.esp, 4);

  // esp <- esp + 4
  rtl_addi(&cpu.esp, &cpu.esp, 4);
}
```

更新标志位

```
static inline void rtl_msb(rtlreg_t* dest, const rtlreg_t* src1, int width) {
  // dest <- src1[width * 8 - 1]
  rtl_shri(dest, src1, width*8-1);
  rtl_andi(dest, dest, 0x1);
}

static inline void rtl_update_ZF(const rtlreg_t* result, int width) {
  // eflags.ZF <- is_zero(result[width * 8 - 1 .. 0])
  rtl_andi(&t0, result, (0xFFFFFFFFu >> (4-width)*8));
  rtl_eq0(&t0, &t0);
  rtl_set_ZF(&t0);
}

static inline void rtl_update_SF(const rtlreg_t* result, int width) {
  // eflags.SF <- is_sign(result[width * 8 - 1 .. 0])
  rtl_msb(&t0, result, width);
  rtl_set_SF(&t0);
}

static inline void rtl_update_ZFSF(const rtlreg_t* result, int width) {
  rtl_update_ZF(result, width);
  rtl_update_SF(result, width);
}
```

call

```
make_EHelper(call) {
  // the target address is calculated at the decode stage
  rtl_li(&t2, decoding.seq_eip);
  rtl_push(&t2);
  decoding.is_jmp = 1;

  print_asm("call %x", decoding.jmp_eip);
}
```

sub

```
make_EHelper(sub) {
  eflags_modify();
  operand_write(id_dest, &t2);

  print_asm_template2(sub);
}
```

xor

```
make_EHelper(xor) {
  rtl_xor(&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(xor);
}
```

ret

```
make_EHelper(ret) {
  rtl_pop(&t2);
  decoding.jmp_eip = t2;
  decoding.is_jmp = 1;

  print_asm("ret");
}
```

实验结果

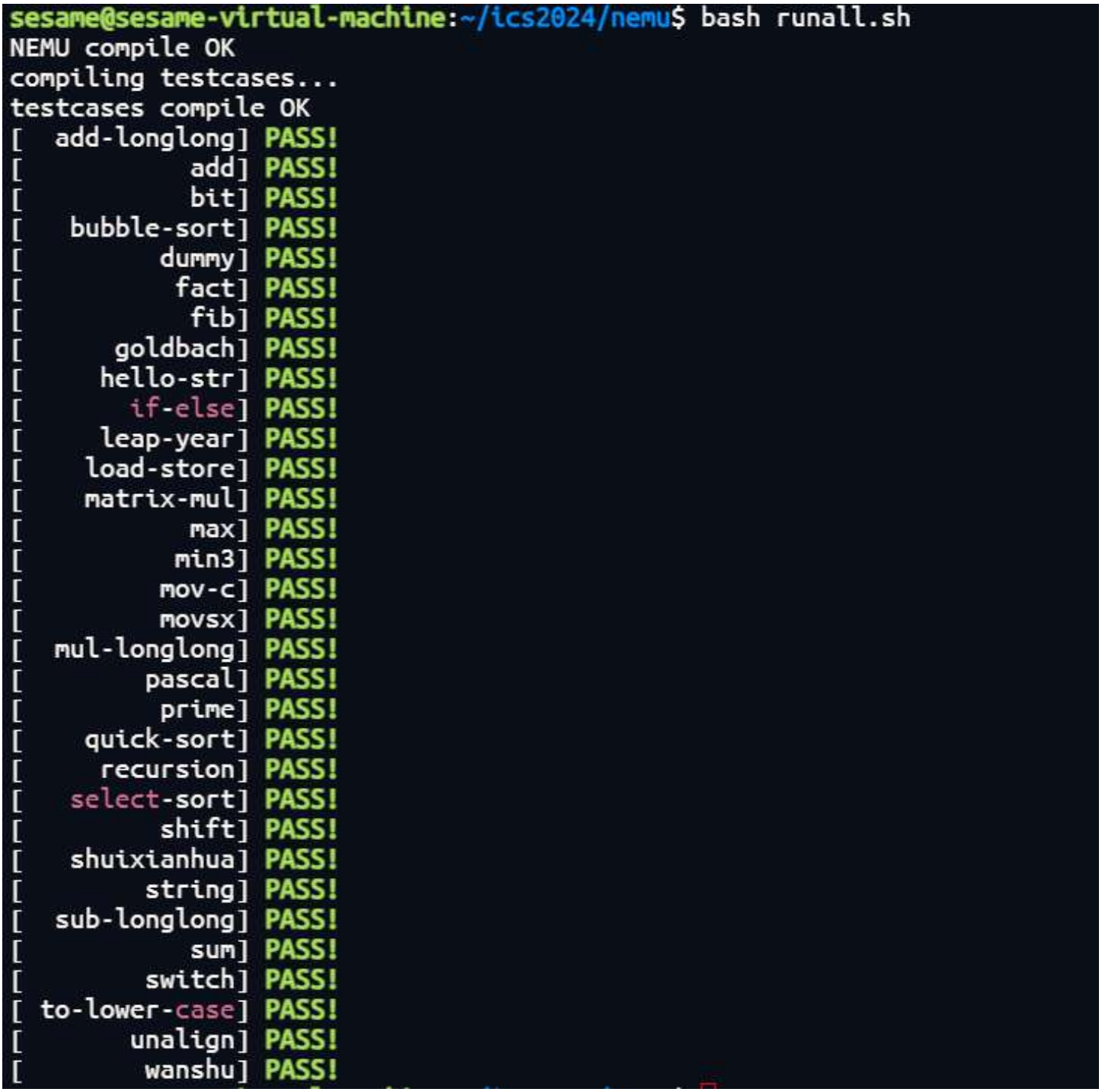
```
sesame@sesame-virtual-machine:~/ics2024/nexus-am/tests/cputest$ make ARCH=x86-nemu ALL=dummy run
Building dummy [x86-nemu]
Building am [x86-nemu]
+ CC arch/x86-nemu/src/trm.c
+ AR /home/sesame/ics2024/nexus-am/am/build/am-x86-nemu.a
make[2]: *** 没有指明目标并且找不到 makefile。 停止。
[src/monitor/monitor.c,65,load_img] The image is /home/sesame/ics2024/nexus-am/tests/cputest/build/dummy-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 19:21:41, Mar 31 2024
For help, type "help"
(nemu) c
nemu: HIT GOOD TRAP at eip = 0x00100052

(nemu) c
Program execution has ended. To restart the program, exit NEMU and run again.
(nemu) q
 dummy
```

运行更多程序

填写opcode_table

```
opcode_entry opcode_table [512] = {
  /* 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,
  /* 0x08 */    IDEXW(G2E, or, 1), IDEX(G2E, add), IDEXW(E2G, or, 1), IDEX(E2G, add),
  /* 0x0c */    IDEXW(I2a, or, 1), IDEX(I2a, add), EMPTY, EX(2byte_esc),
  /* 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,
  /* 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,
  /* 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,
  /* 0x28 */    IDEXW(G2E, sub, 1), IDEX(G2E, sub), IDEXW(E2G, sub, 1), IDEX(E2G, sub),
  /* 0x2c */    IDEXW(I2a, sub, 1), IDEX(I2a, sub), EMPTY, EMPTY,
  /* 0x30 */    IDEXW(G2E, xor, 1), IDEX(G2E, xor), IDEXW(E2G, xor, 1), IDEX(E2G, xor),
  /* 0x34 */    IDEXW(I2a, xor, 1), IDEX(I2a, xor), EMPTY, EMPTY,
  /* 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,
  /* 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),
  /* 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),
  /* 0x50 */    IDEX(r, push), IDEX(r, push), IDEX(r, push), IDEX(r, push),
  /* 0x54 */    IDEX(r, push), IDEX(r, push), IDEX(r, push), IDEX(r, push),
  /* 0x58 */    IDEX(r, pop), IDEX(r, pop), IDEX(r, pop), IDEX(r, pop),
  /* 0x5c */    IDEX(r, pop), IDEX(r, pop), IDEX(r, pop), IDEX(r, pop),
  /* 0x60 */    EX(pusha), EX(popa), EMPTY, EMPTY,
```

一键回归测试


sesame@sesame-virtual-machine:~/ics2024/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!

加入IOE

串口

```
make_EHelper(in) {
  rtl_li(&t0, pio_read(id_src->val, id_dest->width));
  operand_write(id_dest, &t0);

  print_asm_template2(in);

#ifdef DIFF_TEST
  diff_test_skip_qemu();
#endif
}

make_EHelper(out) {
  pio_write(id_dest->val, id_src->width, id_src->val);

  print_asm_template2(out);

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

运行 hello world

```
sesame@sesame-virtual-machine:~/ics2024/nexus-am/apps/hello$ make run
Building hello [native]
make[1]: Entering directory '/home/sesame/ics2024/nexus-am'
make[2]: Entering directory '/home/sesame/ics2024/nexus-am/am'
Building am [native]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/sesame/ics2024/nexus-am/am'
make[1]: Leaving directory '/home/sesame/ics2024/nexus-am'
make[1]: Entering directory '/home/sesame/ics2024/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/sesame/ics2024/nexus-am/libs/klib'
/home/sesame/ics2024/nexus-am/Makefile.compile:86: recipe for target 'klib' failed
make: [klib] Error 2 (ignored)
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

时钟

```c
#include "device/port-io.h"
#include "monitor/monitor.h"
#include <sys/time.h>

#define RTC_PORT 0x48   // Note that this is not the standard

void timer_intr() {
  if (nemu_state == NEMU_RUNNING) {
    extern void dev_raise_intr(void);
    dev_raise_intr();
  }
}

static uint32_t *rtc_port_base;

void rtc_io_handler(ioaddr_t addr, int len, bool is_write) {
  if (!is_write) {
    struct timeval now;
    gettimeofday(&now, NULL);
    uint32_t seconds = now.tv_sec;
    uint32_t useconds = now.tv_usec;
    rtc_port_base[0] = seconds * 1000 + (useconds + 500) / 1000;
  }
}

void init_timer() {
  rtc_port_base = add_pio_map(RTC_PORT, 4, rtc_io_handler);
}

```

运行 timetest

```
sesame@sesame-virtual-machine:~/ics2024/nexus-am/tests/timetest$ make run
Building timetest [native]
make[1]: Entering directory '/home/sesame/ics2024/nexus-am'
make[2]: Entering directory '/home/sesame/ics2024/nexus-am/am'
Building am [native]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/sesame/ics2024/nexus-am/am'
make[1]: Leaving directory '/home/sesame/ics2024/nexus-am'
make[1]: Entering directory '/home/sesame/ics2024/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/sesame/ics2024/nexus-am/libs/klib'
/home/sesame/ics2024/nexus-am/Makefile.compile:86: recipe for target 'klib' failed
make: [klib] Error 2 (ignored)
1 second.
2 seconds.
3 seconds.
4 seconds.
5 seconds.
6 seconds.
7 seconds.
8 seconds.
9 seconds.
10 seconds.
11 seconds.
12 seconds.
13 seconds.
14 seconds.
15 seconds.
16 seconds.
17 seconds.
18 seconds.
19 seconds.
20 seconds.
```

键盘

```
void send_key(uint8_t scancode, bool is_keydown) {
  if (nemu_state == NEMU_RUNNING &&
      keymap[scancode] != _KEY_NONE) {
    uint32_t am_scancode = keymap[scancode] | (is_keydown ? KEYDOWN_MASK : 0);
    key_queue[key_r] = am_scancode;
    key_r = (key_r + 1) % KEY_QUEUE_LEN;
  }
}

void i8042_io_handler(ioaddr_t addr, int len, bool is_write) {
  if (!is_write) {
    if (addr == I8042_DATA_PORT) {
      i8042_status_port_base[0] &= ~I8042_STATUS_HASKEY_MASK;
    }
    else if (addr == I8042_STATUS_PORT) {
      if ((i8042_status_port_base[0] & I8042_STATUS_HASKEY_MASK) == 0) {
        if (key_f != key_r) {
          i8042_data_port_base[0] = key_queue[key_f];
          i8042_status_port_base[0] |= I8042_STATUS_HASKEY_MASK;
          key_f = (key_f + 1) % KEY_QUEUE_LEN;
        }
      }
    }
  }
}
```

运行 keytest

```
Get key: 43 A down
Get key: 43 A up
Get key: 44 S down
Get key: 44 S up
Get key: 44 S down
Get key: 44 S up
Get key: 44 S down
Get key: 44 S up
Get key: 46 F down
Get key: 46 F up
Get key: 44 S down
Get key: 44 S up
Get key: 45 D down
Get key: 45 D up
Get key: 43 A down
Get key: 43 A up
Get key: 43 A down
Get key: 43 A up
Get key: 46 F down
Get key: 46 F up
Get key: 44 S down
Get key: 45 D down
Get key: 44 S up
Get key: 45 D up
```

> 注意：从现在开始一些看似无法跑通的实验不要怀疑自己，放弃 `ssh` 软件进入虚拟机九成可以解决。坑死我了！(ﾒ °Д°)ﾒ ┻┻

VGA

```
void update_screen() {
  SDL_UpdateTexture(texture, NULL, vmem, SCREEN_W * sizeof(vmem[0][0]));
  SDL_RenderClear(renderer);
  SDL_RenderCopy(renderer, texture, NULL, NULL);
  SDL_RenderPresent(renderer);
}

void init_vga() {
  SDL_Init(SDL_INIT_VIDEO);
  SDL_CreateWindowAndRenderer(SCREEN_W * 2, SCREEN_H * 2, 0, &window, &renderer);
  SDL_SetWindowTitle(window, "NEMU");
  texture = SDL_CreateTexture(renderer, SDL_PIXELFORMAT_ARGB8888,
      SDL_TEXTUREACCESS_STATIC, SCREEN_W, SCREEN_H);

  vmem = add_mmio_map(VMEM, 0x80000, vga_vmem_io_handler);
}
```

运行 videotest

运行打字小游戏

运行红白机

> 可惜不能控制，不知道是不是我的问题？

一些 bug

一些问题已经写在上面，这里只说一个最令我哭笑不得、头发狂掉的bug：在进行一键回归测试时发现一个案例都过不了，此时明明已经在之前通过了个案测试。花费了一下午时间发现是位于 `ics2024/nemu/include/common.h` 的两个宏定义 `#define DEBUG` 和 `#define DIFF_TEST` 没有被注释，注释掉后就可以解决。为什么呢？咱也不敢说，咱也不敢问（主要是很有可能根本原因是我自己没有实现好 ..(。•ˇ‸ˇ•。)...）

必答题

static inline

- 经查阅资料后发现

    - inline实际上表示建议内联，但并非强制内联，编译器可以忽略。
    - 若想要确保内联，在使用inline的时候要加入static，否则inline不内联的时候就和普通函数在头文件中的定义是一样的，若多个c文件都包含时就会产生歧义，加入static后代码健壮性高，
- 若只使用inline，编译器内联，两函数实际效果一致。
- 编译器有个原则，以 c/cpp 文件为单位进行逐个编译 obj（中间目标文件）， 每个 c 文件的编译是独立的，该 c 文件用到的外部函数都在编译时预留一个符号，只有等到所有 obj 生成后，链接时才给这些符号地址（链接脚本决定地址）。 对于每个 c/cpp 来说，都包含了 Func 的声明和实现（重定义），而在链接时，链接器会在所有的 Object File 中找寻函数的实现，由于不清楚到底是链接了哪个同名函数，故链接报错

makefile

- makefile 的显式规则:
    - 基本格式：用于定义显示规则，说明如何生成一个或多个目标文件
    - target- 目标文件, 可以是 Object File, 也可以是可执行文件
    - prerequisites - 生成 target 所需要的依赖文件或者目标
    - command- make 需要执行的命令(任意的 shell 命令), Makefile 中必须以 [tab] 开头。
- 基本工作方式
    - 读入主 Makefile (主 Makefile 中可以引用其他 Makefile)
    - 读入被 include 的其他 Makefile
    - 初始化文件中的变量
    - 推导隐晦规则, 并分析所有规则
    - 为所有的目标文件创建依赖关系链
    - 根据依赖关系, 决定哪些目标要重新生成
    - 执行生成命令
- 隐含规则： 自动推导功能
    - 将所有的 `name.o` 的依赖自动推导为 `name.c` 并使用规则 `(CC) -c (FLAGS) (CPPFLAGS)` 得到目标。该规则中只有 `-c` 是隐含规则中的，后面两个为隐含变量，留给用户使用的。
    - `name` 目标依赖于 `name.o` ，其生成命令是： `(CC)(LDFLAGS)name.o (LOADLIBES)(LDLIBS)` 。该规则对于多个被依赖的目标文件同样有效，如 `name : y.o z.o` 。
- 隐含规则所使用的变量（隐含变量）分为两类：
    - 代表一个程序的名字（例如：“**CC**”代表了编译器“**cc**”这个可执行程序）。
    - 代表执行这个程序使用的参数（例如：变量“**CFLAGS**”，多个参数使用空格分开）： **CC**： C 编程程序。默认是“**cc**”。**CFLAGS**： 执行“**CC**”编译器的命令行参数（编译 `.c` 源文件的选项）。
- makefile 的变量
    - 基本变量
        - 实现各文件的路径/名称声明
        - 定义编译的源文件与中间目标文件
- 高级变量
    - $<：规则型变量，表示第一个依赖文件

    - $@：表示目标文件，从而将所有 `.c` 文件编译生成对应的 `.o` 文件。
- makefile 的包含文件特性
    - 包含了当前文件夹下的 `Makefile.git`，其中定义了 `git_commit(msg)` 函数，从而实现在 `Makefile` 中对函数的调用