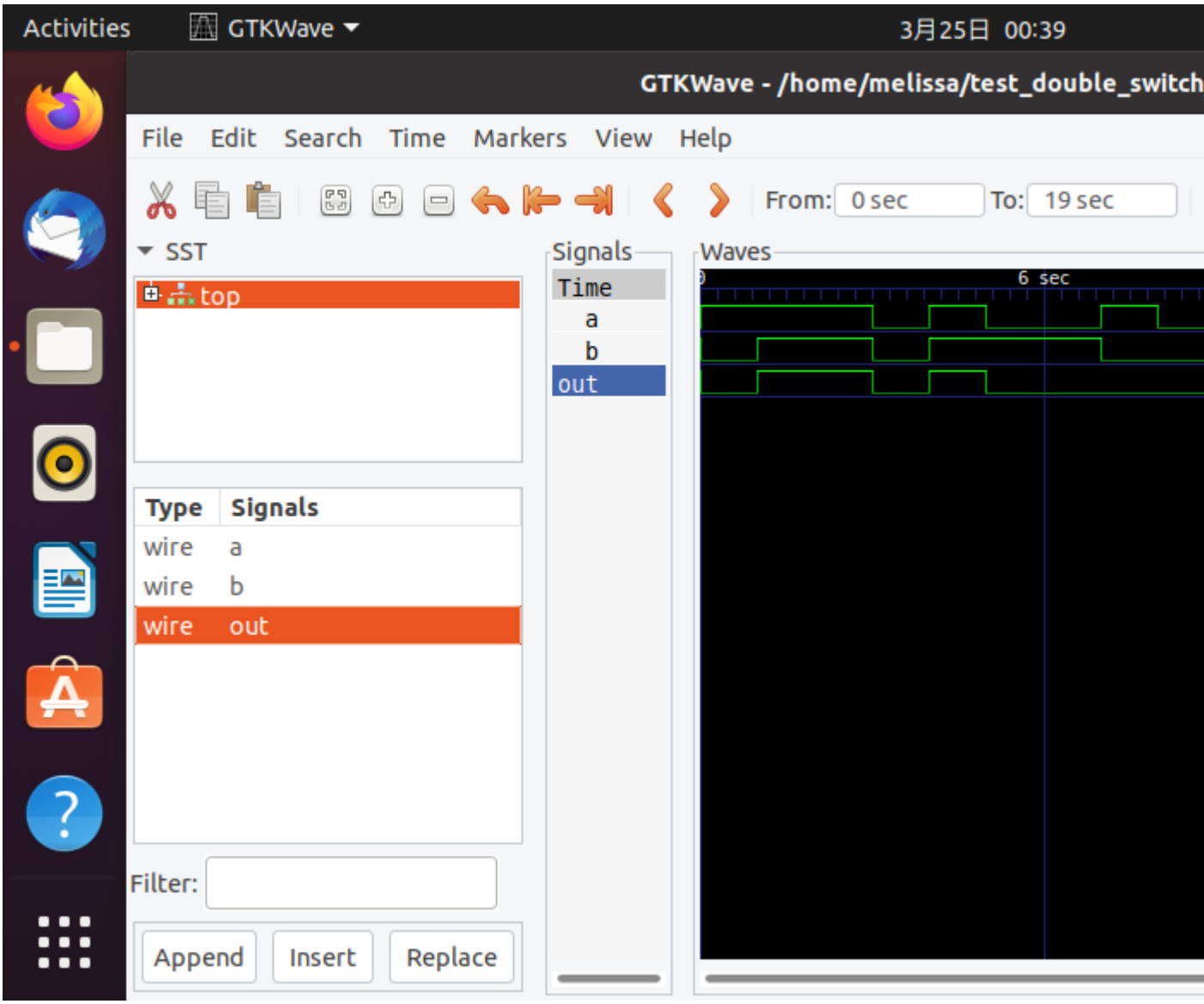
ysyx_22040175-实验报告

波形仿真

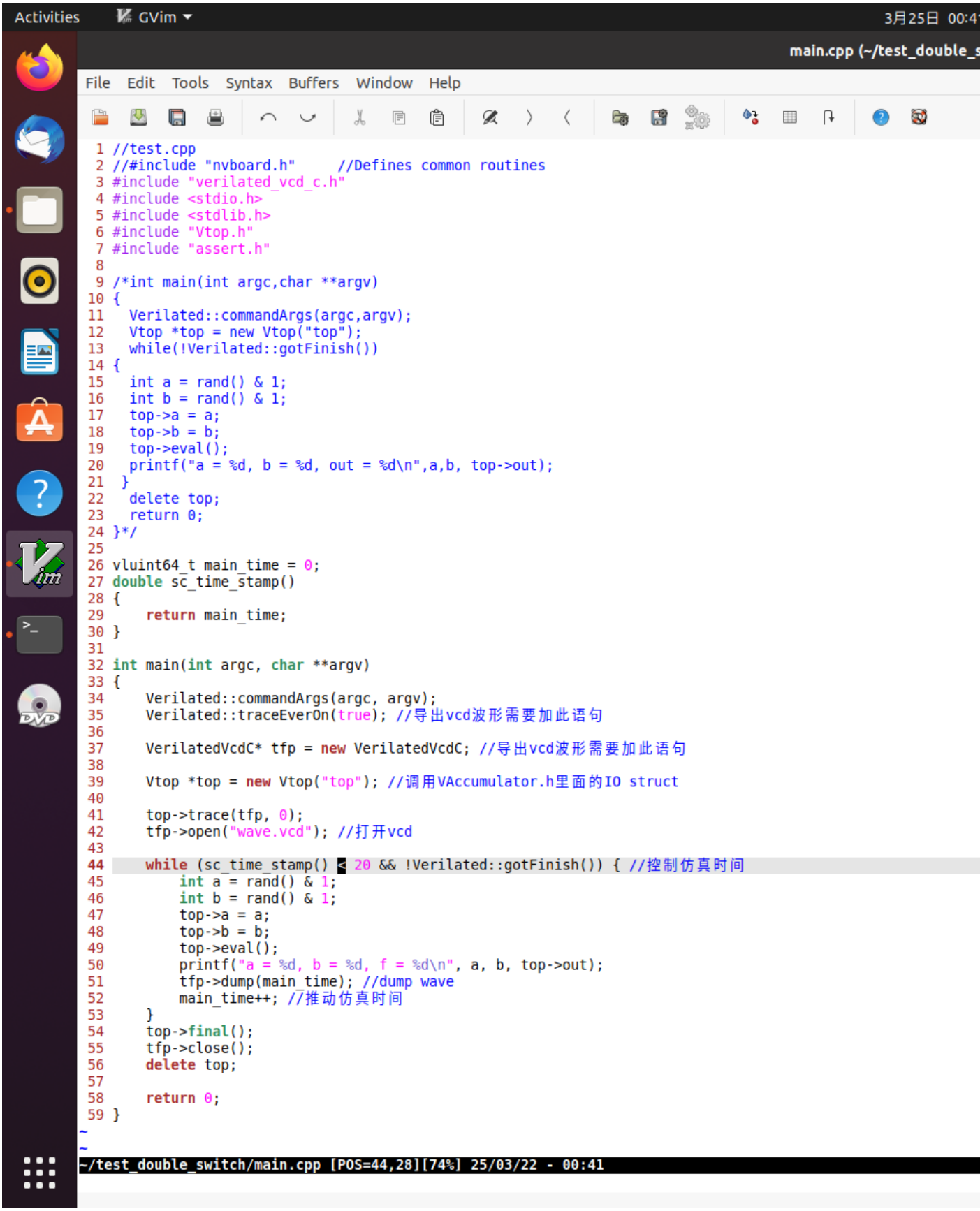


makefile

```
build:
	verilator -Wall --trace --cc top.v --exe main.cpp

comp:
	make -C obj_dir -j -f Vtop.mk Vtop

run:
```

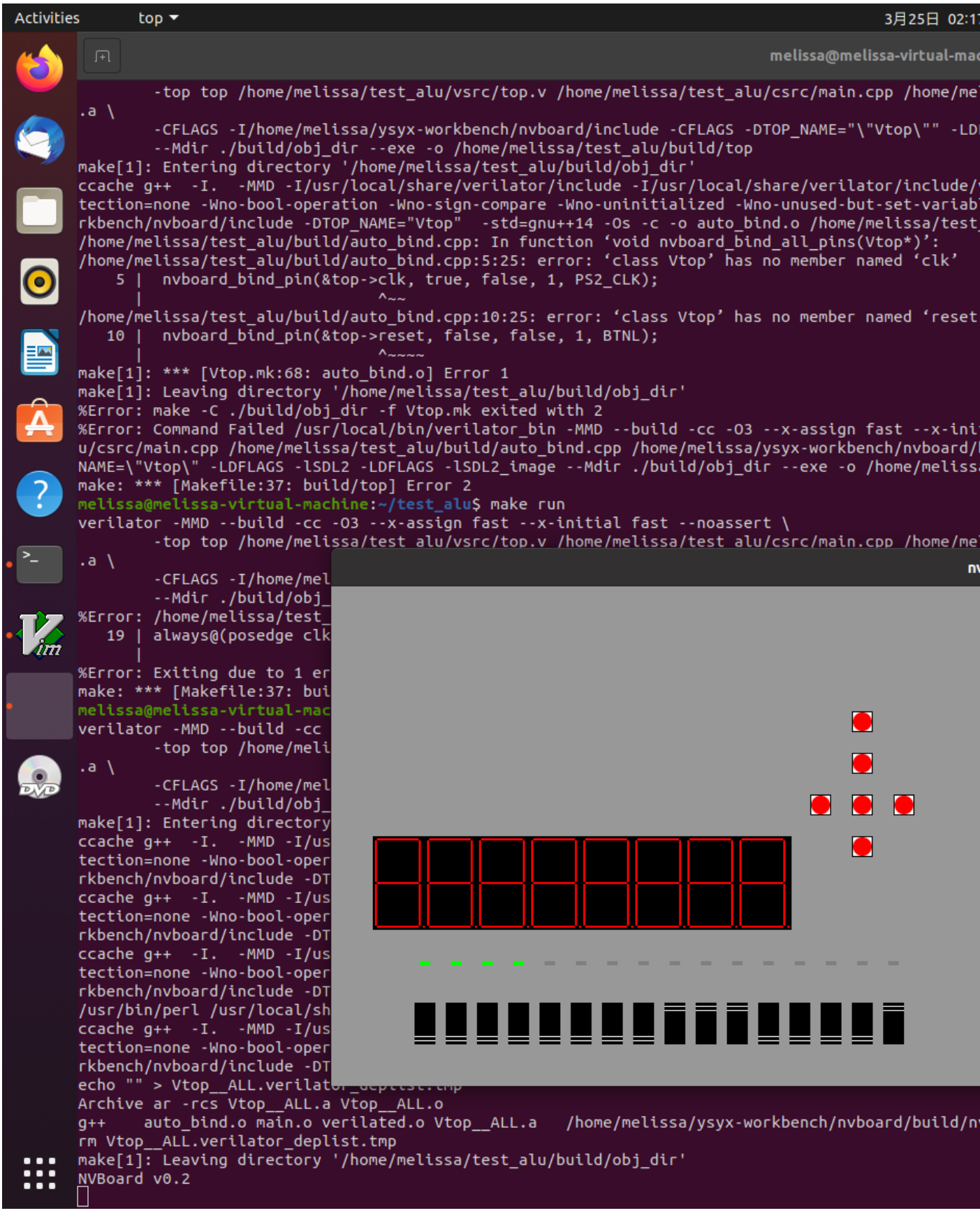
obj_dir/Vtop

main.cpp



注意：cpp 里没用时钟，.v 直接 assign 的

ALU


Activities
top
melissa@melissa-virtual-mac
-top top /home/melissa/test_alu/vsrc/top.v /home/melissa/test_alu/csrc/main.cpp /home/me
.a \
-CFLAGS -I/home/melissa/ysyx-workbench/nvboard/include -CFLAGS -DTOP_NAME="\"Vtop\"" -LD
--Mdir ./build/obj_dir --exe -o /home/melissa/test_alu/build/top
make[1]: Entering directory '/home/melissa/test_alu/build/obj_dir'
ccache g++ -I. -MMD -I/usr/local/share/verilator/include -I/usr/local/share/verilator/include/
tection=none -Wno-bool-operation -Wno-sign-compare -Wno-uninitialized -Wno-unused-but-set-variab
rkbench/nvboard/include -DTOP_NAME="Vtop" -std=gnu++14 -Os -c -o auto_bind.o /home/melissa/test
/home/melissa/test_alu/build/auto_bind.cpp: In function 'void nvboard_bind_all_pins(Vtop*)':
/home/melissa/test_alu/build/auto_bind.cpp:5:25: error: 'class Vtop' has no member named 'clk'
5 | nvboard_bind_pin(&top->clk, true, false, 1, PS2_CLK);
| ^~~
/home/melissa/test_alu/build/auto_bind.cpp:10:25: error: 'class Vtop' has no member named 'reset
10 | nvboard_bind_pin(&top->reset, false, false, 1, BTNL);
| ^~~~~
make[1]: *** [Vtop.mk:68: auto_bind.o] Error 1
make[1]: Leaving directory '/home/melissa/test_alu/build/obj_dir'
%Error: make -C ./build/obj_dir -f Vtop.mk exited with 2
%Error: Command Failed /usr/local/bin/verilator_bin -MMD --build -cc -O3 --x-assign fast --x-ini
u/csrc/main.cpp /home/melissa/test_alu/build/auto_bind.cpp /home/melissa/ysyx-workbench/nvboard/
NAME=\"Vtop\" -LDFLAGS -lSDL2 -LDFLAGS -lSDL2_image --Mdir ./build/obj_dir --exe -o /home/meliss
make: *** [Makefile:37: build/top] Error 2
melissa@melissa-virtual-machine:~/test_alu$ make run
verilator -MMD --build -cc -O3 --x-assign fast --x-initial fast --noassert \
-top top /home/melissa/test_alu/vsrc/top.v /home/melissa/test_alu/csrc/main.cpp /home/me
.a \
-CFLAGS -I/home/mel
--Mdir ./build/obj_
%Error: /home/melissa/test_
19 | always@(posedge clk
|
%Error: Exiting due to 1 er
make: *** [Makefile:37: bui
melissa@melissa-virtual-mac
verilator -MMD --build -cc
-top top /home/meli
.a \
-CFLAGS -I/home/mel
--Mdir ./build/obj_
make[1]: Entering directory
ccache g++ -I. -MMD -I/us
tection=none -Wno-bool-oper
rkbench/nvboard/include -DT
ccache g++ -I. -MMD -I/us
tection=none -Wno-bool-oper
rkbench/nvboard/include -DT
ccache g++ -I. -MMD -I/us
tection=none -Wno-bool-oper
rkbench/nvboard/include -DT
/usr/bin/perl /usr/local/sh
ccache g++ -I. -MMD -I/us
tection=none -Wno-bool-oper
rkbench/nvboard/include -DT
echo "" > Vtop__ALL.verilator_deplist.tmp
Archive ar -rcs Vtop__ALL.a Vtop__ALL.o
g++ auto_bind.o main.o verilated.o Vtop__ALL.a /home/melissa/ysyx-workbench/nvboard/build/n
rm Vtop__ALL.verilator_deplist.tmp
make[1]: Leaving directory '/home/melissa/test_alu/build/obj_dir'
NVBoard v0.2

实现了简单的加减，同或异或，

PA1

单步执行

```c
static int cmd_si(char *args) {
  /* extract the first argument */
  char *arg = strtok(NULL, " ");
  if (arg == NULL) {
    cpu_exec(1);
  }
  else {
    int N = atoi(arg);
    cpu_exec(N);
    }
  return 0;
}
```

```
melissa@melissa-virtual-machine:~/ysyx-workbench/nemu$ make run
+ CC src/monitor/sdb/sdb.c
+ LD /home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter
/home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter --log=/home/
log.txt
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000
[src/monitor/monitor.c:36 load_img] No image is given. Use the default build-
[src/monitor/monitor.c:13 welcome] Trace: ON
[src/monitor/monitor.c:14 welcome] If trace is enabled, a log file will be ge
ead to a large log file. If it is not necessary, you can disable it in menuco
[src/monitor/monitor.c:17 welcome] Build time: 03:02:00, Mar 14 2022
Welcome to riscv64-NEMU!
For help, type "help"
(nemu) si
0x0000000080000000: 97 02 00 00
(nemu) si 2
0x0000000080000004: 23 b8 02 00
0x0000000080000008: 03 b5 02 01
2
(nemu) si 3
0x000000008000000c: 73 00 10 00
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT GOOD TRAP at pc = 0x0000000080000
[src/cpu/cpu-exec.c:69 statistic] host time spent = 108 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 4
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 37,037 inst/s
3
(nemu)
```

打印寄存器

```
static int cmd_info(char *args) {
  /* extract the first argument */
  char *arg = strtok(NULL, " ");

  if (arg == NULL) {
  }
  else {
    if (*arg == 'r')
     isa_reg_display();
    }
  return 0;
}
```

发现打印出来的只有名称，与其他同学的不一样，找到 display 函数，调用 cpu.gpr 改变其输出

```
void isa_reg_display() {
  for(int i=0;i<32;i++){
  printf("%s %08lx\n", regs[i], cpu.gpr[i]);
  }
}
```

```
[src/monitor/monitor.c:14 welcome] If trace is enabled, a log file will be ger
ead to a large log file. If it is not necessary, you can disable it in menucor
[src/monitor/monitor.c:17 welcome] Build time: 03:02:00, Mar 14 2022
Welcome to riscv64-NEMU!
For help, type "help"
(nemu) info r
$0 00000000
ra 00000000
sp 00000000
gp 00000000
tp 00000000
t0 00000000
t1 00000000
t2 00000000
s0 00000000
s1 00000000
a0 00000000
a1 00000000
a2 00000000
a3 00000000
a4 00000000
a5 00000000
a6 00000000
a7 00000000
s2 00000000
s3 00000000
s4 00000000
s5 00000000
s6 00000000
s7 00000000
s8 00000000
s9 00000000
s10 00000000
s11 00000000
t3 00000000
t4 00000000
t5 00000000
```

打印内存数据

```
static int cmd_x(char *args) {
  /* extract the first argument */
  char *arg = strtok(NULL, " ");
  int arg_l = atoi(arg);
  arg = strtok(NULL, " ");
  word_t addr = strtol(arg,NULL,16);
  for (int i=0;i<arg_l;i++){
        printf("%ld\n",paddr_read(addr, 4));
        addr +=4;
  }


  return 0;
}
```

```
melissa@melissa-virtual-machine:~/ysyx-workbench/nemu$ make run
+ LD /home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter
/home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter --log=/home/
log.txt
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000
[src/monitor/monitor.c:36 load_img] No image is given. Use the default build-
[src/monitor/monitor.c:13 welcome] Trace: ON
[src/monitor/monitor.c:14 welcome] If trace is enabled, a log file will be ge
ead to a large log file. If it is not necessary, you can disable it in menuco
[src/monitor/monitor.c:17 welcome] Build time: 03:02:00, Mar 14 2022
Welcome to riscv64-NEMU!
For help, type "help"
(nemu) x 2 0x80000000
663
178211
(nemu)
```

表达式求值

C 语言中使用正则表达式一般分为三步：

1. 编译正则表达式 regcomp()
2. 匹配正则表达式 regexec()
3. 释放正则表达式 regfree()

1、int regcomp (regex_t *compiled, const char *pattern, int cflags)

这个函数把指定的正则表达式 pattern 编译成一种特定的数据格式 compiled，这样可以使匹配更有效。函数 regexec 会使用这个数据在目标文本串中进行模式匹配。执行成功返回 0 。

参数说明：

①regex_t 是一个结构体数据类型，用来存放编译后的正则表达式，它的成员 re_nsub 用来存储正则表达式中的子正则表达式的个数，子正则表达式就是用圆括号包起来的部分表达式。

②pattern 是指向我们写好的正则表达式的指针。

③cflags 有如下 4 个值或者是它们或运算(|)后的值：

REG_EXTENDED 以功能更加强大的扩展正则表达式的方式进行匹配。

REG_ICASE 匹配字母时忽略大小写。

REG_NOSUB 不用存储匹配后的结果。

REG_NEWLINE 识别换行符，这样'$'就可以从行尾开始匹配，'^'就可以从行的开头开始匹配。

2. int regexec (regex_t *compiled, char *string, size_t nmatch, regmatch_t matchptr [], int eflags)

当我们编译好正则表达式后，就可以用 regexec 匹配我们的目标文本串了，如果在编译正则表达式的时候没有指定 cflags 的参数为 REG_NEWLINE，则默认情况下是忽略换行符的，也就是把整个文本串当作一个字符串处理。执行成功返回 0 。

regmatch_t 是一个结构体数据类型，在 regex.h 中定义：

```
typedef struct
{
   regoff_t rm_so;
   regoff_t rm_eo;
} regmatch_t;
```

成员 rm_so 存放匹配文本串在目标串中的开始位置，rm_eo 存放结束位置。通常我们以数组的形式定义一组这样的结构。因为往往我们的正则表达式中还包含子正则表达式。数组 0 单元存放主正则表达式位置，后边的单元依次存放子正则表达式位置。

参数说明：

①compiled 是已经用 regcomp 函数编译好的正则表达式。

②string 是目标文本串。

③nmatch 是 regmatch_t 结构体数组的长度。

④matchptr regmatch_t 类型的结构体数组，存放匹配文本串的位置信息。

⑤eflags 有两个值

REG_NOTBOL 按我的理解是如果指定了这个值，那么'^'就不会从我们的目标串开始匹配。总之

我到现在还不是很明白这个参数的意义；

REG_NOTEOL 和上边那个作用差不多，不过这个指定结束 end of line。

3. void regfree (regex_t *compiled)

当我们使用完编译好的正则表达式后，或者要重新编译其他正则表达式的时候，我们可以用这个函数清空 compiled 指向的 regex_t 结构体的内容，请记住，如果是重新编译的话，一定要先清空 regex_t 结构体。

4. size_t regerror (int errcode, regex_t *compiled, char *buffer, size_t length)

当执行 regcomp 或者 regexec 产生错误的时候，就可以调用这个函数而返回一个包含错误信息的字符串。

参数说明：

①errcode 是由 regcomp 和 regexec 函数返回的错误代号。

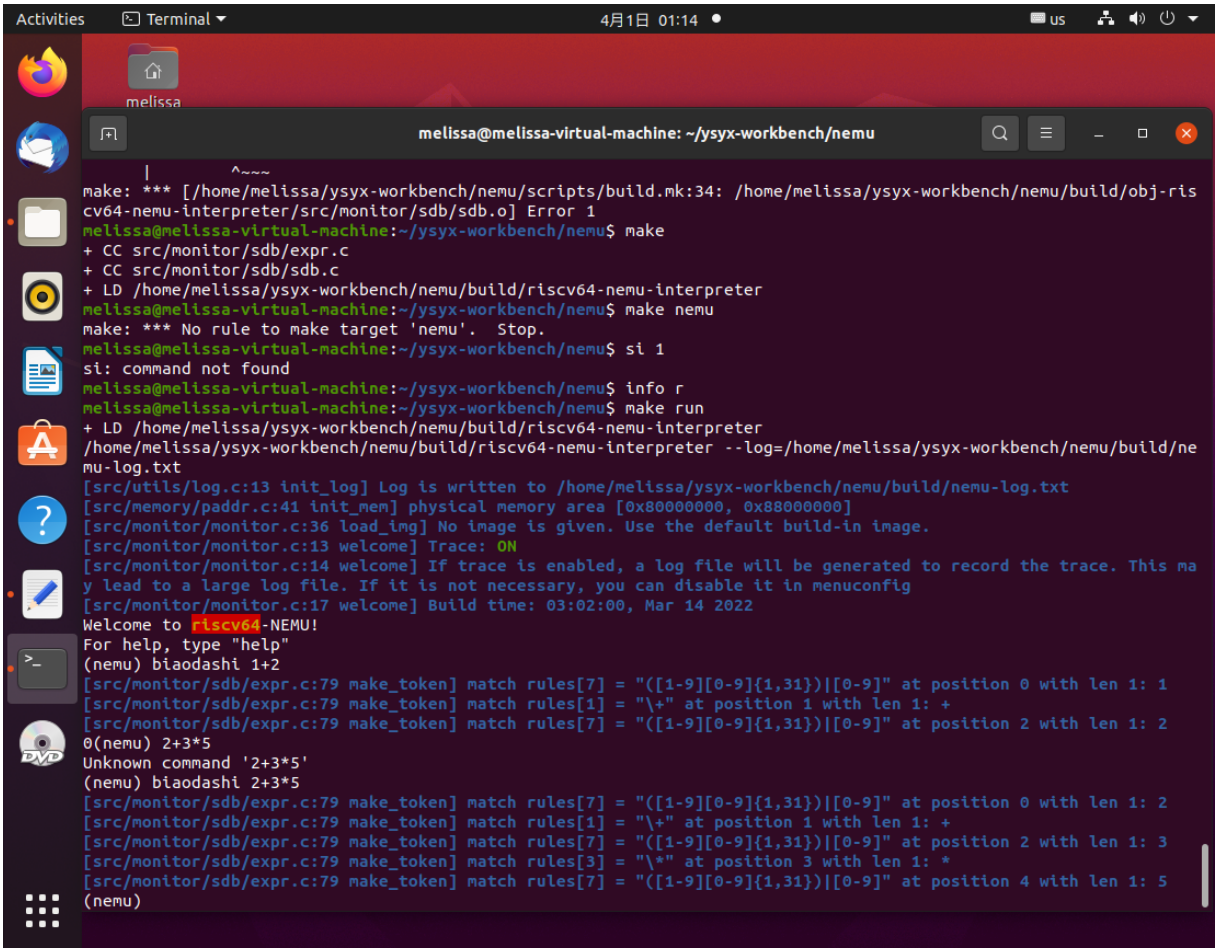
②compiled 是已经用 regcomp 函数编译好的正则表达式，这个值可以为 NULL。

③buffer 指向用来存放错误信息的字符串的内存空间。

④length 指明 buffer 的长度，如果这个错误信息的长度大于这个值，则 regerror 函数会自动截断超出的字符串，但他仍然会返回完整的字符串的长度。所以我们可以用如下的方法先得到错误字符串的长度。

size_t length = regerror (errcode, compiled, NULL, 0);

```
24
25
26
27
28 word_t expr(char *e, bool *success) {
29   if (!make_token(e)) {
30     *success = false;
31     return 0;
32   }
33   else{
34
35   return *(tokens[nr_token].str);
36   }
37   }
```

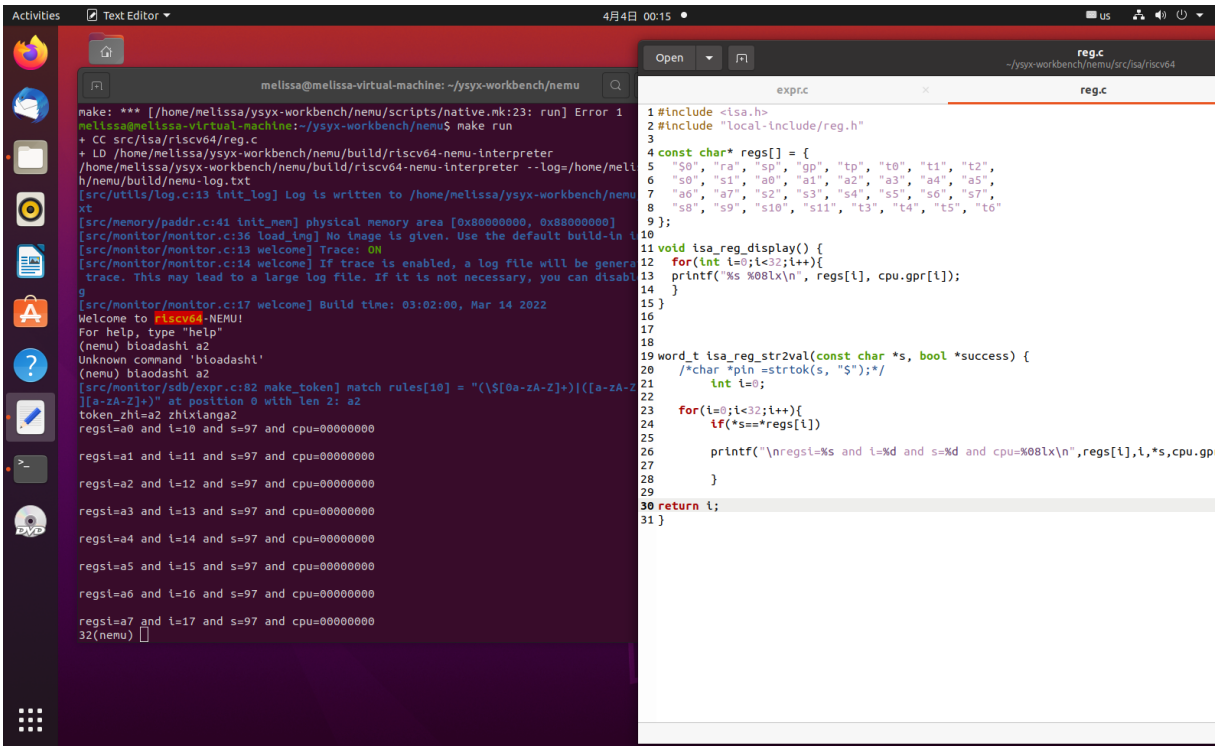


实现打印解析出的中缀表达式

下面通过语义递归树对表达式进行求解

加入了指针参数，指向表达式 token 和 token+nr_oken-1

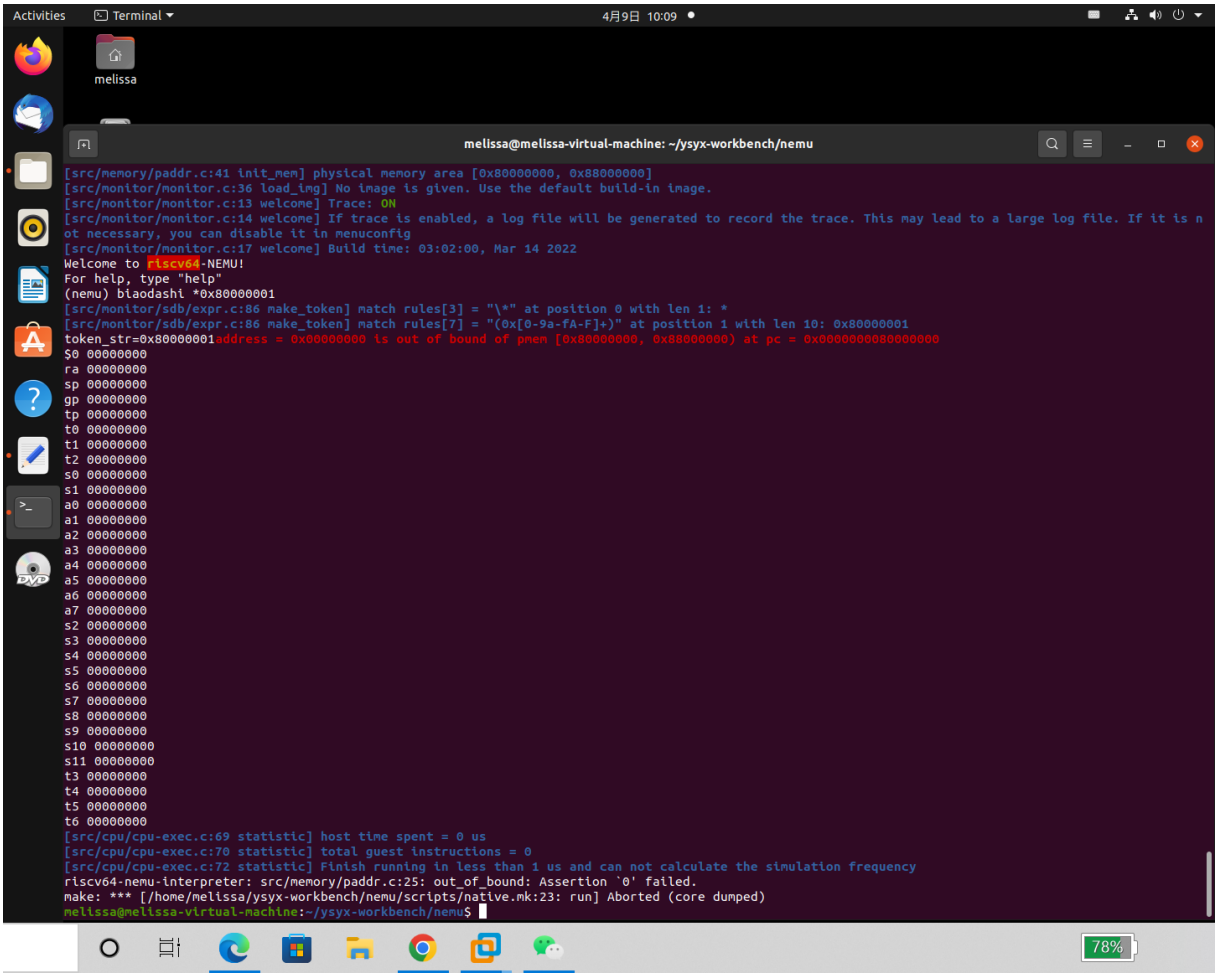
```
[src/monitor/monitor.c:36 load_img] No image is given. Use the default build-in
 image.
[src/monitor/monitor.c:13 welcome] Trace: ON
[src/monitor/monitor.c:14 welcome] If trace is enabled, a log file will be gene
rated to record the trace. This may lead to a large log file. If it is not nece
ssary, you can disable it in menuconfig
[src/monitor/monitor.c:17 welcome] Build time: 03:02:00, Mar 14 2022
Welcome to riscv64-NEMU!
For help, type "help"
(nemu) biaodashi 32*(3-2)+3
[src/monitor/sdb/expr.c:79 make_token] match rules[7] = "([1-9][0-9]{1,31})|[0-
9]" at position 0 with len 2: 32
[src/monitor/sdb/expr.c:79 make_token] match rules[3] = "\*" at position 2 with
 len 1: *
[src/monitor/sdb/expr.c:79 make_token] match rules[5] = "\(" at position 3 with
 len 1: (
[src/monitor/sdb/expr.c:79 make_token] match rules[7] = "([1-9][0-9]{1,31})|[0-
9]" at position 4 with len 1: 3
[src/monitor/sdb/expr.c:79 make_token] match rules[2] = "-" at position 5 with
len 1: -
[src/monitor/sdb/expr.c:79 make_token] match rules[7] = "([1-9][0-9]{1,31})|[0-
9]" at position 6 with len 1: 2
[src/monitor/sdb/expr.c:79 make_token] match rules[6] = "\)" at position 7 with
 len 1: )
[src/monitor/sdb/expr.c:79 make_token] match rules[1] = "\+" at position 8 with
 len 1: +
[src/monitor/sdb/expr.c:79 make_token] match rules[7] = "([1-9][0-9]{1,31})|[0-
9]" at position 9 with len 1: 3
35(nemu)
```



为什么不同的 i 在*regs[i]之后会有相同的值？a0 和 a1 对应字符串的首地址一样？

指针解引用：使用了 paddr_read()，参数为(int)atoi(p->str),4

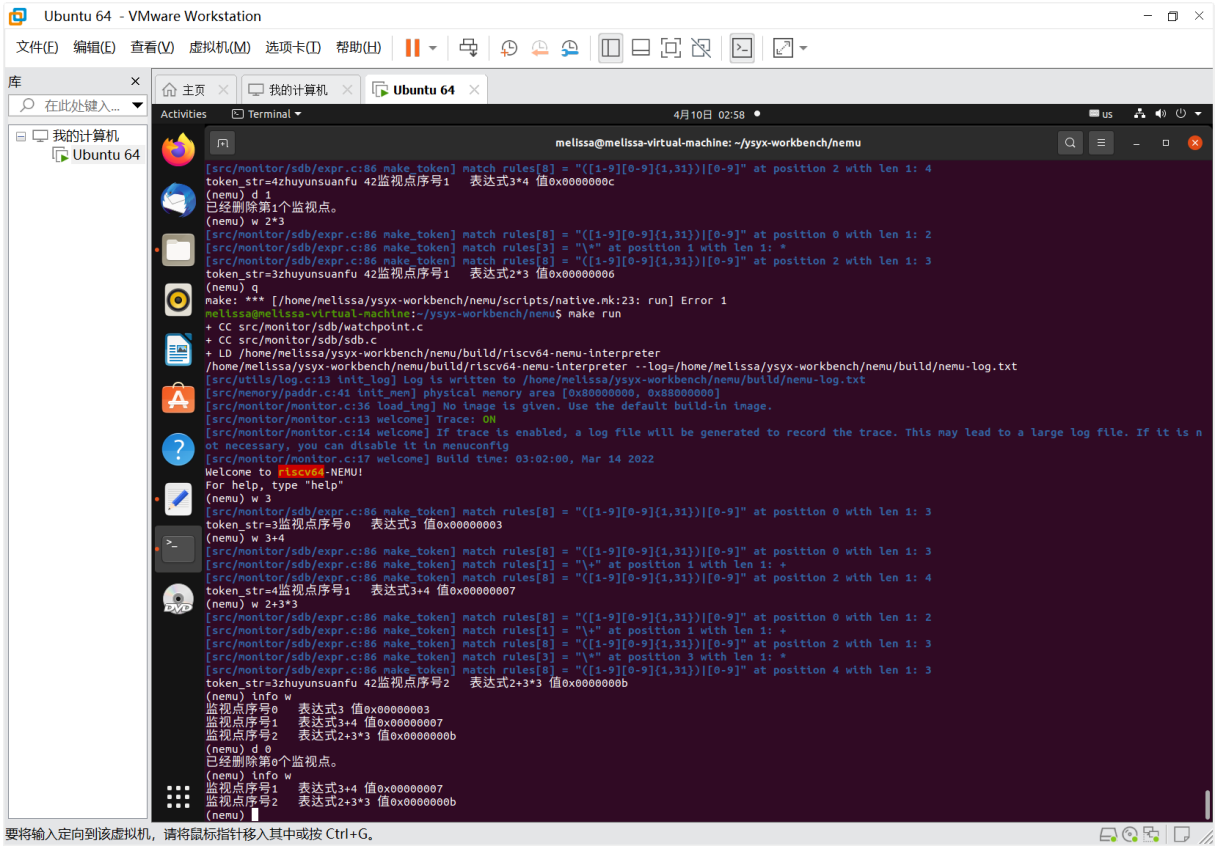
显示：



监视点

csdn 上关于监视点的文章，讲的挺好

https://blog.csdn.net/qq_45655405/article/details/108941025



实现添加，打印，删除监视点功能

对于 cpu_exec()中监测监视点值发生变化程序暂停的功能还没有实现，主要是`trace_and_difftest()`中的内容还没有看懂，下面 mian_loop 中 case stop 会发生什么，怎么暂停运行。

将TAB替换为空格

```
:set expandtab //将tab扩展成空格
:%retab! //按照将tab扩展成空格的格式重新设置当前文件的tab
```

将空格替换为TAB

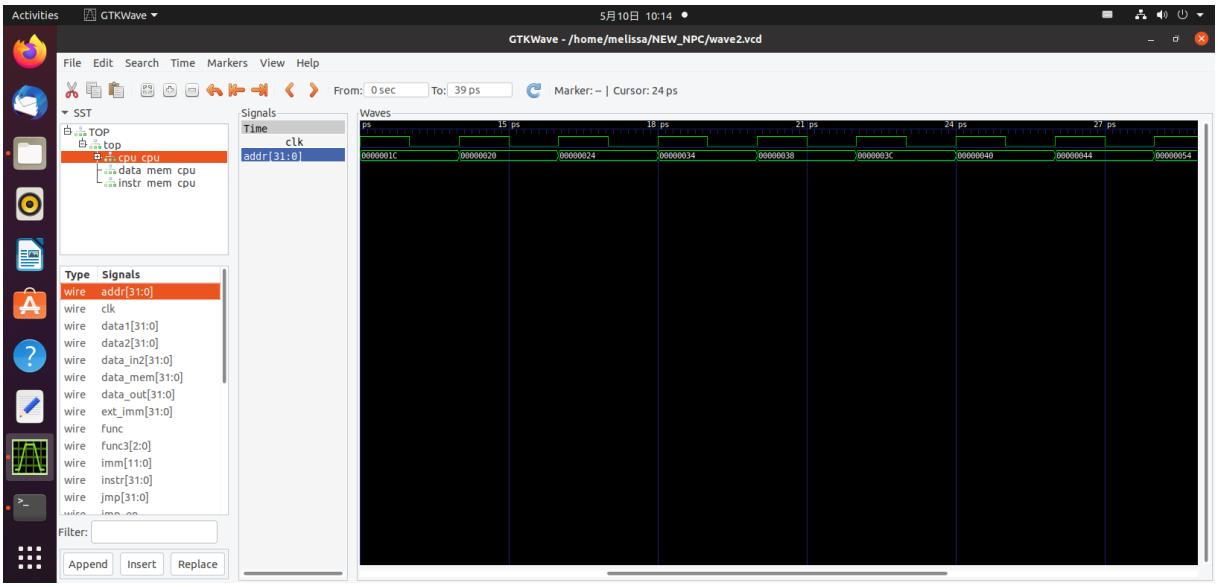
```
:set noexpandtab //将tab不扩展成空格
:%retab! //按照将tab不扩展成空格的格式重新设置当前文件中的tab
```

发现表达式求值 coredump

原因：在 sdb.c 中 expr 中的 bool 类型的指针 success 出问题，因为 sdb.c 中的 bool*一开始是全局变量，而在 watchpiont 函数中需要用到*success 的返回值，所以 core dumped 了

问题：现在不用 expr 中的*success 表达式求值和监视点都可以实现，一用 success 就 coredumped

RTL 单周期处理器

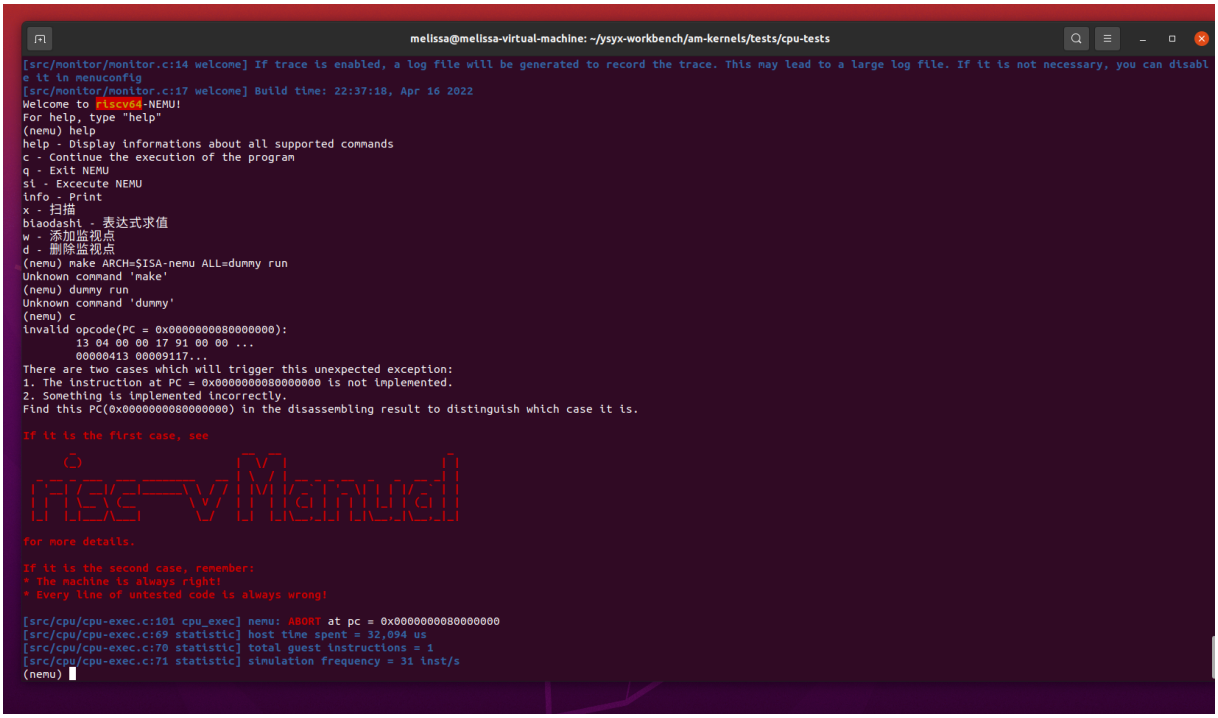


makefile

```
build:
    verilator -Wno-lint --trace --cc top.v --exe main.cpp

comp:
    make -C obj_dir -j -f Vtop.mk Vtop

run:
    obj_dir/Vtop
```

PA2

运行 dummy



运行 string

添加库函数:memcpy,strcmp

```
int strcmp(const char* str1, const char* str2)
{
    while ((*str1) && (*str1 == *str2))
    {
        str1++;
        str2++;
    }


    if (*(unsigned char*)str1 > *(unsigned char*)str2)
    {
        return 1;
    }
    else if (*(unsigned char*)str1 < *(unsigned char*)str2)
    {
        return -1;
    }
    else
    {
        return 0;
    }
}
```

```
int strcmp(const char* str1, const char* str2)
{
    int ret = 0;
    while(!(ret=*(unsigned char*)str1-*(unsigned char*)str2) && *str1)
    {
        str1++;
        str2++
    }


    if (ret < 0)
    {
        return -1;
    }
    else if (ret > 0)
    {
        return 1;
    }
    return 0;
}
```

使用*(unsigned char*)str1 而不是用*str1。这是因为传入的参数为有符号数，有符号字符值的范围是-128~127，无符号字符值的范围是 0~255，而字符串的 ASCII 没有负值，若不转化为无符号数这回在减法实现时出现错误。

例如 str1 的值为 1，str2 的值为 255。

作为无符号数计算时 ret = -254,结果为负值，正确

作为有符号数计算时 ret = 2,结果为正值，错误

memcpy

https://blog.csdn.net/z702143700/article/details/47112971

memcmp

strcpy

https://zhuanlan.zhihu.com/p/45136157

strcat

https://www.programminghunter.com/article/8923797010/

sprintf（必须要看）

嵌入式操作系统---打印函数（printfsprintf）的实现 - 百度文库 (baidu.com)

看下讲义中的手册，还不理解 stdarg.h

http://blog.chinaunix.net/uid-29073321-id-5557641.html

运行 hello-str

需要添加 stdio.c 中的 vsprintf 等

差分测试 difftest

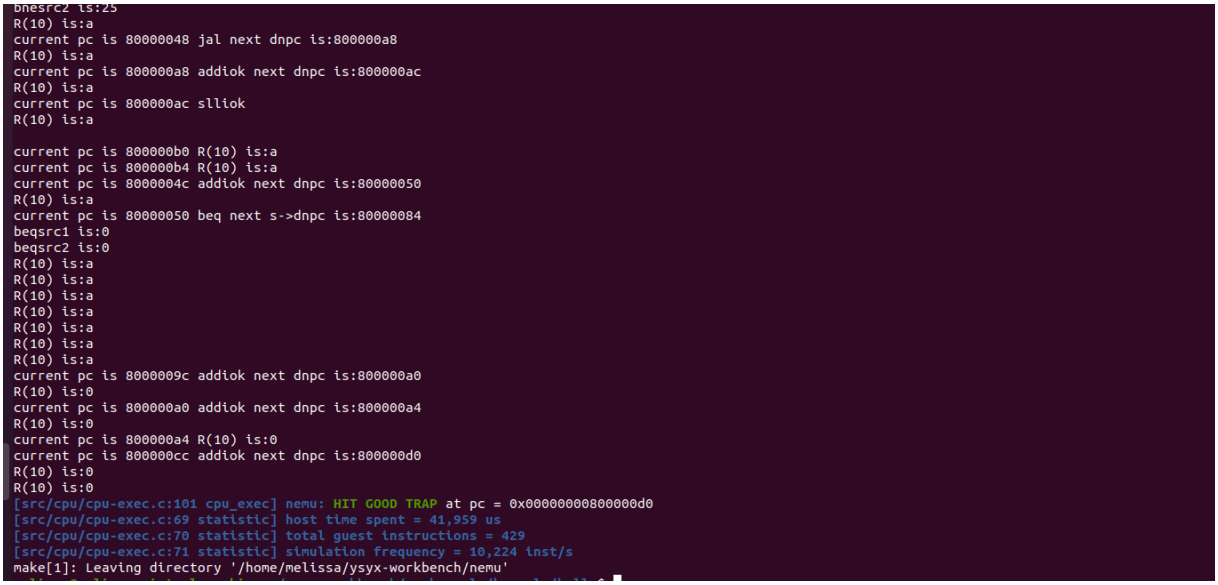
```
 sum if-else add shuixianhua to-lower-case prime dummy div max mov-c
tr bubble-sort shift recursion mul-longlong add-longlong sub-longlon
[           sum] PASS!
[       if-else] PASS!
[           add] PASS!
[   shuixianhua] PASS!
[ to-lower-case] PASS!
[         prime] PASS!
[         dummy] PASS!
[           div] PASS!
[           max] PASS!
[         mov-c] PASS!
[           bit] PASS!
[        pascal] PASS!
[        string] PASS!
[           fib] PASS!
[      goldbach] PASS!
[   select-sort] PASS!
[       unalign] PASS!
[    matrix-mul] PASS!
[         movsx] PASS!
[     leap-year] PASS!
[          fact] PASS!
[        wanshu] PASS!
[     hello-str] PASS!
[   bubble-sort] PASS!
[         shift] PASS!
[     recursion] PASS!
[  mul-longlong] PASS!
[  add-longlong] PASS!
[  sub-longlong] PASS!
[        switch] PASS!
[    quick-sort] PASS!
[    load-store] PASS!
[          min3] PASS!
```

学习 makefile：(2 条消息) makefile 从入门到放弃——博主吐血整理的笔记_万里羊的博客-CSDN 博客

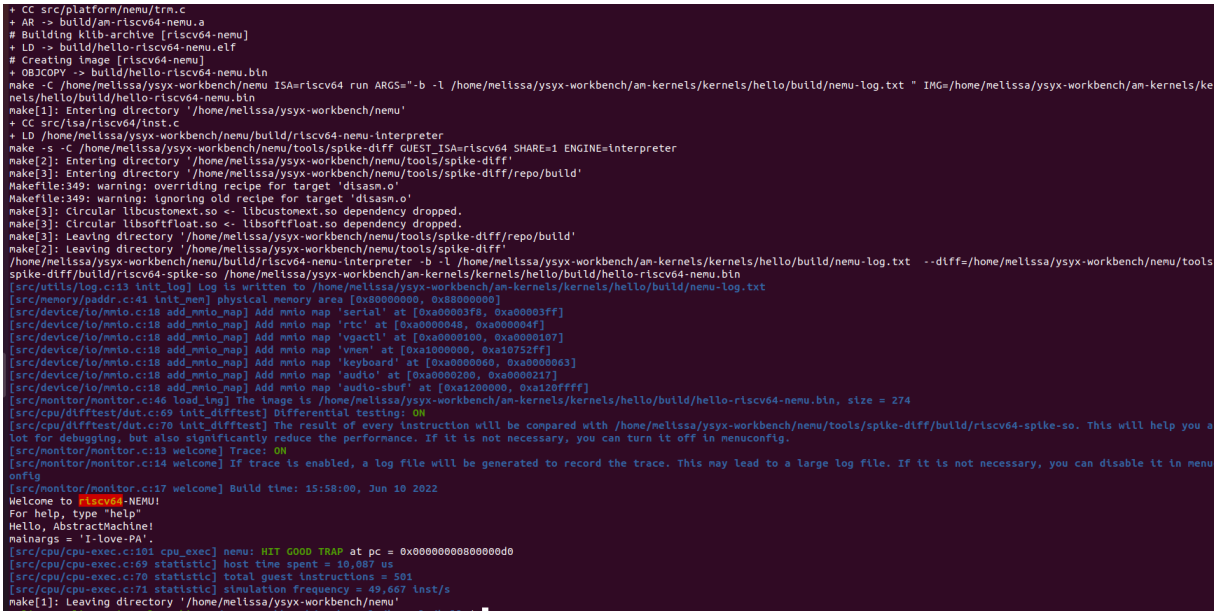
PA2.3

串口

不需要任何修改直接运行 isa 为 riscv64

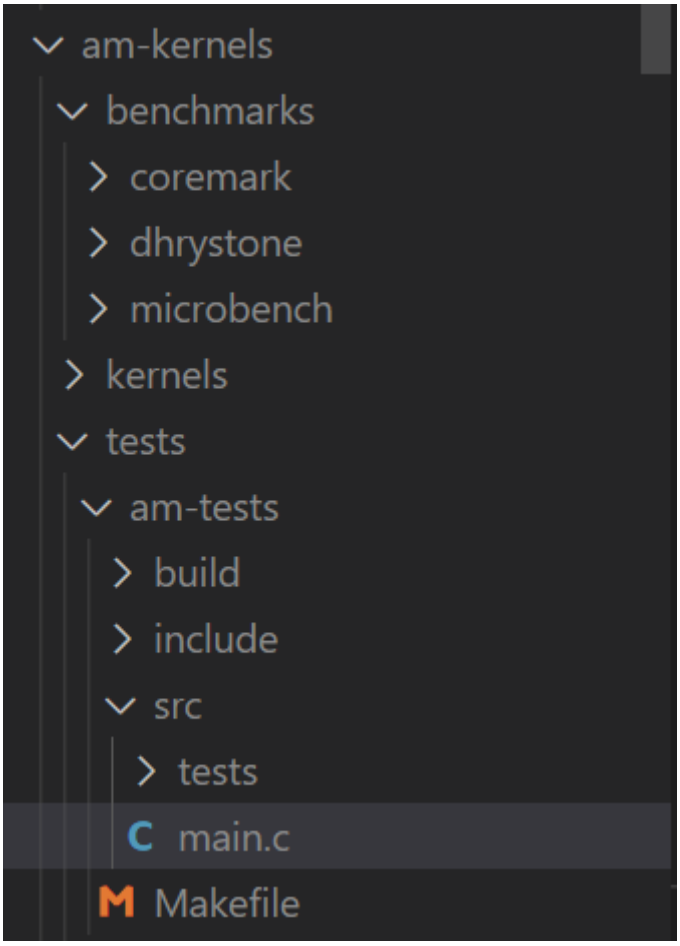


删除调试语句，显示输入语句



时钟

寻找运行 rtc 的命令




```c
#include <amtest.h>

void (*entry)() = NULL; // mp entry

static const char *tests[256] = {
  ['h'] = "hello",
  ['H'] = "display this help message",
  ['i'] = "interrupt/yield test",
  ['d'] = "scan devices",
  ['m'] = "multiprocessor test",
  ['t'] = "real-time clock test",
  ['k'] = "readkey test",
  ['v'] = "display test",
  ['a'] = "audio test",
  ['p'] = "x86 virtual memory test",
};
```

```
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/am-kernels/tests/am-tests/build/nemu-log.txt
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'serial' at [0xa00003f8, 0xa00003ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'rtc' at [0xa0000048, 0xa000004f]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/am-kernels/tests/am-tests/build/amtest-riscv64-
[src/cpu/difftest/dut.c:69 init_difftest] Differential testing: ON
[src/cpu/difftest/dut.c:70 init_difftest] The result of every instruction will be compared with /home/melissa/ysyx-workbench/
you a lot for debugging, but also significantly reduce the performance. If it is not necessary, you can turn it off in menuco
[src/monitor/monitor.c:13 welcome] Trace: ON
[src/monitor/monitor.c:14 welcome] If trace is enabled, a log file will be generated to record the trace. This may lead to a
 menuconfig
[src/monitor/monitor.c:17 welcome] Build time: 15:58:00, Jun 10 2022
Welcome to riscv64-NEMU!
For help, type "help"
1900-0-0 00:00:00 GMT (1 second).
1900-0-0 00:00:00 GMT (2 seconds).
1900-0-0 00:00:00 GMT (3 seconds).
1900-0-0 00:00:00 GMT (4 seconds).
1900-0-0 00:00:00 GMT (5 seconds).
```

printf 函数里面之前把 putch 注释掉了，加上 putch 就可以正常输出。

测试跑分

native 、coremarks

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/benchmarks/coremark$ make ARCH=native run
# Building coremark-run [native]
# Building am-archive [native]
# Building klib-archive [native]
# Creating image [native]
+ LD -> build/coremark-native
/home/melissa/ysyx-workbench/am-kernels/benchmarks/coremark/build/coremark-native
Running CoreMark for 1000 iterations
2K performance run parameters for coremark.
CoreMark Size    : 666
Total time (ms)  : 27
Iterations       : 1000
Compiler version : GCC9.4.0
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xd340
Finised in 27 ms.
==================================================
CoreMark PASS       108200 Marks
               vs. 100000 Marks (i7-7700K @ 4.20GHz)
Exit code = 00h
```

native、dhrystone

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/benchmarks/dhrystone$ make ARCH=native run
# Building dhrystone-run [native]
# Building am-archive [native]
# Building klib-archive [native]
+ CC src/stdio.c
+ AR -> build/klib-native.a
# Creating image [native]
+ LD -> build/dhrystone-native
/home/melissa/ysyx-workbench/am-kernels/benchmarks/dhrystone/build/dhrystone-native
Dhrystone Benchmark, Version C, Version 2.2
Trying 500000 runs through Dhrystone.
Finished in 10 ms
==================================================
Dhrystone PASS         88090 Marks
                  vs. 100000 Marks (i7-7700K @ 4.20GHz)
Exit code = 00h
```

native、microbench

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/benchmarks/microbench$ make ARCH=native run
# Building microbench-run [native]
# Building am-archive [native]
# Building klib-archive [native]
# Creating image [native]
+ LD -> build/microbench-native
/home/melissa/ysyx-workbench/am-kernels/benchmarks/microbench/build/microbench-native
Empty mainargs. Use "ref" by default
======= Running MicroBench [input *ref*] =======
[qsort] Quick sort: * Passed.
  min time: 5.553 ms [79308]
[queen] Queen placement: * Passed.
  min time: 5.029 ms [80910]
[bf] Brainf**k interpreter: * Passed.
  min time: 28.802 ms [58381]
[fib] Fibonacci number: * Passed.
  min time: 21.758 ms [92692]
[sieve] Eratosthenes sieve: * Passed.
  min time: 32.643 ms [106678]
[15pz] A* 15-puzzle search: * Passed.
  min time: 6.058 ms [88478]
[dinic] Dinic's maxflow algorithm: * Passed.
  min time: 7.607 ms [107558]
[lzip] Lzip compression: * Passed.
  min time: 7.265 ms [93530]
[ssort] Suffix sort: * Passed.
  min time: 4.194 ms [95422]
[md5] MD5 digest: * Passed.
  min time: 16.191 ms [93873]
==================================================
MicroBench PASS        89683 Marks
                  vs. 100000 Marks (i9-9900K @ 3.60GHz)
Scored time: 135.100 ms
Total  time: 171.907 ms
Exit code = 00h
```

nemu、coremarks

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/benchmarks/coremark$ make ARCH=riscv64-nemu run
# Building coremark-run [riscv64-nemu]
# Building am-archive [riscv64-nemu]
+ CC src/platform/nemu/trm.c
+ AR -> build/am-riscv64-nemu.a
# Building klib-archive [riscv64-nemu]
+ LD -> build/coremark-riscv64-nemu.elf
# Creating image [riscv64-nemu]
+ OBJCOPY -> build/coremark-riscv64-nemu.bin
make -C /home/melissa/ysyx-workbench/nemu ISA=riscv64 run ARGS="-b -l /home/melissa/ysyx-workbench/am-kernels/benchmarks/coremark/build/nemu-log.txt " IMG=/
u.bin
make[1]: Entering directory '/home/melissa/ysyx-workbench/nemu'
+ LD /home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter
/home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter -b -l /home/melissa/ysyx-workbench/am-kernels/benchmarks/coremark/build/nemu-log.txt   /hom
in
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/am-kernels/benchmarks/coremark/build/nemu-log.txt
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'serial' at [0xa00003f8, 0xa00003ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'rtc' at [0xa0000048, 0xa000004f]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/am-kernels/benchmarks/coremark/build/coremark-riscv64-nemu.bin, size = 18304
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 01:37:47, Jun 13 2022
Welcome to riscv64-NEMU!
For help, type "help"
Running CoreMark for 1000 iterations
2K performance run parameters for coremark.
CoreMark Size    : 666
Total time (ms)  : 19451
Iterations       : 1000
Compiler version : GCC9.4.0
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xd340
Finised in 19451 ms.
==================================================
CoreMark PASS       150 Marks
             vs. 100000 Marks (i7-7700K @ 4.20GHz)
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT GOOD TRAP at pc = 0x00000000800021d0
[src/cpu/cpu-exec.c:69 statistic] host time spent = 19,454,090 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 367,620,575
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 18,896,827 inst/s
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
```

nemu 、dhrystone(在 ubuntu 和 vscode 跑分不一致，vscode 为 76 分)

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/benchmarks/dhrystone$ make ARCH=riscv64-nemu run
# Building dhrystone-run [riscv64-nemu]
# Building am-archive [riscv64-nemu]
+ CC src/platform/nemu/trm.c
+ AR -> build/am-riscv64-nemu.a
# Building klib-archive [riscv64-nemu]
+ LD -> build/dhrystone-riscv64-nemu.elf
# Creating image [riscv64-nemu]
+ OBJCOPY -> build/dhrystone-riscv64-nemu.bin
make -C /home/melissa/ysyx-workbench/nemu ISA=riscv64 run ARGS="-l /home/melissa/ysyx-workbench/am-kernels/benchmarks/dhrystone/build/nemu-log.txt " IMG=/home/me
u.bin
make[1]: Entering directory '/home/melissa/ysyx-workbench/nemu'
+ LD /home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter
/home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter -l /home/melissa/ysyx-workbench/am-kernels/benchmarks/dhrystone/build/nemu-log.txt   /home/melis
in
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/am-kernels/benchmarks/dhrystone/build/nemu-log.txt
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'serial' at [0xa00003f8, 0xa00003ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'rtc' at [0xa0000048, 0xa000004f]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/am-kernels/benchmarks/dhrystone/build/dhrystone-riscv64-nemu.bin, size = 21964
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 01:37:47, Jun 13 2022
Welcome to riscv64-NEMU!
For help, type "help"
(nemu) c
Dhrystone Benchmark, Version C, Version 2.2
current pc is 80000ef8 R(10) is:8000dad7
current pc is 80000f14 R(10) is:8000dad7
current pc is 80000ef8 R(10) is:8000dad7
current pc is 80000f14 R(10) is:8000dad7
current pc is 80000ef8 R(10) is:8000dad7
current pc is 80000f14 R(10) is:8000dad7
current pc is 80000ef8 R(10) is:8000dad7
current pc is 80000f14 R(10) is:8000dad7
current pc is 80000ef8 R(10) is:8000dad7
current pc is 80000f14 R(10) is:8000dad7
current pc is 80000ef8 R(10) is:8000dad7
current pc is 80000f14 R(10) is:8000dad7
Trying 500000 runs through Dhrystone.
current pc is 80000388 R(10) is:8000df60
current pc is 800004f0 R(10) is:1
current pc is 80000ef8 R(10) is:8000dadc
current pc is 80000f14 R(10) is:8000dadc
current pc is 80000ef8 R(10) is:8000dadc
current pc is 80000f14 R(10) is:8000dadc
current pc is 80000ef8 R(10) is:8000dadc
current pc is 80000f14 R(10) is:8000dadc
current pc is 80000ef8 R(10) is:8000dadc
current pc is 80000f14 R(10) is:8000dadc
current pc is 80000ef8 R(10) is:8000dadc
current pc is 80000f14 R(10) is:8000dadc
Finished in 11912 ms
================================================
current pc is 80000ef8 R(10) is:8000dae7
current pc is 80000f14 R(10) is:8000dae7
current pc is 80000ef8 R(10) is:8000dae7
current pc is 80000f14 R(10) is:8000dae7
Dhrystone PASS         73 Marks
                vs. 100000 Marks (i7-7700K @ 4.20GHz)
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT GOOD TRAP at pc = 0x0000000080000bd8
[src/cpu/cpu-exec.c:69 statistic] host time spent = 11,913,168 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 220,006,709
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 18,467,523 inst/s
```

nemu 、microbench

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/benchmarks/microbench$ make ARCH=riscv64-nemu run
# Building microbench-run [riscv64-nemu]
# Building am-archive [riscv64-nemu]
+ CC src/platform/nemu/trm.c
+ AR -> build/am-riscv64-nemu.a
# Building klib-archive [riscv64-nemu]
+ LD -> build/microbench-riscv64-nemu.elf
# Creating image [riscv64-nemu]
+ OBJCOPY -> build/microbench-riscv64-nemu.bin
make -C /home/melissa/ysyx-workbench/nemu ISA=riscv64 run ARGS="-b -l /home/melissa/ysyx-workbench/am-kernels/benchmarks/microbench/build
64-nemu.bin
make[1]: Entering directory '/home/melissa/ysyx-workbench/nemu'
+ LD /home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter
/home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter -b -l /home/melissa/ysyx-workbench/am-kernels/benchmarks/microbench/buil
nemu.bin
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/am-kernels/benchmarks/microbench/build/nemu-log.txt
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'serial' at [0xa00003f8, 0xa00003ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'rtc' at [0xa0000048, 0xa000004f]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/am-kernels/benchmarks/microbench/build/microbench-riscv64-n
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 01:37:47, Jun 13 2022
Welcome to riscv64-NEMU!
For help, type "help"
Empty mainargs. Use "ref" by default
======= Running MicroBench [input *ref*] =======
[qsort] Quick sort: * Passed.
  min time: 1613.379 ms [272]
[queen] Queen placement: * Passed.
  min time: 2041.324 ms [199]
[bf] Brainf**k interpreter: * Passed.
  min time: 12383.184 ms [135]
[fib] Fibonacci number: * Passed.
  min time: 18564.156 ms [108]
[sieve] Eratosthenes sieve: X Failed.
  min time: 0.018 ms [0]
[15pz] A* 15-puzzle search: * Passed.
  min time: 2085.609 ms [256]
[dinic] Dinic's maxflow algorithm: * Passed.
  min time: 3026.325 ms [270]
[lzip] Lzip compression: * Passed.
  min time: 3242.910 ms [209]
[ssort] Suffix sort: * Passed.
  min time: 1075.892 ms [371]
[md5] MD5 digest: X Failed.
  min time: 16770.101 ms [0]
==================================================
MicroBench FAIL        182 Marks
                vs. 100000 Marks (i9-9900K @ 3.60GHz)
Scored time: 60802.898 ms
Total  time: 73536.070 ms
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT GOOD TRAP at pc = 0x0000000080004614
[src/cpu/cpu-exec.c:69 statistic] host time spent = 73,537,972 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 1,527,979,658
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 20,778,104 inst/s
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
```

键盘

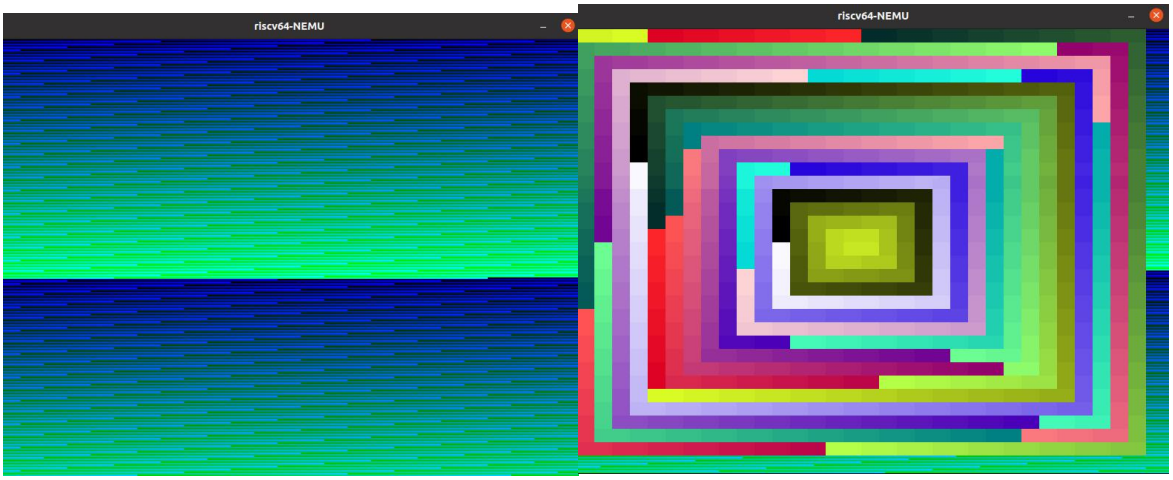
观看 nemu 中的 send_key 函数：

```
  uint32_t am_scancode = keymap[scancode] | (is_keydown ? KEYDOWN_MASK : 0);
```

可以发现如果有按键按下时，32 位键码的第 16 位一定为 1，所以通过取回的 32 位键码的第 16 位来判断按键是否按下。

```
melissa@melissa-virtual-machine:~/ysyx-workbench/am-kernels/tests/am-tests$ make ARCH=riscv64-nemu mainargs=k run
# Building amtest-run [riscv64-nemu]
# Building am-archive [riscv64-nemu]
+ CC src/platform/nemu/trm.c
+ AR -> build/am-riscv64-nemu.a
# Building klib-archive [riscv64-nemu]
+ LD -> build/amtest-riscv64-nemu.elf
# Creating image [riscv64-nemu]
+ OBJCOPY -> build/amtest-riscv64-nemu.bin
make -C /home/melissa/ysyx-workbench/nemu ISA=riscv64 run ARGS="-b -l /home/melissa/ysyx-workbench/am-kernels/tests/am-tests/build/nemu-log.txt " IMG=/home/melissa/y
make[1]: Entering directory '/home/melissa/ysyx-workbench/nemu'
+ LD /home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter
/home/melissa/ysyx-workbench/nemu/build/riscv64-nemu-interpreter -b -l /home/melissa/ysyx-workbench/am-kernels/tests/am-tests/build/nemu-log.txt   /home/melissa/ysyx
[src/utils/log.c:13 init_log] Log is written to /home/melissa/ysyx-workbench/am-kernels/tests/am-tests/build/nemu-log.txt
[src/memory/paddr.c:41 init_mem] physical memory area [0x80000000, 0x88000000]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'serial' at [0xa00003f8, 0xa00003ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'rtc' at [0xa0000048, 0xa000004f]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/am-kernels/tests/am-tests/build/amtest-riscv64-nemu.bin, size = 615108
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 01:37:47, Jun 13 2022
Welcome to riscv64-NEMU!
For help, type "help"
Try to press any key (uart or keyboard)...
Got  (kbd): F3 (4) DOWN
Got  (kbd): F3 (4) UP
Got  (kbd): F4 (5) DOWN
Got  (kbd): F4 (5) UP
Got  (kbd): 3 (17) DOWN
Got  (kbd): 3 (17) UP
Got  (kbd): A (43) DOWN
Got  (kbd): A (43) UP
Got  (kbd): O (37) DOWN
Got  (kbd): K (50) DOWN
Got  (kbd): O (37) UP
Got  (kbd): K (50) UP
Got  (kbd): RALT (71) DOWN
Got  (kbd): A (43) DOWN
Got  (kbd): RALT (71) UP
Got  (kbd): A (43) UP
Got  (kbd): LCTRL (67) DOWN
Got  (kbd): Z (56) DOWN
Got  (kbd): LCTRL (67) UP
Got  (kbd): Z (56) UP
Got  (kbd): Q (29) DOWN
Got  (kbd): Q (29) UP
[src/cpu/cpu-exec.c:69 statistic] host time spent = 31,766,004 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 842,659,832
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 26,527,095 inst/s
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
```

屏幕



打字小游戏还不太好使，不显示字母

PA3

深刻的认识到了这几条 git 指令

git merge 指令

```
melissa@melissa-virtual-machine:~/ysyx-workbench$ git checkout master
Switched to branch 'master'
melissa@melissa-virtual-machine:~/ysyx-workbench$ cd ysyx-workbench
bash: cd: ysyx-workbench: No such file or directory
melissa@melissa-virtual-machine:~/ysyx-workbench$ git branch
  main
  master
  pa1
* pa3
```

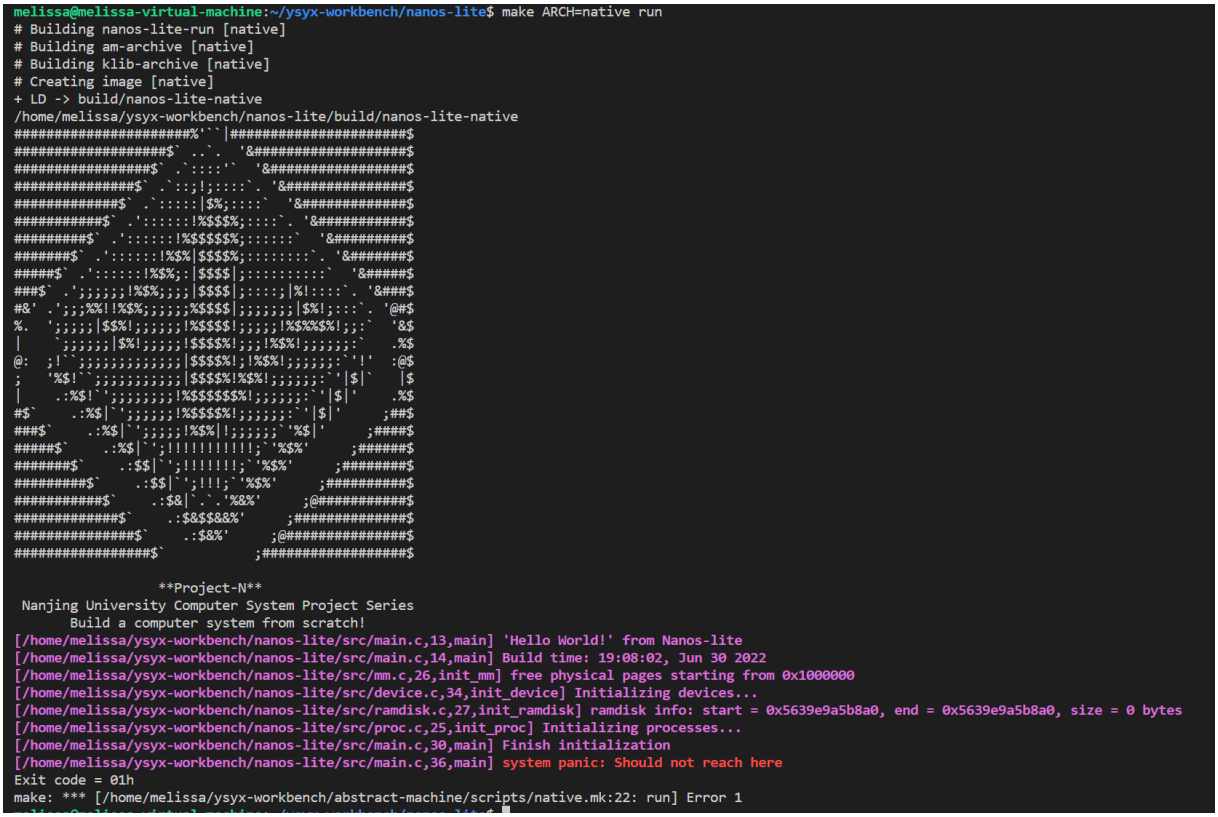
https://segmentfault.com/a/1190000023734704、

又出问题了删掉 pa3 分支，重新修好程序，先搞 npc，再接着搞 pa3 吧

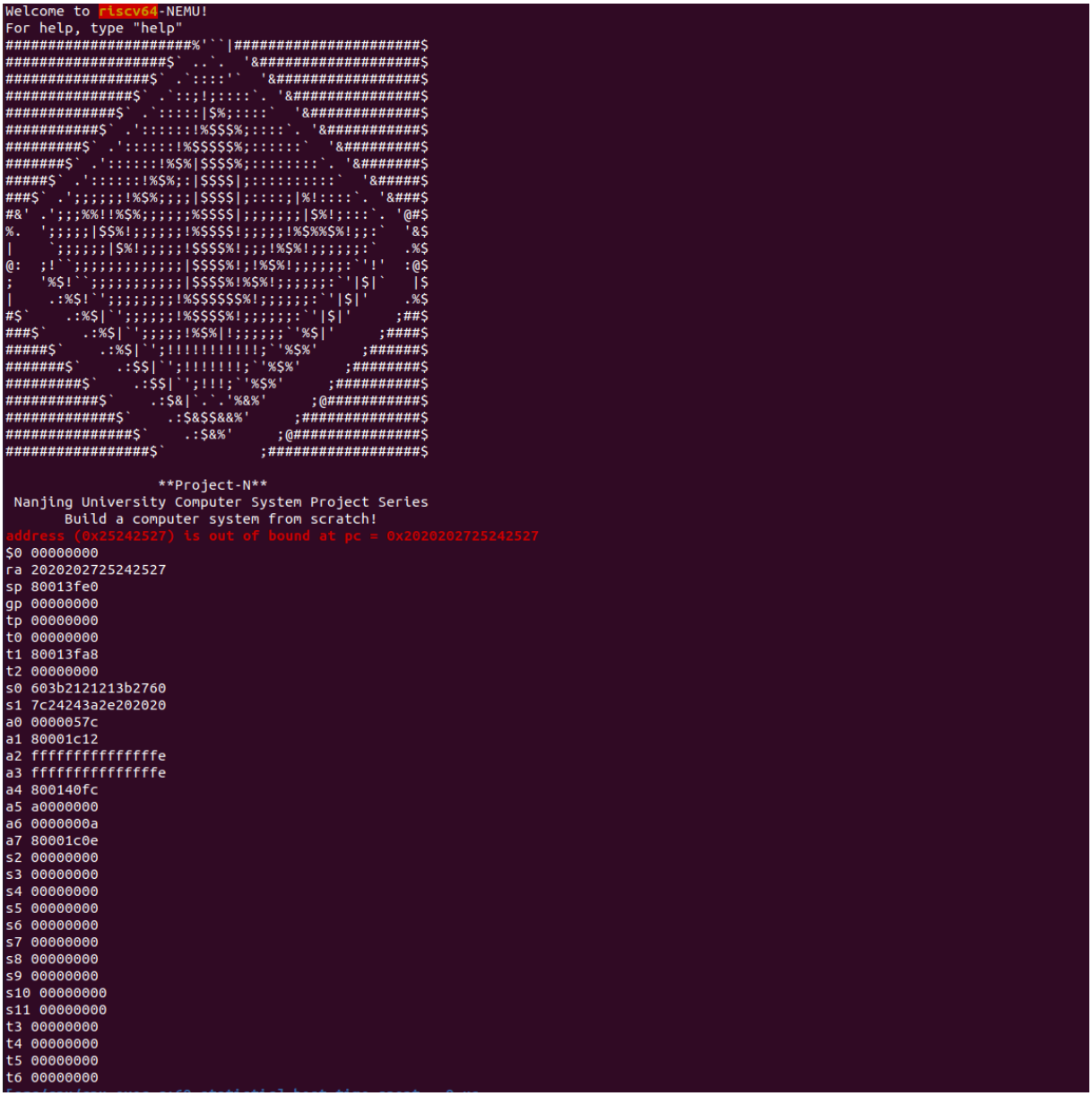
请把 `ysyx-workbench` 作为PA讲义中的项目目录, 即将PA讲义中的 `ics2022` 看成是 `ysyx-workbench`

批处理系统

native 直接运行



nemu 运行：（需要打开 Device）



两个模拟器都触发异常操作，需要进行一系列操作完成异常响应机制

穿越时空的旅程

自陷操作

设置异常入口地址：根据讲义地址在 CTE 中定义 HAS_CTE，Nanos-lite 会调用 cte_init()进而：

设置异常入口地址，异常入口地址设置到 mtvec 寄存器中即可.

```
bool cte_init(Context*(*handler)(Event, Context*)) {
  // initialize exception entry
  asm volatile("csrw mtvec, %0" : : "r"(__am_asm_trap));

  // register event handler
  user_handler = handler;

  return true;
}
```

```
struct Context {
  // TODO: fix the order of these members to match trap.S
  uintptr_t mepc, mcause, gpr[32], mstatus;
  void *pdir;
};

#define GPR1 gpr[17] // a7
```

```
elcome to riscv64-NEMU!
or help, type "help"
/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 01:11:55, Jul  2 2022
/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 0000000080014000
/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002079, end = 0000000080002079, size = 0 bytes
/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,12,init_irq] Initializing interrupt/exception handler...
/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,25,init_proc] Initializing processes...
/home/melissa/ysyx-workbench/nanos-lite/src/main.c,29,main] Finish initialization
ump outcurrent pc is 80000574 R(10) is:5e
/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,5,do_event] system panic: Unhandled event ID = 4
src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT BAD TRAP at pc = 0x0000000080000224
src/cpu/cpu-exec.c:69 statistic] host time spent = 5,429 us
src/cpu/cpu-exec.c:70 statistic] total guest instructions = 21,899
src/cpu/cpu-exec.c:71 statistic] simulation frequency = 4,033,707 inst/s
ake[1]: *** [/home/melissa/ysyx-workbench/nemu/scripts/native.mk:23: run] Error 1
ake[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
ake: *** [/home/melissa/ysyx-workbench/abstract-machine/scripts/platform/nemu.mk:27: run] Error 2
elissa@melissa-virtual-machine:~/ysyx-workbench/nanos-lite$
```

发生错误事件，在使用指令时，根据 txt 对出了控制寄存器的顺序也通过了 diff，但是发生错误，可能是因为 R(10)不为 0 的原因

```
INSTPAT("??????? ????? ????? 010 ????? 11100 11", csrrs,   I, R(dest) = cpu.sr[BITS(src2,3,0)], cpu.sr[BITS(src2,3,0)] = cpu.sr[BITS(src2,3,0)] | src1);
INSTPAT("??????? ????? ????? 001 ????? 11100 11", csrrw,   I, R(dest) = cpu.sr[BITS(src2,3,0)], cpu.sr[BITS(src2,3,0)] = src1);
INSTPAT("0000000 00000 00000 000 00000 11100 11", ecall , I, s->dnpc = isa_raise_intr(11,s->pc),printf("current pc is %lx ",s->pc),printf("R(10) is:%lx\n",
```

保存上下文

需要 include arch/riscv64-nemu.h

```c
struct Context {
  // TODO: fix the order of these members to match trap.S
  //uintptr_t mepc, mcause, gpr[32], mstatus;
  //void *pdir;
  uintptr_t gpr[32];
  uintptr_t mcause;
  uintptr_t mstatus;
  uintptr_t mepc;
  void *pdir;
};
```

```
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,12,init_irq] Initializing interrupt/exception hand
csrrw cpu.sr[src2] = 800005e0
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,25,init_proc] Initializing processes...
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,29,main] Finish initialization
jump outecall current pc is 800005d4
R(10) is:5e
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 800005d4
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 800005d4
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,5,do_event] system panic: Unhandled event ID = 4
```

> **异常号的保存**
>
> x86通过软件来保存异常号, 没有类似cause的寄存器. mips32和riscv32也可以这样吗? 为什么?

> **诡异的x86代码**
>
> x86的 `trap.S` 中有一行 `pushl %esp` 的代码, 乍看之下其行为十分诡异. 你能结合前后的代码理解它的行为吗? Hint: 程序是个状态机.

在网上查与栈的操作有关，将硬件中的值存入栈？

事件分发

```
Welcome to riscv64-NEMU!
For help, type "help"
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 22:56:05, Jul  2 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 0000000080014000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002131, end = 0000000080002131, size = 0 bytes
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[src2] = 80000608
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,25,init_proc] Initializing processes...
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,29,main] Finish initialization
jump outecall current pc is 80000600
R(10) is:5e
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 80000600
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 80000600
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,5,do_event] system panic: Unhandled event ID = 1
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT BAD TRAP at pc = 0x000000008000023c
[src/cpu/cpu-exec.c:69 statistic] host time spent = 24,526 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 23,386
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 953,518 inst/s
make[1]: *** [/home/melissa/ysyx-workbench/nemu/scripts/native.mk:23: run] Error 1
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
make: *** [/home/melissa/ysyx-workbench/abstract-machine/scripts/platform/nemu.mk:27: run] Error 2
```

加两个 case 语句

abstract-machine > am > src > riscv > nemu > C cte.c > cte_init(Context *(*)(Event,Context *))

```c
Context* __am_irq_handle(Context *c) {
  if (user_handler) {
    Event ev = {0};
    switch (c->mcause) {
      case(11):ev.event = EVENT_YIELD; break;
      default: ev.event = EVENT_ERROR; break;
    }
    printf("mcause = %lx\n",c->mcause);
    printf("mstatus = %lx\n",c->mstatus);
    printf("mepc = %lx\n",c->mepc);
    c = user_handler(ev, c);
    assert(c != NULL);
  }

  return c;
}
```

```c
static Context* do_event(Event e, Context* c) {
  switch (e.event) {
    case(1): panic("Unhandled event ID = %d", e.event);
    default: panic("Unhandled event ID = %d", e.event);
  }

  return c;
}
```

恢复上下文

恢复上下文需要 risc 在软件中给 pc 加 4

```
Context* __am_irq_handle(Context *c) {
  if (user_handler) {
    Event ev = {0};
    switch (c->mcause) {
      case(11):ev.event = EVENT_YIELD;c->mepc= c->mepc + 4; break;
      default: ev.event = EVENT_ERROR; break;
    }
    printf("mcause = %lx\n",c->mcause);
    printf("mstatus = %lx\n",c->mstatus);
    printf("mepc = %lx\n",c->mepc);
    c = user_handler(ev, c);
    assert(c != NULL);
  }
```

然后实现 mret 指令

```
word_t isa_query_intr() {
  //return INTR_EMPTY;
  //自家加实现mret
  return cpu.sr[833];
}
```

```
INSTPAT("0011000 00010 00000 000 00000 11100 11", mret ,    R, s->dnpc = isa_query_intr() );
```

```
Welcome to riscv64-NEMU!
For help, type "help"
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 22:56:05, Jul  2 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 0000000080014000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002171, end = 0000000080002
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[src2] = 80000630
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,25,init_proc] Initializing processes...
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,29,main] Finish initialization
jump outecall current pc is 80000628
R(10) is:5e
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 80000628
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 8000062c
do evevt event ID = 1
csrrw cpu.sr[src2] = a00001800
csrrw cpu.sr[src2] = 8000062c
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,35,main] system panic: Should not reach here
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT BAD TRAP at pc = 0x0000000080000258
[src/cpu/cpu-exec.c:69 statistic] host time spent = 21,368 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 23,794
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 1,113,534 inst/s
make[1]: *** [/home/melissa/ysyx-workbench/nemu/scripts/native.mk:23: run] Error 1
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
make: *** [/home/melissa/ysyx-workbench/abstract-machine/scripts/platform/nemu.mk:27: run] Error 2
```

需要补程序流程

加载用户程序和系统调用

加载第一个用户程序

如果不 load

```
[src/cpu/difftest/dut.c:70 init_difftest] The result of every instruction will be compared with /home/melissa/ysyx-workbench/nemu/tools/spike-diff/build/riscv64-spike-so. T
 help you a lot for debugging, but also significantly reduce the performance. If it is not necessary, you can turn it off in menuconfig.
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 16:26:23, Jul  2 2022
Welcome to riscv64-NEMU!
For help, type "help"
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 00:53:12, Jul  7 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 000000008001b000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002199, end = 0000000080009ab9, size = 31008 bytes
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[src2] = 80000650
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,24,init_proc] Initializing processes...
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,29,main] Finish initialization
jump outecall current pc is 80000648
R(10) is:5e
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 80000648
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 8000064c
do evevt event ID = 1
okokcsrrw cpu.sr[src2] = a00001800
csrrw cpu.sr[src2] = 8000064c
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,35,main] system panic: Should not reach here
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT BAD TRAP at pc = 0x0000000080000258
[src/cpu/cpu-exec.c:69 statistic] host time spent = 6,189 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 24,112
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 3,895,944 inst/s
make[1]: *** [/home/melissa/ysyx-workbench/nemu/scripts/native.mk:23: run] Error 1
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
make: *** [/home/melissa/ysyx-workbench/abstract-machine/scripts/platform/nemu.mk:27: run] Error 2
```

使用 load 之后

```
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 00:53:12, Jul  7 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 000000008001c000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002479, end = 0000000080009d99, size = 31008 bytes
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[src2] = 80000828
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,24,init_proc] Initializing processes...
loderok1
loderok2
eok2
eok4
e_entryad = 0x83000430
[/home/melissa/ysyx-workbench/nanos-lite/src/loader.c,44,naive_uload] Jump to entry = 0000000083000430
jump outecall current pc is 83000110
R(10) is:0
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 83000110
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 83000114
address (0x00004c38) is out of bound at pc = 0x0000000000004c38
$0 00000000
ra 80000788
sp 8001be40
gp 00000000
tp 00000000
t0 00000020
t1 00000007
t2 00000000
s0 8001bea0
s1 80013080
a0 8001be40
a1 8001bea0
```

```
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 13:08:42, Jul  7 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 000000008001c000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002581, end = 0000000080009e99, size = 31000 bytes
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[src2] = 800008e8
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,24,init_proc] Initializing processes...
loderok1
ehdr.e_phnum = 3
for i = 0
eok2
for i = 1
eok2
eok3
eok32
for i = 2
[/home/melissa/ysyx-workbench/nanos-lite/src/loader.c,69,naive_uload] Jump to entry = 0000000083000430
jump outecall current pc is 83000110
R(10) is:0
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 83000110
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 83000114
address (0x00004c30) is out of bound at pc = 0x0000000000004c30
$0 00000000
ra 80000848
```

最后效果无法实现自陷，但是可以触发异常事件，如果在 inst.c 中改变 instr 事件号的话

```
[src/cpu/difftest/dut.c:69 init_difftest] Differential testing: ON
[src/cpu/difftest/dut.c:70 init_difftest] The result of every instruction will be compared with /home/melissa/ysyx-workbench/nemu/tools/spike-diff/build/riscv64-spike-so. This w
 help you a lot for debugging, but also significantly reduce the performance. If it is not necessary, you can turn it off in menuconfig.
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 16:26:23, Jul  2 2022
Welcome to riscv64-NEMU!
For help, type "help"
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 13:08:42, Jul  7 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 000000008001c000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002571, end = 0000000080009e89, size = 31000 bytes
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[src2] = 80000850
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,24,init_proc] Initializing processes...
[/home/melissa/ysyx-workbench/nanos-lite/src/loader.c,36,naive_uload] Jump to entry = 0000000083000430
jump out
ecall current pc is 83000110
R(10) is:0
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 83000110
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 83000114
do evevt event ID = 1
okok
csrrw cpu.sr[src2] = a00001800
csrrw cpu.sr[src2] = 83000114
jump out
ecall current pc is 83000124
R(10) is:0
csrrs cpu.sr[src2] = b
csrrs cpu.sr[src2] = a00001800
csrrs cpu.sr[src2] = 83000124
csrrw cpu.sr[src2] = a00021800
mcause = b
mstatus = a00001800
mepc = 83000128
do evevt event ID = 1
okok
csrrw cpu.sr[src2] = a00001800
csrrw cpu.sr[src2] = 83000128
^Z
[1]+  Stopped                 make ARCH=riscv64-nemu run
```

无论是否打开 diff 都会产生程序卡在 83000128 位置，最后执行的指令是 mret，根据 printf 原来是事件分发出了问题导致运行 loader 的 dummy 程序产生地址溢出的问题，应该将 cte.c 中的函数改成如下形式：

abstract-machine > am > src > riscv > nemu > C cte.c > __am_irq_handle(Context *)

```c
#include <arch/riscv64-nemu.h> //自己加
#include <klib.h>

static Context* (*user_handler)(Event, Context*) = NULL;

Context* __am_irq_handle(Context *c) {
  if (user_handler) {
    Event ev = {0};
    switch (c->mcause) {
      case(11):{
        if((c->gpr[17] >=0) &&(c->gpr[17] <=19)){
          ev.event = EVENT_SYSCALL;break;
        }
        if(c->gpr[17] == -1){
          ev.event = EVENT_YIELD;break;
        }
        else {
          ev.event = EVENT_ERROR;break;
        }
      }
      //case(11):ev.event = EVENT_YIELD;break;
      default: ev.event = EVENT_ERROR;break;
    }
    printf("mcause = %lx\n",c->mcause);
    printf("mstatus = %lx\n",c->mstatus);
    printf("mepc = %lx\n",c->mepc);
    c = user_handler(ev, c);
    printf("ok");
    assert(c != NULL);
    printf("ok\n");
  }

  return c;
}
```

加载用户程序的结果如下：

```
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/nanos-lite/build/nanos-lite-riscv64-nemu.bin, size = 77960
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 16:53:08, Jul  8 2022
Welcome to riscv64-NEMU!
For help, type "help"
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 15:03:19, Jul  8 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 000000008001c000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,34,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080002561, end = 0000000080009e79, size = 31000 byte
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,13,init_irq] Initializing interrupt/exception handler...
csrrw cpu.sr[305] = 800008d0
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,24,init_proc] Initializing processes...
loderok1
ehdr.e_phnum = 3
eok2
eok2
eok3
eok32
[/home/melissa/ysyx-workbench/nanos-lite/src/loader.c,43,naive_uload] Jump to entry = 0000000083000430
jump out
ecall current pc is 83000110
R(10) is:0
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = 0
csrrs cpu.sr[341] = 83000110
csrrw cpu.sr[300] = 20000
mcause = b
mstatus = 0
mepc = 83000110
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,6,do_event] system panic: Unhandled event ID = 2
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT BAD TRAP at pc = 0x00000000800004c0
[src/cpu/cpu-exec.c:69 statistic] host time spent = 20,155 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 144,616
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 7,175,192 inst/s
make[1]: *** [/home/melissa/ysyx-workbench/nemu/scripts/native.mk:23: run] Error 1
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
make: *** [/home/melissa/ysyx-workbench/abstract-machine/scripts/platform/nemu.mk:27: run] Error 2
```

系统调用

实现函数

nanos-lite > src > C syscall.c > do_syscall(Context *)

```c
#include <common.h>
#include "syscall.h"
//自己加
//extern void yield();
//extern void halt();
void do_syscall(Context *c) {
  uintptr_t a[4];
  a[0] = c->GPR1;
  a[1] = c->GPR2;
  a[2] = c->GPR3;
  a[3] = c->GPR4;
  a[4] = c->GPRx;
  switch (a[0]) {
    case 1:{
      yield();a[0] = 0;
      printf("a[0] = 1 ok");break;
      }
    case 0:{
      halt(c->GPR1);break;    //是否指向这个宏存疑，以及每个宏代表的寄存器
    }
    default: panic("Unhandled syscall ID = %d", a[0]);
  }
}

```

调用

```
nanos-lite > src > C irq.c > do_syscall(Context *)
1   #include <common.h>
2   //自己加
3   extern void do_syscall(Context *c);
4   static Context* do_event(Event e, Context* c) {
5     switch (e.event) {
6       case 1: printf("do evevt event ID = %d\n", e.event);c->mepc = c->mepc +4;break;
7       case 2: do_syscall(c);c->mepc = c->mepc +4;printf("do evevt event ID = %d\n", e.event);break;
8       default: panic("Unhandled event ID = %d", e.event);break;
9     }
10
11    return c;
12  }
13
14  void init_irq(void) {
15    Log("Initializing interrupt/exception handler...");
16    cte_init(do_event);
17  }
18
```

结果

```
[/home/melissa/ysyx-workbench/nanos-lite/src/loader.c,43,naive_uload] Jump to entry = 0000000083000430
jump out
ecall current pc is 83000110
R(10) is:0
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 83000110
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 83000110
jump out
ecall current pc is 80000958
R(10) is:8001bea0
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00021800
csrrs cpu.sr[341] = 80000958
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00021800
mepc = 80000958
do evevt event ID = 1
okok
csrrw cpu.sr[300] = a00021800
csrrw cpu.sr[341] = 8000095c
mret 函数ok
do evevt event ID = 2
okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 83000114
mret 函数ok
jump out
ecall current pc is 83000124
R(10) is:0
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 83000124
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 83000124
[src/cpu/cpu-exec.c:101 cpu_exec] nemu: HIT GOOD TRAP at pc = 0x0000000080000554
[src/cpu/cpu-exec.c:69 statistic] host time spent = 19,113 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 147,213
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 7,702,244 inst/s
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
```

标准输出

实现 sys_write 的连接，函数参数的传递，do_syscall()中 write()函数的编写

不同寄存器对应关系如下：

gpr a7 - C->GPR0　传递系统调用参数（即调用什么函数例如：SYS_write）

gpr a0 - C->GPR1　传递函数调用参数 1（即调用函数的参数例如：write（int））

gpr a1 - C->GPR2　传递函数调用参数 2（即调用函数的参数例如：write（int，void*））

gpr a2 - C->GPR3　传递函数调用参数 3（即调用函数的参数例如：write（int，void*，size_t））

gpr a0 - C->GPRx　传递函数返回值

堆区管理

实现 sbrk，获取_end，按照手册内容在用户层补充_sbrk 函数

简易文件系统

补全 fs_open,fs_read,fs_close

fs_offset 是为了取出文件的磁盘偏移量

```
  int fd = fs_open(filename,0,0);
  size_t offset = fs_offset(fd);
  printf("打开文件 %d\n",fd);
  printf("loderok1\n");
  ramdisk_read(&ehdr, offset, sizeof(Elf_Ehdr));
  printf("ehdr.e_phnum = %lx\n",ehdr.e_phnum);
  assert(*(uint32_t *)ehdr.e_ident == 0x464c457f);
  //ramdisk_read(phdr, sizeof(Elf_Ehdr),ehdr.e_phnum);
  for(int i=0; i<ehdr.e_phnum; i++){
    ramdisk_read(&phdr, ehdr.e_phoff+ offset +i*sizeof(phdr),sizeof(Elf64_Phdr));
    if(phdr.p_type == PT_LOAD){
      ramdisk_read((void*)(phdr.p_vaddr), phdr.p_offset+offset, phdr.p_filesz);
      printf("eok2\n");
    }
    if(phdr.p_memsz > phdr.p_filesz){
      printf("eok3\n");
      //uint64_t length = phdr.p_paddr + phdr.p_filesz;
      //ramdisk_read((void*)(length), 0x0, phdr.p_memsz - phdr.p_filesz);
      memset((void*)(phdr.p_vaddr + phdr.p_filesz),offset,phdr.p_memsz - phdr.p_filesz+offset);
      printf("eok32\n");
  }
  }
   return  ehdr.e_entry;
}
```

运行 hello 结果（改文件名在 up_native 的时候）

```
csrrs cpu.sr[341] = 83004bac
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 83004bac
do evevt event ID = 2
gpr return a0 = 1
gpr a0 = 4
[/home/melissa/ysyx-workbench/nanos-lite/src/syscall.c,10,sys_write] sys_write:fd=1,count=47

Hello World from Navy-apps for the 152th time!
gpr x = 0
okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 83004bb0
mret cpu.sr[833]=83004bb0
ecall current pc is 83004bac
jump out
ecall current pc is 83004bac
R(10) is:1
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 83004bac
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 83004bac
do evevt event ID = 2
gpr return a0 = 1
gpr a0 = 4
[/home/melissa/ysyx-workbench/nanos-lite/src/syscall.c,10,sys_write] sys_write:fd=1,count=47
```

完成了 read，lseek，write 函数的添加，终于把 file-test 的 pass 打出来了！！！

中途参考了一下代码，再学习一下，之前写函数 fs_write 中的 open_offset 有一些问题，在写的时候没有考虑 open_offset 偏移量与 disk_offset

```
PASS!!!
gpr x = 0
okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 8300794c
mret cpu.sr[833]=8300794c
ecall current pc is 830079a4
jump out
ecall current pc is 830079a4
R(10) is:0
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 830079a4
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 830079a4
do evevt event ID = 2
gpr return a1 = 0
关闭okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 830079a8
mret cpu.sr[833]=830079a8
ecall current pc is 830079a4
jump out
ecall current pc is 830079a4
R(10) is:1
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 830079a4
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 830079a4
do evevt event ID = 2
gpr return a1 = 1
关闭okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 830079a8
mret cpu.sr[833]=830079a8
ecall current pc is 830079a4
jump out
ecall current pc is 830079a4
R(10) is:2
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 830079a4
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 830079a4
do evevt event ID = 2
gpr return a1 = 2
关闭okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 830079a8
mret cpu.sr[833]=830079a8
```

加了一些函数之后可以调取出键盘“文件”，但是无法打印数值，中间过程的 printf 过多

```
mcause = b
mstatus = a00001800
mepc = 83004d28
do evevt event ID = 2
gpr return a1 = 83008048
okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 83004d2c
mret cpu.sr[833]=83004d2c
ecall current pc is 83004d7c
jump out
ecall current pc is 83004d7c
R(10) is:4
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 83004d7c
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 83004d7c
do evevt event ID = 2
gpr return a1 = 4
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,85,fs_read] fs_read:fd=4,open_offset=0 ,count=1024

okok
csrrw cpu.sr[300] = a00001800
csrrw cpu.sr[341] = 83004d80
mret cpu.sr[833]=83004d80
ecall current pc is 83004d28
jump out
ecall current pc is 83004d28
R(10) is:83008048
csrrs cpu.sr[342] = b
csrrs cpu.sr[300] = a00001800
csrrs cpu.sr[341] = 83004d28
csrrw cpu.sr[300] = a00021800
mcause = b
mstatus = a00001800
mepc = 83004d28
do evevt event ID = 2
gpr return a1 = 83008048
okok
```

把注释删掉之后发现也无法输出打印的字符，甚至运行的 NDL 函数中打印的字符也没有出来，目前还没有找到问题

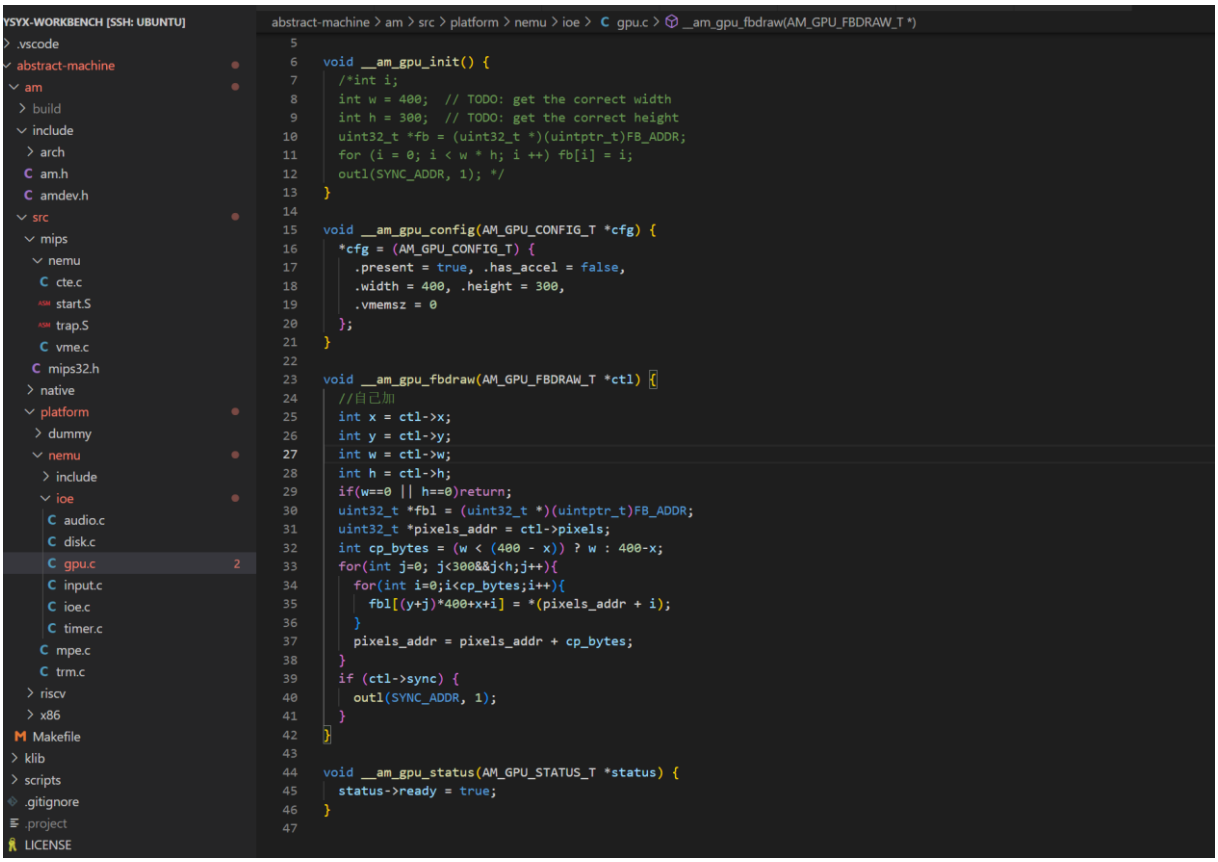
```
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'rtc' at [0xa0000048, 0xa000004f]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vgactl' at [0xa0000100, 0xa0000107]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'vmem' at [0xa1000000, 0xa10752ff]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'keyboard' at [0xa0000060, 0xa0000063]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio' at [0xa0000200, 0xa0000217]
[src/device/io/mmio.c:18 add_mmio_map] Add mmio map 'audio-sbuf' at [0xa1200000, 0xa120ffff]
[src/monitor/monitor.c:46 load_img] The image is /home/melissa/ysyx-workbench/nanos-lite/build/nanos-lite-riscv64-nemu.bin, size = 614544
[src/cpu/difftest/dut.c:69 init_difftest] Differential testing: ON
[src/cpu/difftest/dut.c:70 init_difftest] The result of every instruction will be compared with /home/melissa/ysyx-workbench/nemu/tools/spike-diff/build/r
the performance. If it is not necessary, you can turn it off in menuconfig.
[src/monitor/monitor.c:13 welcome] Trace: OFF
[src/monitor/monitor.c:17 welcome] Build time: 23:26:40, Jul 31 2022
Welcome to riscv64-NEMU!
For help, type "help"
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,12,main] 'Hello World!' from Nanos-lite
[/home/melissa/ysyx-workbench/nanos-lite/src/main.c,13,main] Build time: 23:26:39, Jul 31 2022
[/home/melissa/ysyx-workbench/nanos-lite/src/mm.c,26,init_mm] free physical pages starting from 000000008009f000
[/home/melissa/ysyx-workbench/nanos-lite/src/device.c,89,init_device] Initializing devices...
[/home/melissa/ysyx-workbench/nanos-lite/src/ramdisk.c,27,init_ramdisk] ramdisk info: start = 0000000080003419, end = 000000008008c019, size = 560128 byte
[/home/melissa/ysyx-workbench/nanos-lite/src/irq.c,17,init_irq] Initializing interrupt/exception handler...
[/home/melissa/ysyx-workbench/nanos-lite/src/proc.c,24,init_proc] Initializing processes...
打开文件 25loderok1
ehdr.e_phnum = 3
eok2
eok2
eok3
eok32
[/home/melissa/ysyx-workbench/nanos-lite/src/loader.c,48,naive_uload] Jump to entry = 0000000083005064
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=25,fd=1,open_offset=0,count=10

keyboard0
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=25,fd=1,open_offset=0,count=9

keyboard
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku D
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku S
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku D
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku S
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku D
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku S
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku D
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku S
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: kd D
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku D
[/home/melissa/ysyx-workbench/nanos-lite/src/fs.c,106,fs_write] fs_write:open_i=4,fd=1,open_offset=0,count=20

receive event: ku S
[src/cpu/cpu-exec.c:69 statistic] host time spent = 12,999,655 us
[src/cpu/cpu-exec.c:70 statistic] total guest instructions = 82,445,067
[src/cpu/cpu-exec.c:71 statistic] simulation frequency = 6,342,096 inst/s
make[1]: Leaving directory '/home/melissa/ysyx-workbench/nemu'
```

问题位置：在调用 read 之后，需要在 fs.c 中分类将 fd= 4 的情况分为使用 event_read 读取

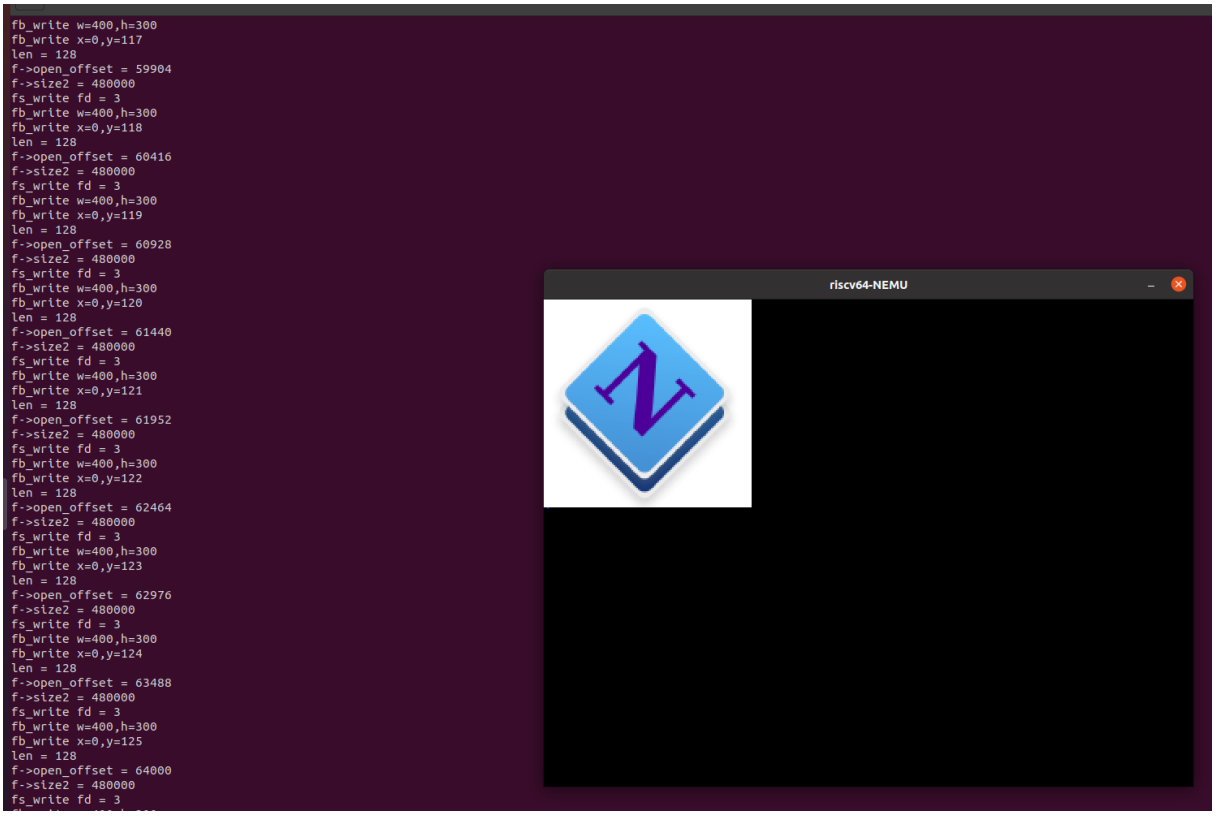
成功实现输出图片，但是由于 NDL 中 DRAWREACT()函数未成功添加，图片未居中一直在左上

角，后来查明是因为调用了 nemu 的

找到 am 中的参考



重新实现



精彩纷呈的应用程序

因为拷贝 windows 的 ppt 再次熟悉了一下 git 的操作

https://blog.csdn.net/heimu24/article/details/81081775

更丰富的运行环境

```
typedef struct {
    uint32_t flags;
    SDL_PixelFormat *format;
    int w, h;
    uint16_t pitch;
    uint8_t *pixels;
} SDL_Surface;
```

`SDL_Surface`是用于存储图像，可以用于图像绘制的结构体。 这里我们来看一下SDL2中的`SDL_Surface`结构体和与其有关的函数操作。 # SDL_Surface结构体 其实操作`SDL_Surface`的函数有很多，而且都很简单。但是如果不先认识`SDL_Surface`这个结构体的话，函数上的学习会比较困难。

```
typedef struct{
    SDL_PixelFormat* format;    //存储着和像素有关的格式                  read-only
    int w;                      //图像宽度                               read-only
    int h;                      //图像高度                               read-only
    int pitch;                  //pixels中一行有多少个像素（以Bytes计）   read-only
    void* pixels;               //实际的像素数据                          read-write
    void* userdata;             //用户数据，用户可以自己随意存储，读取    read-write
    SDL_Rect clip_rect;         //裁切矩形                               read-only
    int refcount;               //引用计数，一般由SDL函数自己改变
}SDL_Surface;
```

SDL_PixelFormat

A structure that contains pixel format information.

Data Fields

Uint32	format	one of the SDL_PixelFormatEnum values
SDL_Palette*	palette	an SDL_Palette structure associated with this pixel format, or NULL if the format doesn't have a palette
Uint8	BitsPerPixel	the number of significant bits in a pixel value, eg: 8, 15, 16, 24, 32
Uint8	BytesPerPixel	the number of bytes required to hold a pixel value, eg: 1, 2, 3, 4; see Remarks for related data type
Uint32	Rmask	a mask representing the location of the red component of the pixel
Uint32	Gmask	a mask representing the location of the green component of the pixel
Uint32	Bmask	a mask representing the location of the blue component of the pixel
Uint32	Amask	a mask representing the location of the alpha component of the pixel or 0 if the pixel format doesn't have any alpha information
Uint8	Rloss	(internal use)
Uint8	Gloss	(internal use)
Uint8	Bloss	(internal use)
Uint8	Aloss	(internal use)
Uint8	Rshift	(internal use)
Uint8	Gshift	(internal use)
Uint8	Bshift	(internal use)
Uint8	Ashift	(internal use)
int	refcount	(internal use)
SDL_PixelFormat*	next	(internal use)

SDL_Palette

A structure that contains palette information.

Data Fields

int	**ncolors**	the number of colors in the palette
SDL_Color*	**colors**	an array of SDL_Color structures representing the palette
Uint32	**version**	incrementally tracks changes to the palette (internal use)
int	**refcount**	reference count (internal use)

SDL_Color

A structure that represents a color.

Data Fields

Uint8	**r**	the red component in the range 0-255
Uint8	**g**	the green component in the range 0-255
Uint8	**b**	the blue component in the range 0-255
Uint8	**a**	the alpha component in the range 0-255

write函数（写入文件）

它的主要功能是：将某个文件缓冲区的数据，写入某个文件内。

系统调用格式：

number = write(handle, buffer, n) ；

write函数各个参数定义如下：

l handle： 这是一个已经打开的文件句柄，表示将数据写入这个文件句柄所表示的文件内。

l buffer： 表示缓冲区，也就是把这个缓冲区的数据写入文件句柄所表示的文件内。

l n： 表示调用一次write操作，应该写如多少字符。

l number：表示系统实际写入的字符数量。

如果调用write失败，则系统返回 - 1给number。

```c
void NDL_DrawRect(uint32_t *pixels, int x, int y, int w, int h) {
  int fd =open("/dev/fb",O_RDWR);
  printf("NDL_DrawRect w is %d h is %d\n",canvas_w,canvas_h);
  int offset = (y >=0 ? y:0)*screen_w + (x>=0 ? x: 0);
  int len;
  if(x>=0){
    len = (x + w < canvas_w ) ? w: canvas_w -x;
  }
  else {
    len = (x + w < canvas_w) ? x+w : canvas_w;
  }
  printf("len= %d\n",len);
  int  cnt_r = 0;
  for(int i=0; i<h; i++){
    //printf("drawing2\n");
    if(y+i>=0 && y+i < canvas_h){
      lseek(fd,offset*4 + cnt_r*screen_w*4,SEEK_SET);
      //printf("offset = %d",offset + cnt_r*canvas_w*4);
      write(fd,pixels+i*canvas_w,len*4);
      //printf("writing\n");
      cnt_r++;
    }
    else printf("error");
  }
  printf("Finish DRAWRECT!\n");
  //close(fd);
}
```

```c
size_t fb_write(const void *buf, size_t offset, size_t len) {
  //return 0;
  //自己加
  //参考代码
  int w = io_read(AM_GPU_CONFIG).width;
  int h = io_read(AM_GPU_CONFIG).height;
  printf("fb_write w=%d,h=%d\n",w,h);
  int x = ((offset )/4 )%w;
  int y = ((offset )/4)/w;

  printf("fb_write x=%d,y=%d\n",x,y);
  io_write(AM_GPU_FBDRAW,x,y,(uint32_t*)buf,len/4,1,true);
  printf("len = %d\n",len);
  return len;

}
```

这个 io_write 写的是像素？

运行 sh 命令 sh convert.sh

终于跑出来了 ppt 但是像素不是 32bits 的情况有地址越界，会出现 segmentation fault 很奇怪

先接着往后做，没有管地址溢出的问题

Navy 中的应用程序

翻页使用 SDL_WaitEvent()函数，隐含调用 SDL_PumpEvents()函数

SDL_WaitEvent

Wait indefinitely for the next available event.

Syntax

```
int SDL_WaitEvent(SDL_Event * event);
```

Function Parameters

event	the SDL_Event structure to be filled in with the next event from the queue, or NULL

Return Value

Returns 1 on success or 0 if there was an error while waiting for events; call SDL_GetError() for more information.

Remarks

If event is not NULL, the next event is removed from the queue and stored in the SDL_Event structure pointed to by event.

As this function may implicitly call SDL_PumpEvents(), you can only call this function in the thread that initialized the video subsystem.

SDL_PollEvent

Poll for currently pending events.

Syntax

```
int SDL_PollEvent(SDL_Event * event);
```

Function Parameters

event	the SDL_Event structure to be filled with the next event from the queue, or NULL

Return Value

Returns 1 if there is a pending event or 0 if there are none available.

```
typedef struct {
  uint8_t sym;
} SDL_keysym;

typedef struct {
  uint8_t type;
  SDL_keysym keysym;
} SDL_KeyboardEvent;

typedef struct {
  uint8_t type;
  int code;
  void *data1;
  void *data2;
} SDL_UserEvent;

typedef union {
  uint8_t type;
  SDL_KeyboardEvent key;
  SDL_UserEvent user;
} SDL_Event;
```

主要有 key，user

SDL_Keysym

A structure that contains key information used in key events.

Data Fields

SDL_Scancode	**scancode**	SDL physical key code; see SDL_Scancode for details
SDL_Keycode	**sym**	SDL virtual key code; see SDL_Keycode for details
Uint16	**mod**	current key modifiers; see SDL_Keymod for details
Uint32	**unused**	

SDL_UserEvent

A structure that contains an application-defined event type.

Data Fields

Uint32	**type**	value obtained from SDL_RegisterEvents()
Uint32	**timestamp**	timestamp of the event
Uint32	**windowID**	the associated window, if any
Sint32	**code**	user defined event code
void*	**data1**	user defined data pointer
void*	**data2**	user defined data pointer

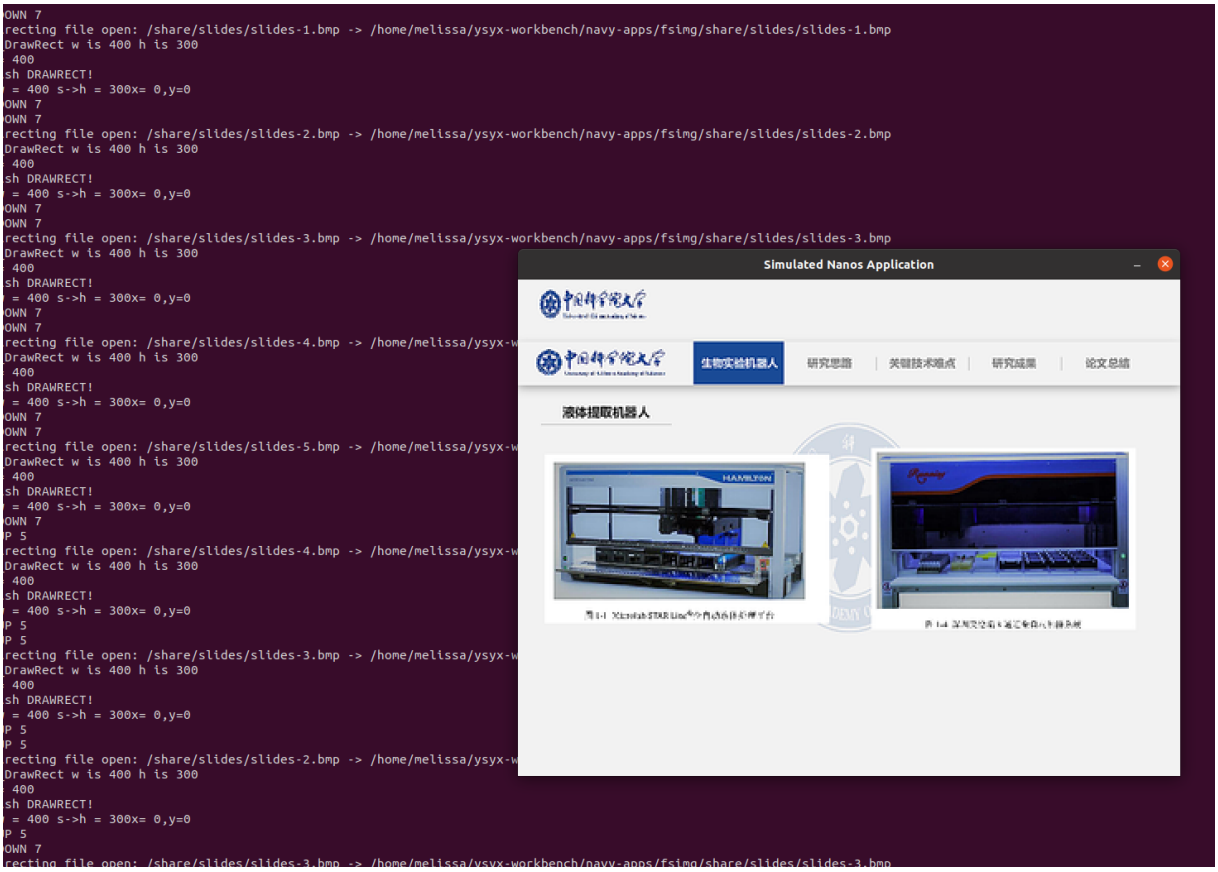
Code Examples

```
Uint32 myEventType = SDL_RegisterEvents(1);
if (myEventType != ((Uint32)-1)) {
    SDL_Event event;
    SDL_memset(&event, 0, sizeof(event)); /* or SDL_zero(event) */
    event.type = myEventType;
    event.user.code = my_event_code;
    event.user.data1 = significant_data;
    event.user.data2 = 0;
    SDL_PushEvent(&event);
}
```

strncmp 用来比较指定长度两个字符串的大小

函数介绍

函数原型：**int strncmp(const char* str1, const char* str2, size_t num)**
头 文 件：**#include <string.h>**
返 回 值：（与strncmp相同）str1 = str2 则返回0，
str1 > str2 则返回大于0的值，
str1 < str2 则返回小于0的值



开机菜单

SDL_FillRect

Perform a fast fill of a rectangle with a specific color.

Syntax

```
int SDL_FillRect
    (SDL_Surface * dst, const SDL_Rect * rect, Uint32 color);
```

Function Parameters

dst	the SDL_Surface structure that is the drawing target
rect	the SDL_Rect structure representing the rectangle to fill, or NULL to fill the entire surface
color	the color to fill with

SDL_BlitSurface

Use this function to perform a fast surface copy to a destination surface.

Syntax

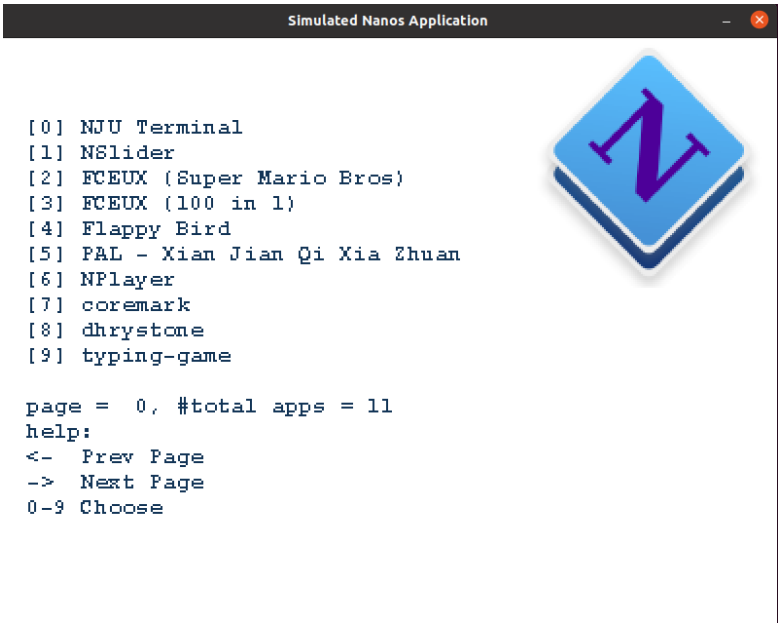
```
int SDL_BlitSurface(SDL_Surface*    src,
                    const SDL_Rect* srcrect,
                    SDL_Surface*    dst,
                    SDL_Rect*       dstrect)
```

Function Parameters

src	the SDL_Surface structure to be copied from
srcrect	the SDL_Rect structure representing the rectangle to be copied, or NULL to copy the entire surface
dst	the SDL_Surface structure that is the blit target
dstrect	the SDL_Rect structure representing the rectangle that is copied into

填充颜色的代码需要思考 ncolor 和 color 的关系

现象：(出现屏幕，按照讲义讲的确实无法选择选项)



NTerm

SDL_GetTicks

Get the number of milliseconds since SDL library initialization.

Syntax

```
Uint32 SDL_GetTicks(void);
```

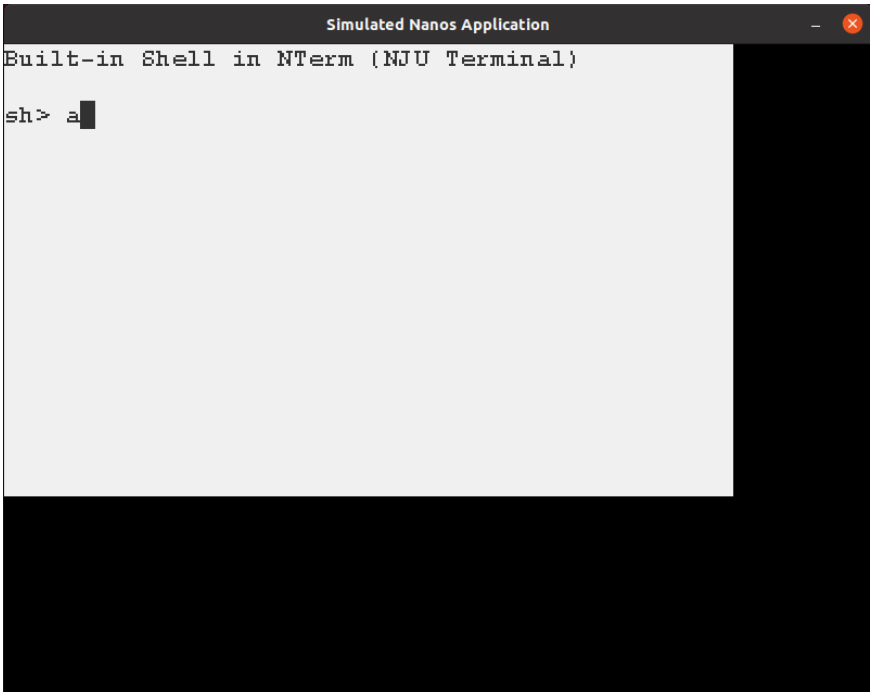
Return Value

Returns an unsigned 32-bit value representing the number of milliseconds since the SDL library initialized.

在这个函数中直接调用 NDL_GetTicks(),NDL 函数中使用了 gettimeofday

之前实现 SDL_POLLevent()，wait 的函数调用了 pollevent()

未考虑居中的情况，直接先显示,后续有时间可以回来考虑下 x,y 转化为 centerx,centery

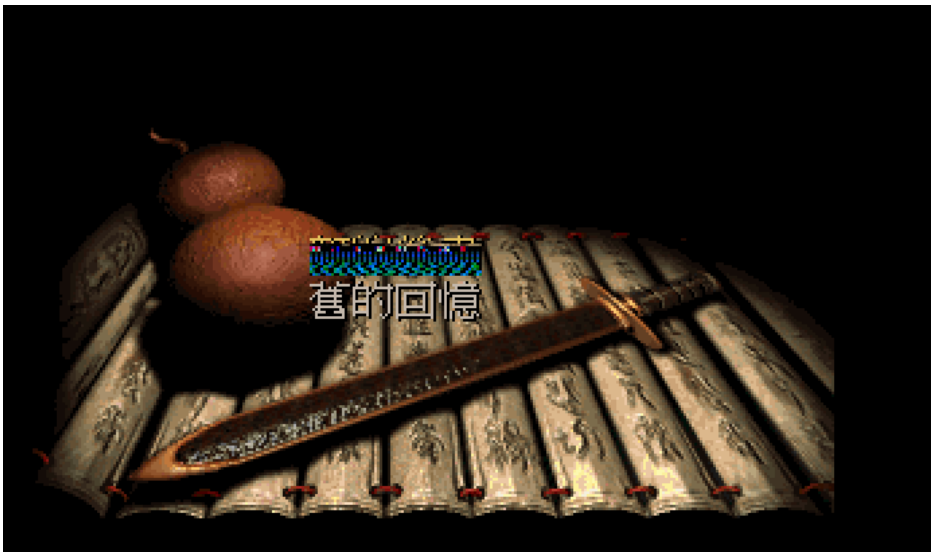


Flappy Bird

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
```

参数

- **ptr** -- 这是指向带有最小尺寸 *size*nmemb* 字节的内存块的指针。
- **size** -- 这是要读取的每个元素的大小，以字节为单位。
- **nmemb** -- 这是元素的个数，每个元素的大小为 size 字节。
- **stream** -- 这是指向 FILE 对象的指针，该 FILE 对象指定了一个输入流。



仙剑跑成了这个样子，很奇怪是因为没刷新？还是按键的问题？

单周期 NPC

nemu 的 makefile：

```
remove_quote = $(patsubst "%",%,$(1))
```

字符串替换，$(1)表示第一个参数变量

```
call
```

call 函数是唯一一个可以用来创建新的参数化的函数。你可以写一个非常复杂的表达式，这个表达式中，你可以定义许多参数，然后你可以用 call 函数来向这个表达式传递参数。其语法是：

```
$(call <expression>,<parm1>,<parm2>,<parm3>...)
```

当 make 执行这个函数时，<expression>参数中的变量，如$(1)，$(2)，$(3)等，会被参数<parm1>，<parm2>，<parm3>依次取代。而<expression>的返回值就是 call 函数的返回值。例如：

```
reverse = $(1) $(2)
foo = $(call reverse,a,b)
```

那么，foo 的值就是“a b”。当然，参数的次序是可以自定义的，不一定是顺序的，如：

```
reverse = $(2) $(1)
foo = $(call reverse,a,b)
```

此时的 foo 的值就是“b a”。

五、if 函数

if 函数很像 GNU 的 make 所支持的条件语句——ifeq（参见前面所述的章节），if 函数的语法是：

$(if <condition>,<then-part>)

或是

$(if <condition>,<then-part>,<else-part>)

可见，if 函数可以包含“else”部分，或是不含。即 if 函数的参数可以是两个，也可以是三个。<condition>参数是 if 的表达式，如果其返回的为非空字符串，那么这个表达式就相当于返回真，于是，<then-part>会被计算，否则<else-part>会被计算。

而 if 函数的返回值是，如果<condition>为真（非空字符串），那个<then-part>会是整个函数的返回值，如果<condition>为假（空字符串），那么<else-part>会是整个函数的返回值，此时如果<else-part>没有被定义，那么，整个函数返回空字串。

所以，<then-part>和<else-part>只会有一个被计算。

加 target run

在 main.cpp 函数中加入一些 paddr 函数，可以参考 nemu 的 memory 中 paddr.c

加 diff

参考同学的加入方式，但是不知道自己的 ereak()函数和 unknown_inst()是否好用，根据自己写的时钟不断判断参考 spike 动态库的寄存器与 npc 寄存器值，实现方式如下所示：

```cpp
#include "verilated_vcd_c.h"
#include <cassert>
#include <stdio.h>
#include <stdlib.h>
#include "Vysyx_22040175_top.h"
#include "verilated.h"
#include "verilated_dpi.h"
#include "assert.h"
#include <dlfcn.h>  //动态链接库相关函数
//加run和target
#define CONFIG_MBASE 0x80000000
#define CONFIG_MSIZE 0X2800000

typedef uint64_t word_t;
typedef  uint32_t paddr_t;
typedef  word_t vaddr_t;

static uint8_t *pimem =NULL;

enum{DIFFTEST_TO_DUT,DIFFTEST_TO_REF,NPC_STOP,NPC_RUNNING,NPC_END,NPC_ABORT};
uint32_t current_inst = 0;

//加ebreak
typedef struct {
  word_t gpr[32];
  vaddr_t pc;
} CPU_state; //nemu的CPU状态用于比较

CPU_state cpu = {};

uint32_t ebreak_flag = 0;
uint32_t unknown_code_flag = 0;
//DPIC
extern "C" void ebreak(){
  ebreak_flag = 1;
}
extern "C" void unknown_inst(){
  unknown_code_flag = 1;
}
//加difftest
void (*ref_difftest_memcpy)(paddr_t addr, void *buf, size_t n, bool direction) = NULL;
```

```
//加difftest
void (*ref_difftest_memcpy)(paddr_t addr, void *buf, size_t n, bool direction) = NULL;
void (*ref_difftest_regcpy)(void *dut, bool direction) = NULL;
void (*ref_difftest_exec)(uint64_t n) = NULL;
void (*ref_difftest_raise_intr)(word_t NO) = NULL;

int npc_state = NPC_STOP;
static int port = 1234;

//导出寄存器值 DPIC
uint64_t *cpu_gpr = NULL;
extern "C" void set_gpr_ptr(const svOpenArrayHandle r){
  cpu_gpr = (uint64_t *) (((VerilatedDpiOpenVar*)r) -> datap());
}
//函数声明
void init_imem();
uint8_t *guest_to_host(paddr_t paddr);
static long load_img(char *img_file);
word_t host_read(void *addr, int len);
int is_exit_status_bad();
void init_difftest(long img_size, int port);
void difftest_step(vaddr_t pc);
static void checkregs(CPU_state *ref, vaddr_t pc);
bool isa_difftest_checkregs(CPU_state *ref_r, vaddr_t pc);
static inline void host_write(void *addr, int len, word_t data);


//加为DPIC函数
/*extern "C" void pmem_read(paddr_t raddr,word_t *rdata){
  if(raddr <= CONFIG_MBASE){
    *rdata = host_read(guest_to_host(raddr), 8);
    printf("rdata = 0x%lx\n",*rdata);
  }
  else{
    *rdata = 0;
    printf("Warning: Invalid Instruction !\n");
  }
}*/
```

```
}*/

static word_t pmem_read(paddr_t addr, int len) {
  word_t ret = host_read(guest_to_host(addr), len);
  //printf("pmem_read success addr");
  return ret;
}

VerilatedContext *contextp = new VerilatedContext;
  // init top verilog instance
Vysyx_22040175_top* top = new Vysyx_22040175_top;
  // init trace dump

VerilatedVcdC* tfp = new VerilatedVcdC;




vluint64_t main_time = 0;
const vluint64_t max_sim_time = 2000;
//Vysyx_22040175_top *top;
int main(int argc, char **argv, char **env) {

  //Verilated::commandArgs(argc, argv);
  Verilated::traceEverOn(true);


  int i;
  int clk;
  int a = 0;

  top->trace (tfp, 99);
  tfp->open ("Vysyx_22040175.vcd");
  // initialize simulation inputs
  top->clk = 1;
  top->rst = 1;
  // run simulation for 100 clock periods
  char* img_file = *(argv + 1);

  printf("开始imem初始化\n");
```

```
116   printf("开始imem初始化\n");
117   init_imem();
118   long img_size = load_img(img_file);
119   init_difftest(img_size,port);
120   for (i=0; i<200; i++) {
121     top->rst = (i < 2);
122     // dump variables into VCD file and toggle clock
123     if(ebreak_flag){
124       printf("ebreak: program is finished !\n");
125       npc_state = NPC_END;
126       break;
127     }
128     for (clk=0; clk<2; clk++) {
129       tfp->dump (2*i+clk);
130       top->clk = !top->clk;
131 
132       top->eval ();
133   }
134 
135     if(top->clk==1){
136       top->inst = pmem_read(top->pc,8);
137       printf("main_time = %d\n",i);
138       printf("PC:0x%0x;Inst:0x%x;\n",top->pc,top->inst);
139       a= a+1;
140 
141 
142     }
143     if (a>2){
144       //printf("a =%d \n",a);
145       difftest_step(top->pc);
146     }
147       if(npc_state == NPC_ABORT){
148         printf("false:ABORT!The false PC is 0x%0lx\n",cpu.pc);
149         break;
150       }
151 
152   }
153 
154 
```

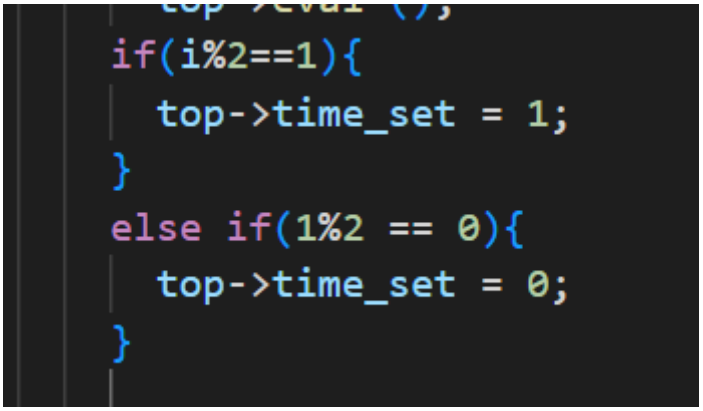
```
154
155     tfp->close();
156     delete top;
157     delete contextp;
158     return is_exit_status_bad();
159     if (Verilated::gotFinish())  exit(0);
160
161   }
162
163   void init_imem(){
164     pimem = (uint8_t *) malloc(CONFIG_MSIZE);
165    // printf("pimem _ success");
166     assert(pimem);
167   }
168
169   uint8_t *guest_to_host(paddr_t paddr){
170     uint8_t *tmpl = pimem + paddr -CONFIG_MBASE;
171     //printf("guest to host success pimem = %hhn\n",pimem);
172     //printf("guest to host success paddr  = %d\n",paddr );
173     //printf("guest to host success addr = %hhn\n",tmpl);
174     return tmpl;
175   }
176   inline word_t host_read(void *addr, int len) {
177      //printf("host_read success addr");
178     switch (len) {
179       case 1: return *(uint8_t  *)addr;
180       case 2: return *(uint16_t *)addr;
181       case 4: return *(uint32_t *)addr;
182       case 8: return *(uint64_t *)addr;
183       default: {printf("host_to_read is error !\n");assert(0);return 4096;};
184     }
185   }
186   static inline void host_write(void *addr, int len,word_t data){
187     switch (len){
188       case 1: *(uint8_t *)addr = data; return;
189       case 2: *(uint16_t *)addr = data; return;
190       case 4: *(uint32_t *)addr = data;return;
```

```
    case 8: *(uint64_t *)addr = data;return;
    default:{printf("hoost_write is error !\n"); assert(0);};
  }
}


static long load_img(char*img_file){
  if(img_file == NULL){
    printf("Error: No image is given !\n");
    assert(0);
    return 4096;
  }
  FILE *fp = fopen(img_file,"rb");
  assert(fp);
  fseek(fp, 0, SEEK_END);
  long size = ftell(fp);
  printf("The image is %s, size = %ld\n", img_file,size);
  fseek(fp, 0, SEEK_SET);
  int ret = fread(guest_to_host(CONFIG_MBASE), size, 1,fp);
  assert(ret == 1);
  fclose(fp);
  return size;
}
int is_exit_status_bad(){
  int good = (npc_state == NPC_END && cpu_gpr[10] == 0);
  if(good){
    printf("\033[1;32m HIT GOOD TRAP! \033[0m \n");
  }
  else{
    printf("\033[1;31m HIT BAD TRAP! \033[0m \n");
  }
  return !good;
}
//Difftest初始化
void init_difftest(long img_size,int port){
  char const *ref_so_file = "/home/melissa/ysyx-workbench/nemu/tools/spike-diff/build/riscv64-spike-so";
```

```
    char const *ref_so_file = "/home/melissa/ysyx-workbench/nemu/tools/spike-diff/build/riscv64-spike-so";
    if(ref_so_file != NULL){
      printf("文件加载成功! \n");
    }
    assert(ref_so_file != NULL);
    void *handle;
    handle = dlopen(ref_so_file,RTLD_LAZY); //将动态库加载到内存中
    if(handle == NULL){
      printf("库打开失败! \n");
      printf("dlopen error. msg:%s", dlerror());
    }
    assert(handle);
    //用函数指针指向动态库中的对应函数，以便调用
    ref_difftest_memcpy = (void(*)(paddr_t, void*, size_t, bool))dlsym(handle,"difftest_memcpy");
    assert(ref_difftest_memcpy);

    ref_difftest_regcpy = (void(*)(void*,bool))dlsym(handle,"difftest_regcpy");
    assert(ref_difftest_regcpy);

    ref_difftest_exec = (void(*)(uint64_t))dlsym(handle,"difftest_exec");
    assert(ref_difftest_exec);

    ref_difftest_raise_intr = (void(*)(uint64_t))dlsym(handle,"difftest_raise_intr");
    assert(ref_difftest_raise_intr);

    void(*ref_difftest_init)(int) = (void(*)(int))dlsym(handle,"difftest_init");
    assert(ref_difftest_init);

    ref_difftest_init(port);
    cpu.pc = CONFIG_MBASE;
    ref_difftest_memcpy(CONFIG_MBASE,guest_to_host(CONFIG_MBASE), img_size, DIFFTEST_TO_REF);
    for(int i = 0; i < 32;i++){
      cpu.gpr[i] = 0;
    }
    ref_difftest_regcpy(&cpu, DIFFTEST_TO_REF);
  }
  //Difftest在CPU中比较功能的实现
  void difftest_step (vaddr_t dnpc){
    CPU_state ref_r;
    ref_difftest_regcpy(&ref_r, DIFFTEST_TO_DUT);
    checkregs(&ref_r, dnpc);
```

```
    checkregs(&ref_r, dnpc);
    ref_difftest_exec(1);
  }


  static void checkregs(CPU_state *ref, vaddr_t dnpc){
    if(!isa_difftest_checkregs(ref,dnpc)){
      npc_state = NPC_ABORT;
    }
  }
  bool isa_difftest_checkregs(CPU_state *ref_r, vaddr_t dnpc){
    int i = 0;
    bool DIF_result = true;
    if(ref_r -> pc != dnpc){
      printf("False: PC is false! ref_dnpc is 0x%0lx;npc_dnpc is 0x%0lx; Instruction is 0x%x\n",ref_r->pc,dnpc,current_inst);
      DIF_result = false;
    }
    for (i=0; i<32;i++){
      if(ref_r->gpr[i] != cpu_gpr[i]){
        printf("False: Reg is false! ref_gpr[%d]: 0x%08lx;npc_gpr[%d]; 0x%08lx; Instruction is 0x%x\n",i,ref_r->gpr[i],i,cpu_gpr[i],top->inst);
        DIF_result = false;
      }
    }
    return DIF_result;
  }
```

运行 bit 文件内存 sh 出现问题，对于涉及内存写操作，根据时间不同加一个限制，保证指令在自己的周期运行。



verilog 算术右移 简单方法

原创 Matts Tian 于 2020-06-23 16:13:57 发布 2706 收藏

result = ($signed (B)) >>> A;

通过 bit

bubble-sort 新增指令：slli、bge、mulw

div：divw

区分 beq/divw 有符号数与无符号数

goldbach：remw

hello-str：bltu、bgeu 没通过出错位置：li

if-else：blt

if-else、leap-year 通过

load-store：lh、lhu 通过

matrix-mul 通过

发现 addi 截取为 32 位，max 通过，hello-str 解决错误

hello-str：ori、通过

min3、mov-c 通过

movsx:sraiw 通过

mul-longlong:新增 mul 通过

pascal 通过、recursion 通过、select-sort 通过

shift：sraliw、sraw、srlw 通过

shuixianhua、string、sub-longlong、sum、swtich、to-lower-case、unalign、wanshu 通过

五级流水 CPU

改造数据通路

首先添加各级寄存器，为方便理清逻辑关系，将级与级、级与寄存器的连接放到了顶层模块，下一步需要理清数据冒险与控制冒险

解决数据冒险与控制冒险

使用旁路的方式解决一般跳变（两拍，为了通过 diff，两拍 pc 值一样但是在第二拍指令变为 nop 指令，nop：addiw x0 0）

使用阻塞与旁路的方式解决需要上一条指令访存值与当前指令取值的冲突（一共两个时钟周期）

使用旁路的方式解决上一条指令执行值与当前指令取值的冲突

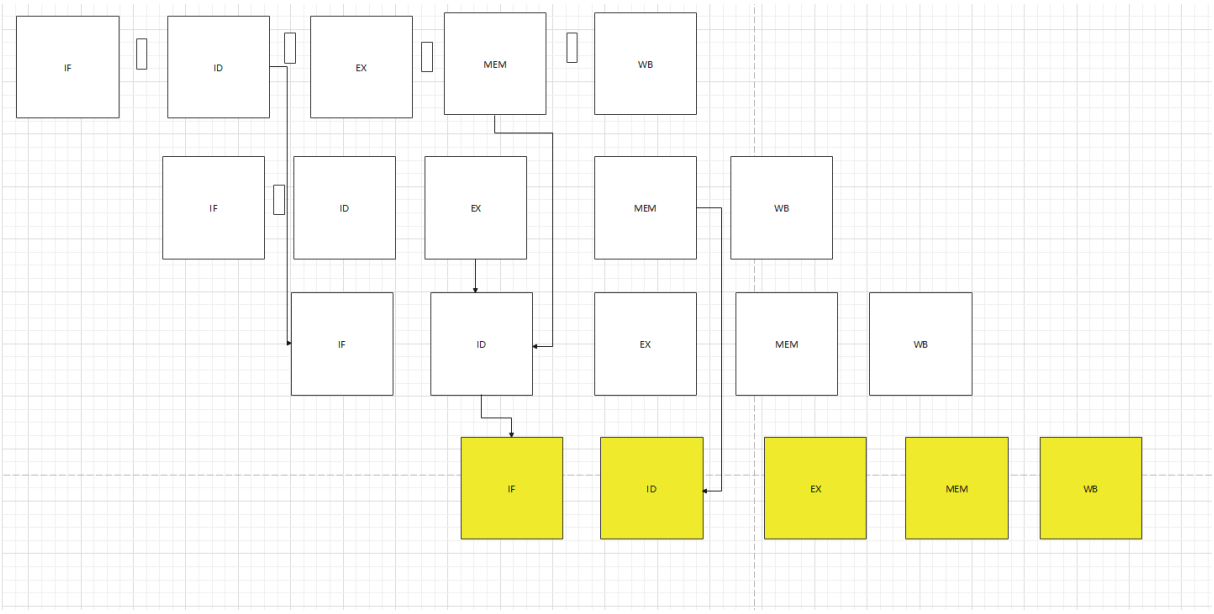
使用旁路的方式解决上上条指令访存值与当前指令取值的冲突

使用旁路的方式解决一般跳变 + 上一条指令执行值与当前指令取值的冲突

使用旁路的方式解决一般跳变 + 上上条指令访存值与当前指令取值的冲突（两拍？）

使用阻塞加旁路解决 跳变 + 上一条指令访存值与当前指令取值的冲突（一共三个时钟周期）

下面的图比较简易主要是为了提示自己目前遇到的冲突



通过一顿 debug 操作终于：

```
[           sum] PASS!
[       if-else] PASS!
[           add] PASS!
[   shuixianhua] PASS!
[ to-lower-case] PASS!
[         prime] PASS!
[         dummy] PASS!
[           div] PASS!
[           max] PASS!
[         mov-c] PASS!
[           bit] PASS!
[       pascal] PASS!
[       string] PASS!
[           fib] PASS!
[      goldbach] PASS!
[   select-sort] PASS!
[       unalign] PASS!
[    matrix-mul] PASS!
[         movsx] PASS!
[     leap-year] PASS!
[          fact] PASS!
[        wanshu] PASS!
[     hello-str] PASS!
[   bubble-sort] PASS!
[         shift] PASS!
[     recursion] PASS!
[  mul-longlong] PASS!
[  add-longlong] PASS!
[  sub-longlong] PASS!
[        switch] PASS!
[    quick-sort] PASS!
[    load-store] PASS!
[          min3] PASS!
```

总结：还是需要先搭好 diff 然后看波形 debug，总是会有没有考虑到的情况出现，在过程中不

断加条件打补丁，最后程序结果尤其是译码阶段 mux_dt_pipe 模块条件比较多，整体程序使用了太多 always@(*)听说这样很不好，下一步思考为什么这种组合逻辑不好，并优化一下整体架构，希望可以使整体架构变得更简单。

在解决五级流水 latch 的部分时遇到了 signal unoptfloat 警告，大致意思是形成环路，信号随时钟振荡，打破环路就行，

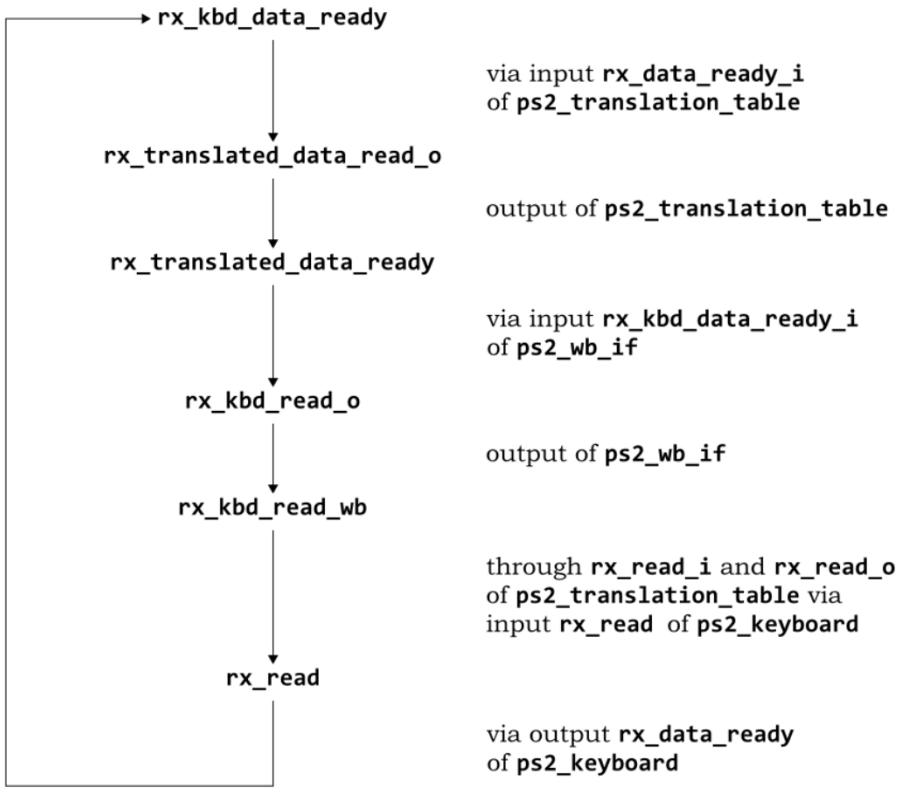
错误代码：reg[63:0] reg_f[0:31];

always@(*)中在 default 的情况使用了 reg_f = reg_f;自己形成了环路，

将 defau 情况中的赋值语句改为 reg_f = reg1_wdadta[31:0]就可以解决这个警告

需要注意的是 unoptfloat 警告在 verilator 中哪怕在 mk 中写入-Wno-lint 也不会忽略

必须解决此种警告程序才可以被 vreilator 仿真



AXI 总线与类 SRAM 总线

通道	信号	位宽	源	描述
全局	ACLK	1	-	全局时钟信号，其余信号均在时钟上升沿被采集
	ARESETn	1	-	全局复位信号，低电平有效
写地址	AWADDR	64	主	写请求的地址
	AWVALID	1	主	写地址握手信号，表示写地址通道信号有效
	AWREADY	1	从	写地址握手信号，表示 slave 可以接收写地址通道信号
写数据	WDATA	64	主	写请求的数据
	WSTRB	8	主	写请求选通信号，表示哪些写请求数据的哪些字节为有效数据
	WVALID	1	主	写数据握手信号，表示写数据通道信号有效
	WREADY	1	从	写数据握手信号，表示 slave 可以接收写数据通道信号
写响应	BVALID	1	从	写响应握手信号，表示 slave 完成了一次写事务
	BREADY	1	主	写响应握手信号，表示 master 可以接收写响应通道信号
读地址	ARADDR	64	主	读请求的地址
	ARVALID	1	主	读请求握手信号，表示读地址通道信号有效
	ARREADY	1	从	读请求握手信号，表示 slave 可以接收读地址通道信号
读数据	RDATA	64	从	读请求读回的数据
	RVALID	1	从	读数据握手信号，表示 slave 完成了一次读事务，并返回了读到的数据
	RREADY	1	主	读数据握手信号，表示 master 可以接收读回的数据

主要任务：

1. 修改异步 ram 为同步 ram
2. 添加类 sram 接口
3. 添加 AXI 接口
4. 添加类 sram 转 AXI 接口
5. 完成总线读，进行读取指令的测试

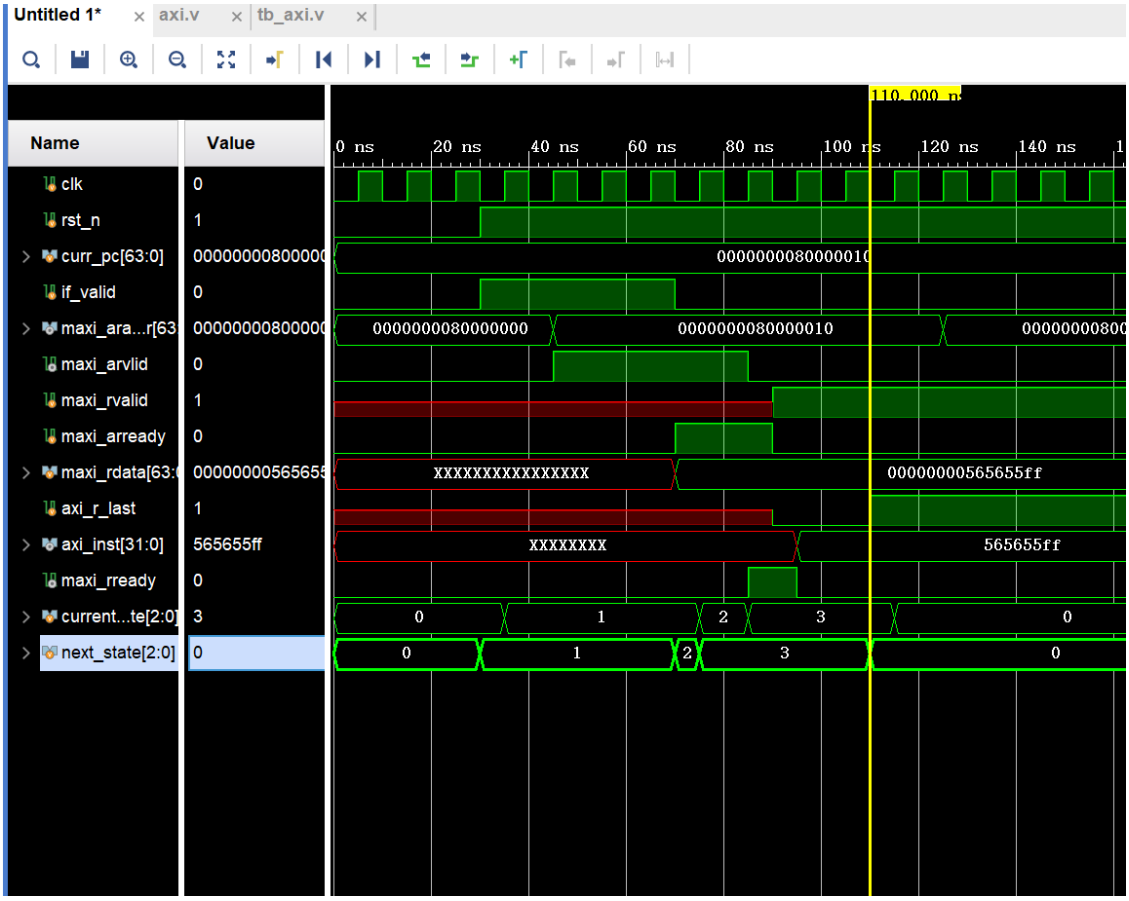
//好像 ysyx 直接改就可以了，不需要像设计实战一样先转化为类 RAM 接口

今天发现可以直接先考虑读指令的加入，因为可以直接把接口拉出来作为 AXI 接口尝试一下，可能不需要按照 CPU 设计实战讲的将接口先转化为类 SRAM 接口再搭转接桥最后再到 AXI。

考虑 IF 和 MEM 同时读取冲突，仲裁问题

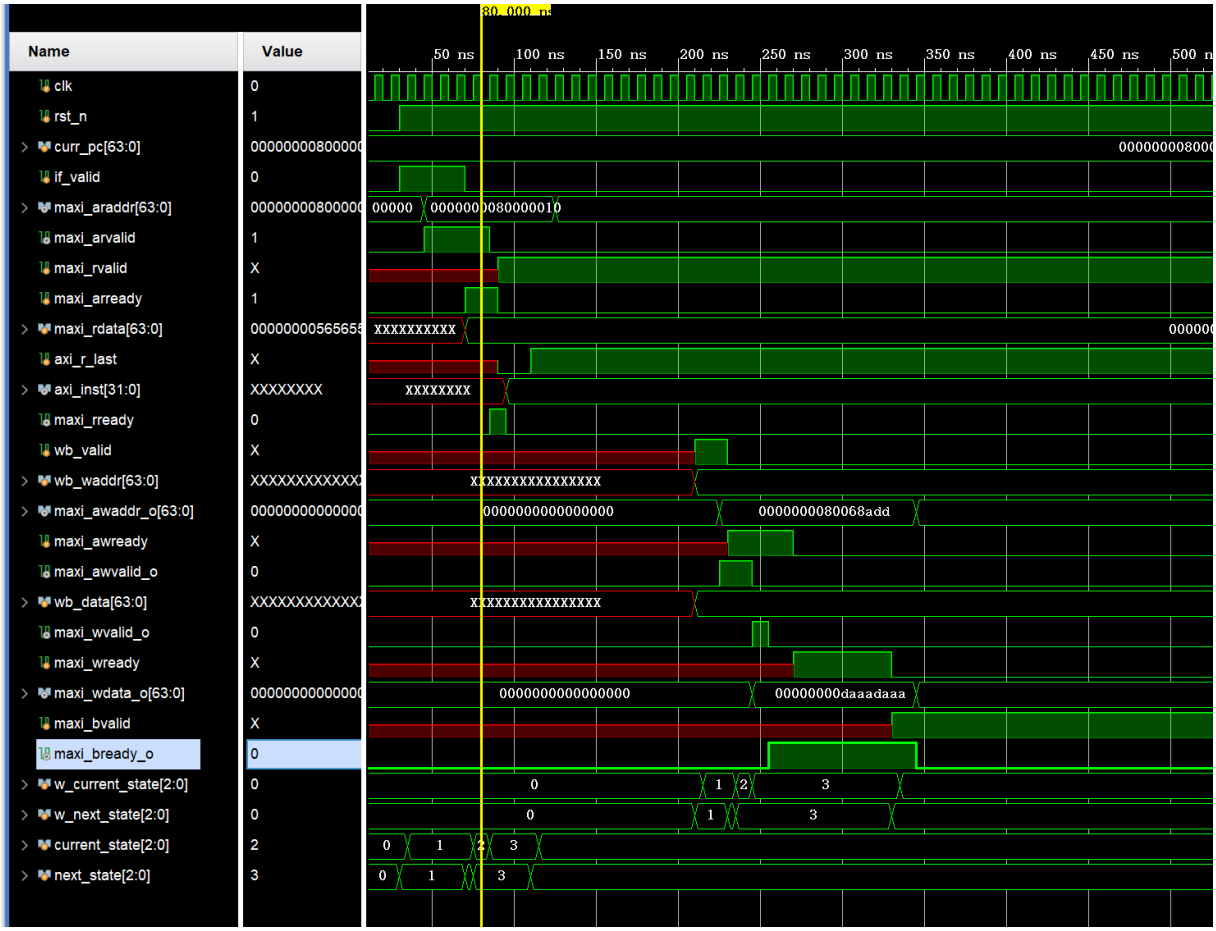
完成总线读指令：

1. 只考虑单周期 if 读指令的情况
2. 进行 tb 仿真测试
3. 采用三段式状态机完成总线读



任务：

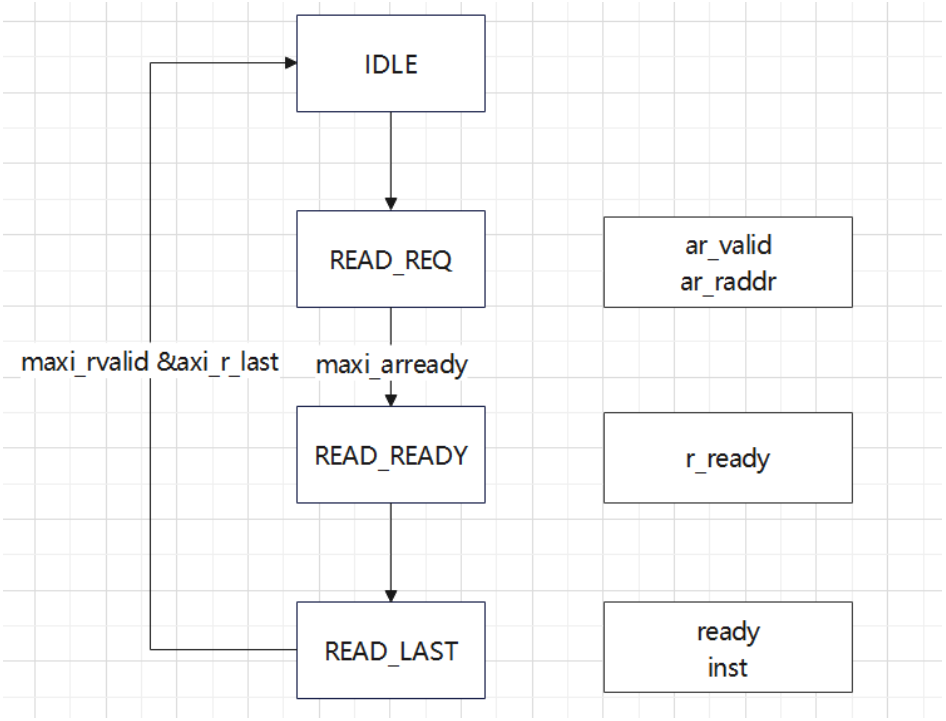
1. 完成总线写的状态机
2. 进行 tb 测试



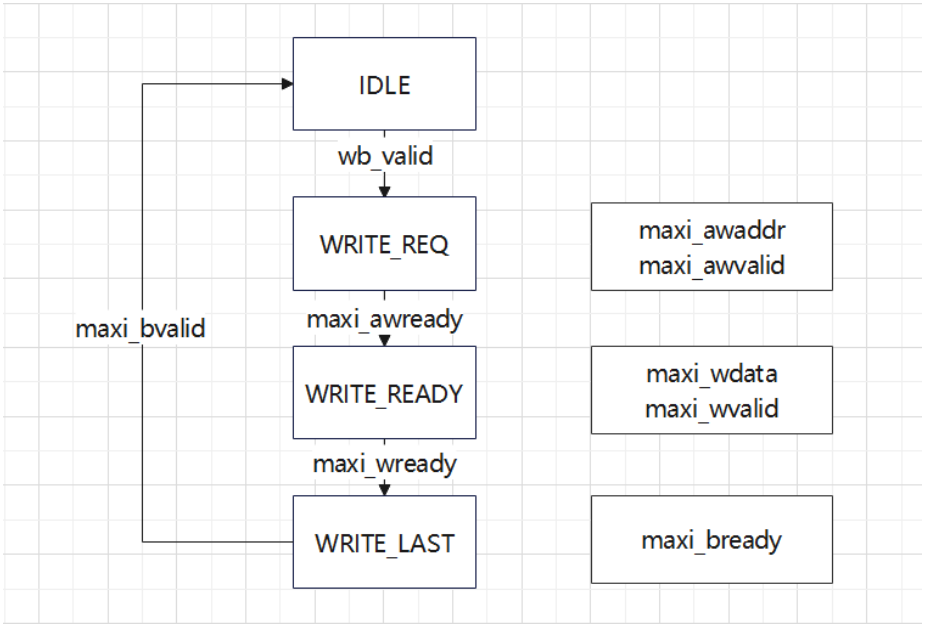
读写总线的状态机：(简易版)

没有考虑突发、窄传输，多主机（if，mem）的情况

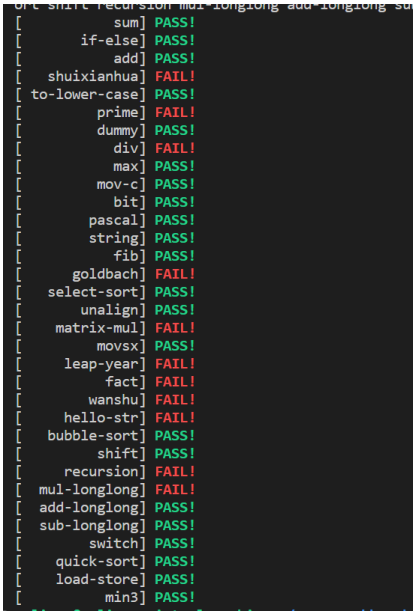
写状态机



读状态机



写从机验证总线读取指令



查测试的反汇编发现是 beq 附近有问题，明天查一下

特定情况让 dd_r_done 为 1 为什么不可以？？？和 mul 指令有关

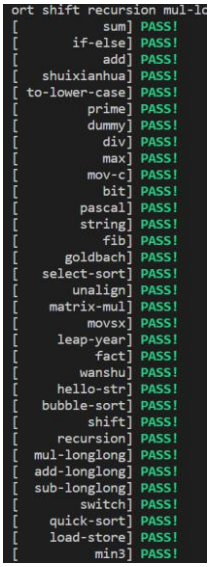
```
ort shift recursion mul-long
[           sum] PASS!
[       if-else] PASS!
[           add] PASS!
[   shuixianhua] FAIL!
[ to-lower-case] PASS!
[         prime] PASS!
[         dummy] PASS!
[           div] PASS!
[           max] PASS!
[         mov-c] PASS!
[           bit] PASS!
[        pascal] PASS!
[        string] PASS!
[           fib] PASS!
[      goldbach] FAIL!
[   select-sort] PASS!
[       unalign] PASS!
[    matrix-mul] PASS!
[         movsx] PASS!
[     leap-year] PASS!
[          fact] FAIL!
[        wanshu] PASS!
[     hello-str] PASS!
[   bubble-sort] PASS!
[         shift] PASS!
[     recursion] PASS!
[  mul-longlong] PASS!
[  add-longlong] PASS!
[  sub-longlong] PASS!
[        switch] PASS!
[    quick-sort] PASS!
[    load-store] PASS!
[          min3] PASS!
```

需要解决连续乘除的情况

```
[           sum] PASS!
[       if-else] FAIL!
[           add] PASS!
[   shuixianhua] PASS!
[ to-lower-case] PASS!
[         prime] PASS!
[         dummy] PASS!
[           div] PASS!
[           max] FAIL!
[         mov-c] PASS!
[           bit] PASS!
[        pascal] FAIL!
[        string] FAIL!
[           fib] PASS!
[      goldbach] PASS!
[   select-sort] PASS!
[       unalign] PASS!
[    matrix-mul] PASS!
[         movsx] PASS!
[     leap-year] PASS!
[          fact] PASS!
[        wanshu] FAIL!
[     hello-str] FAIL!
[   bubble-sort] PASS!
[         shift] PASS!
[     recursion] FAIL!
[  mul-longlong] PASS!
[  add-longlong] PASS!
[  sub-longlong] PASS!
[        switch] FAIL!
[    quick-sort] PASS!
[    load-store] PASS!
[          min3] FAIL!
```

改了 pc_predict 中 control_rest 的时候出现了很多错误

终于跑通了，总线读取指令已完成



开始接访存总线，访存读指令完成

读、写内存的总线已接好，分别在 mem 和 write 中

```
[           sum] PASS!
[       if-else] PASS!
[           add] PASS!
[   shuixianhua] PASS!
[ to-lower-case] PASS!
[         prime] PASS!
[         dummy] PASS!
[           div] PASS!
[           max] PASS!
[         mov-c] FAIL!
[           bit] PASS!
[        pascal] PASS!
[        string] PASS!
[           fib] PASS!
[      goldbach] PASS!
[   select-sort] PASS!
[       unalign] FAIL!
[    matrix-mul] PASS!
[         movsx] FAIL!
[     leap-year] PASS!
[          fact] PASS!
[        wanshu] PASS!
[     hello-str] FAIL!
[   bubble-sort] PASS!
[         shift] PASS!
[     recursion] PASS!
[  mul-longlong] PASS!
[  add-longlong] PASS!
[  sub-longlong] PASS!
[        switch] PASS!
[    quick-sort] PASS!
[    load-store] PASS!
[          min3] PASS!
```

解决 sd

```
 sum if-else add shuixianhua to-lower-case prime dummy div max mov-c bit pascal string fib goldbach select-sort unalign matrix-mul movsx leap-year fact wanshu h
ort shift recursion mul-longlong add-longlong sub-longlong switch quick-sort load-store min3
[           sum] PASS!
[       if-else] PASS!
[           add] PASS!
[   shuixianhua] PASS!
[ to-lower-case] PASS!
[         prime] PASS!
[         dummy] PASS!
[           div] PASS!
[           max] PASS!
[         mov-c] PASS!
[           bit] PASS!
[        pascal] PASS!
[        string] PASS!
[           fib] PASS!
[      goldbach] PASS!
[   select-sort] PASS!
[       unalign] PASS!
[    matrix-mul] PASS!
[         movsx] PASS!
[     leap-year] PASS!
[          fact] PASS!
[        wanshu] PASS!
[     hello-str] PASS!
[   bubble-sort] PASS!
[         shift] PASS!
[     recursion] PASS!
[  mul-longlong] PASS!
[  add-longlong] PASS!
[  sub-longlong] PASS!
[        switch] PASS!
[    quick-sort] PASS!
[    load-store] PASS!
[          min3] PASS!
```

新增 AXI4 的 xID 信号

通道	信号	位宽	源	描述
写地址	AWID	4	主	发出写地址的 master 的 ID
写响应	BID	4	从	响应 ID 为 BID 的 master 所发出的写请求
读地址	ARID	4	主	发出读地址的 master 的 ID
读数据	RID	4	从	响应 ID 为 RID 的 master 所发出的读请求

窄传输

- 示例1
  - 该 burst 含有 5 个 transfer
  - 起始地址 (AxADDR) 为 0
  - 每一个 transfer 为 8 比特
  - 数据总线位宽 32
  - burst 类型为 INCR

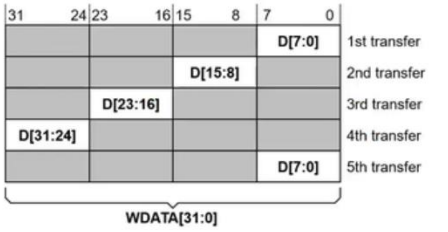


Figure A3-8 Narrow transfer example with 8-bit transfers

- 示例2
  - 该 burst 含有 3 个 transfer
  - 起始地址 (AxADDR) 为 4
  - 每一个 transfer 为 32 比特
  - 数据总线位宽 64
  - burst 类型为 INCR

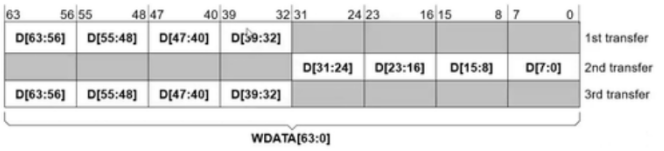
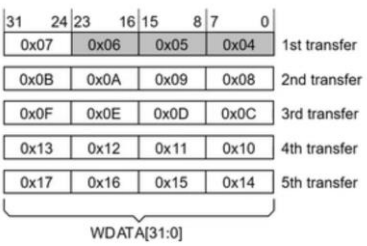


Figure A3-9 Narrow transfer example with 32-bit transfers

非对齐传输

- 示例
  - burst 长度为 5
  - burst 宽度为 32 比特
  - burst 类型为 INCR
  - 起始地址为 7



- 起始地址 AxADDR 与**突发宽度** AxSIZE 不对齐
- 对于非对齐传输，master 可以把它**变成对齐**的：
  - 在首个 transfer 填充无效数据至对齐
  - 将起始地址调整为对齐的地址
- 然后，问题就退化为了窄传输/正常传输
- 特别提示
  - master 必须保证 WSTRB **只在对应字节有效时为高**

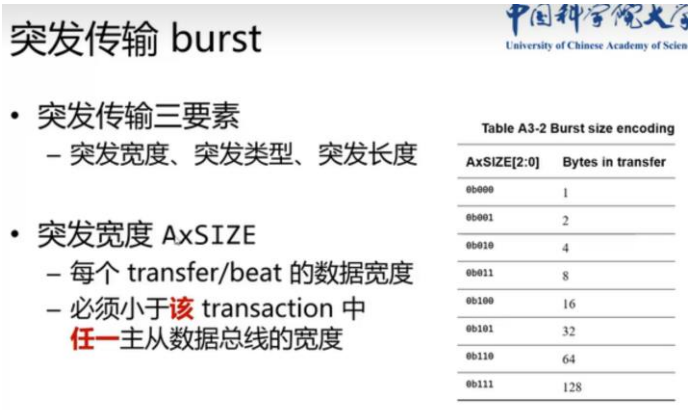
更改文件权限

```
chmod: cannot access 'build/sh': No such file or directory
melissa@melissa-virtual-machine:~/windowsrepo/oscpu-framework$
melissa@melissa-virtual-machine:~/windowsrepo/oscpu-framework$ sudo chmod -R 777 build.sh
melissa@melissa-virtual-machine:~/windowsrepo/oscpu-framework$ ls -alh ./build.sh
-rwxrwxrwx 1 melissa melissa 12K 7月  29 10:39 ./build.sh
```

更改 nemu 的内存

```
                                Memory Configuration
hlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to

                        (0x80000000) Memory base address
                        (0x8000000) Memory size
                        (0) Offset of reset vector from the base of memory
                            Physical memory definition (Using global array)  --->
```

突发传输



Cache

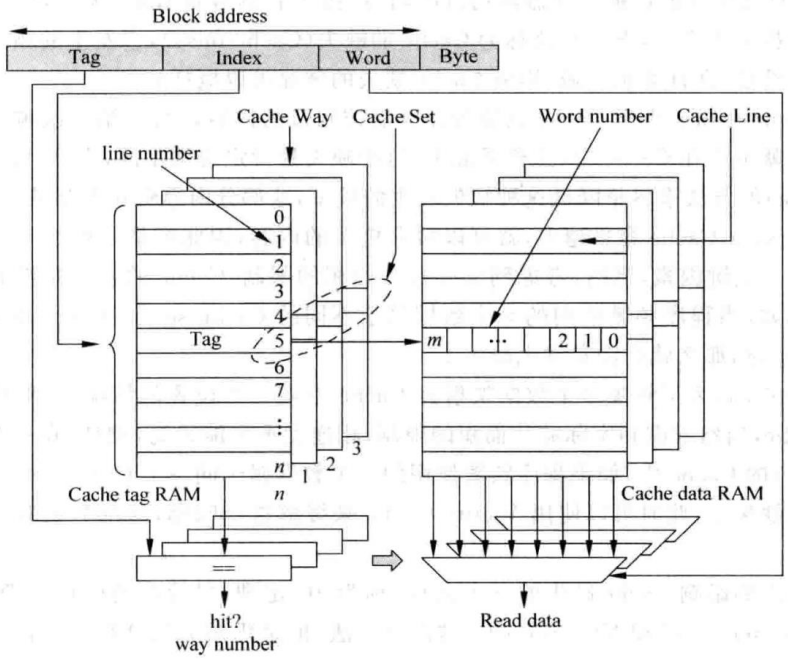


图 2.2 Cache 的结构

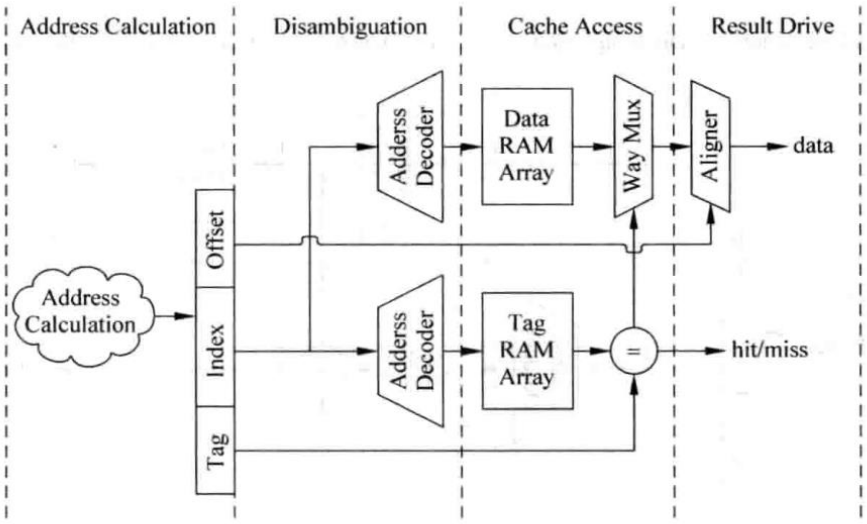


图 2.8 并行访问的流水线

对于顺序执行的处理器，cache 选择并行访问的方式

我打算使用两路组相连的方式实现本次的 cache

cache 的写入 （使用 L2-D）

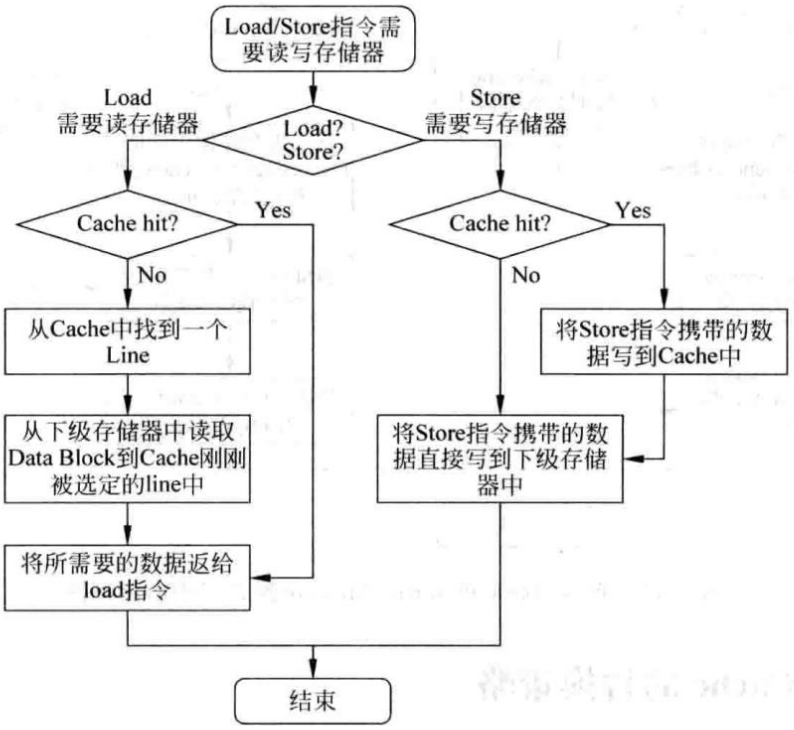


图 2.13　Write Through 和 Non-Write Allocate 两种方法配合工作的流程图

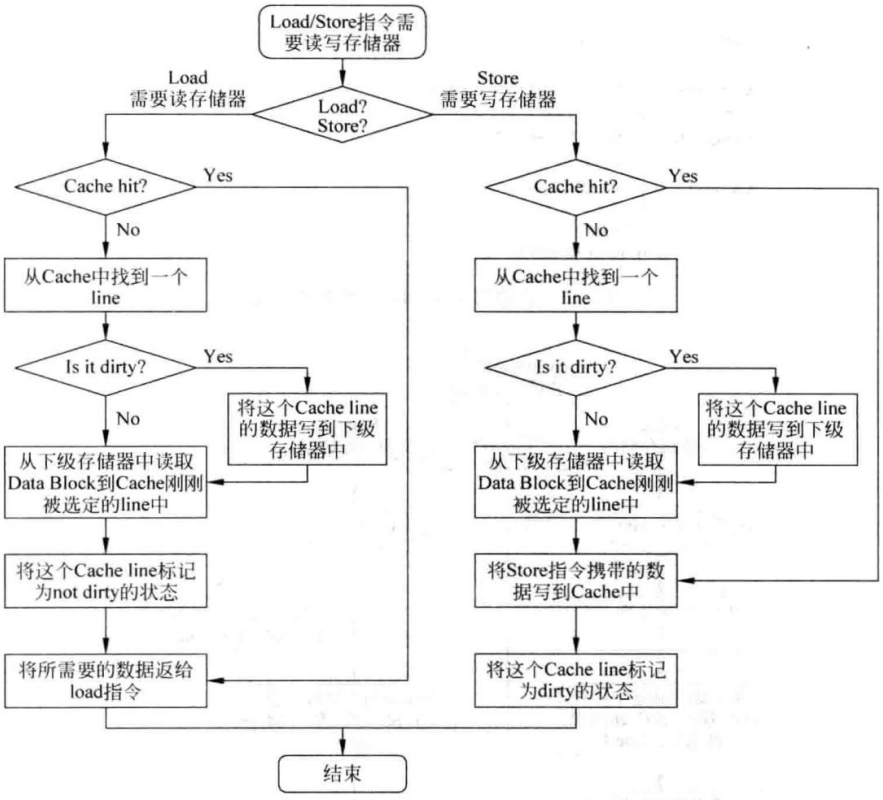
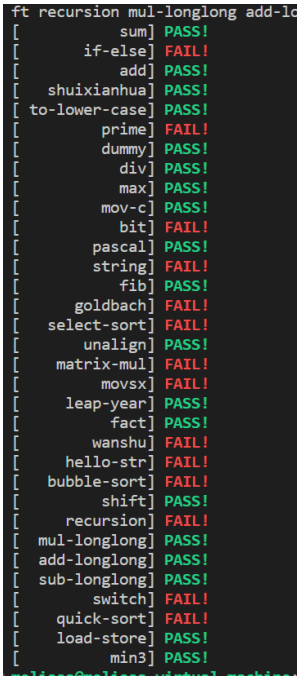
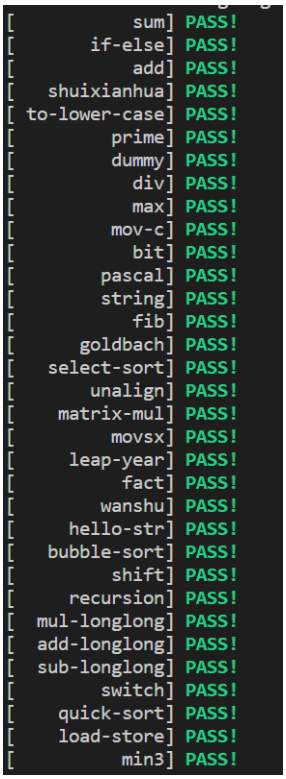


图 2.14　Write back 和 Write Allocate 配合工作的流程图

8.27 加 icache 后

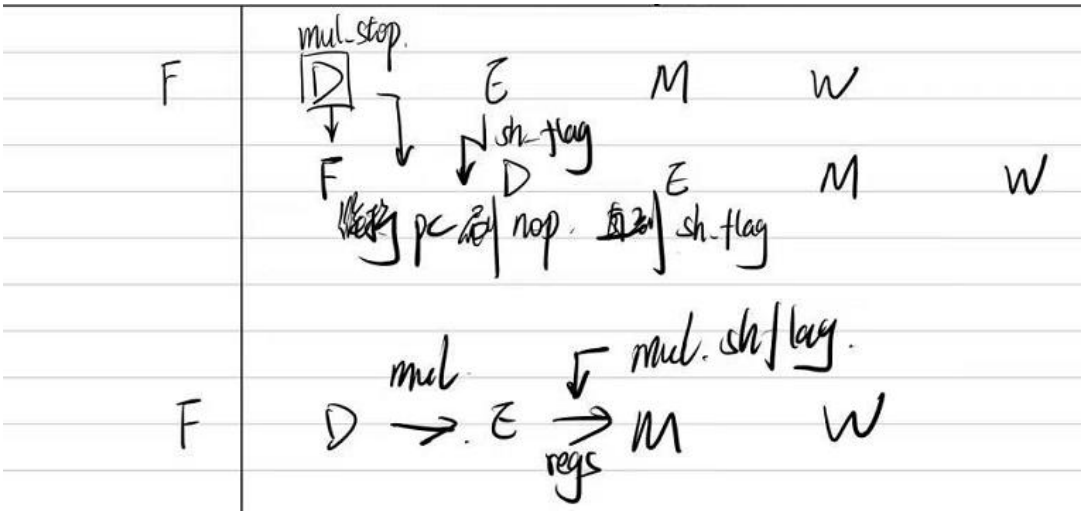

ft recursion mul-longlong add-lo
[sum] PASS!
[if-else] FAIL!
[add] PASS!
[shuixianhua] PASS!
[to-lower-case] PASS!
[prime] FAIL!
[dummy] PASS!
[div] PASS!
[max] PASS!
[mov-c] PASS!
[bit] FAIL!
[pascal] PASS!
[string] FAIL!
[fib] PASS!
[goldbach] FAIL!
[select-sort] FAIL!
[unalign] PASS!
[matrix-mul] FAIL!
[movsx] FAIL!
[leap-year] PASS!
[fact] PASS!
[wanshu] FAIL!
[hello-str] FAIL!
[bubble-sort] FAIL!
[shift] PASS!
[recursion] FAIL!
[mul-longlong] PASS!
[add-longlong] PASS!
[sub-longlong] PASS!
[switch] FAIL!
[quick-sort] FAIL!
[load-store] PASS!
[min3] PASS!

icache 调试成功


[sum] PASS!
[if-else] PASS!
[add] PASS!
[shuixianhua] PASS!
[to-lower-case] PASS!
[prime] PASS!
[dummy] PASS!
[div] PASS!
[max] PASS!
[mov-c] PASS!
[bit] PASS!
[pascal] PASS!
[string] PASS!
[fib] PASS!
[goldbach] PASS!
[select-sort] PASS!
[unalign] PASS!
[matrix-mul] PASS!
[movsx] PASS!
[leap-year] PASS!
[fact] PASS!
[wanshu] PASS!
[hello-str] PASS!
[bubble-sort] PASS!
[shift] PASS!
[recursion] PASS!
[mul-longlong] PASS!
[add-longlong] PASS!
[sub-longlong] PASS!
[switch] PASS!
[quick-sort] PASS!
[load-store] PASS!
[min3] PASS!

乘法器

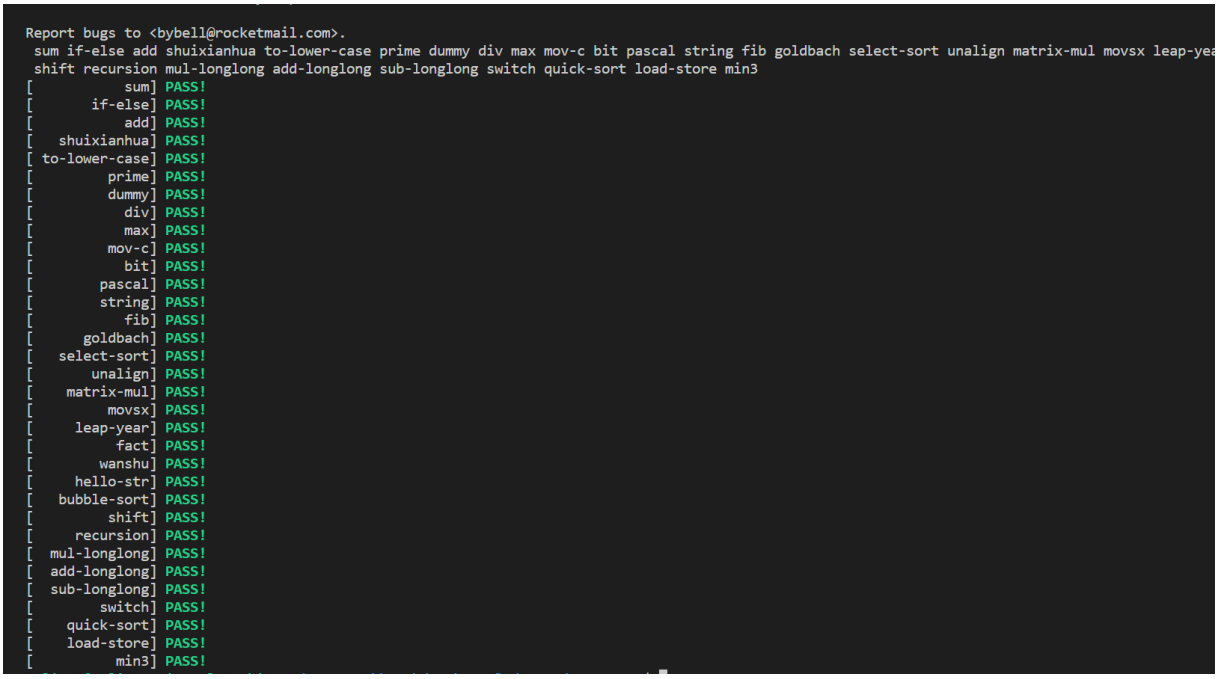
采用二位 booth 编码写了乘法器，没有使用华莱士树，乘法器的暂停关系如下



除法器

按照讲义的说法目标实现 radix-2 除法器

上述错误是没有考虑到 res 需要延迟一拍再赋值给 alu_res2,考虑到 id_rest，需要将执行结果传给下一条指令的译码单元，除法器的接入思路与乘法器类似



除法器要点：在判断被除数与除数减法结果，替换被除数与左移的操作的可以例化为模块，再在外部使用 generate for 来通过 genvar i 不断进行调用操作，通过 i 来控制模块例化次数。

主要矛盾点：

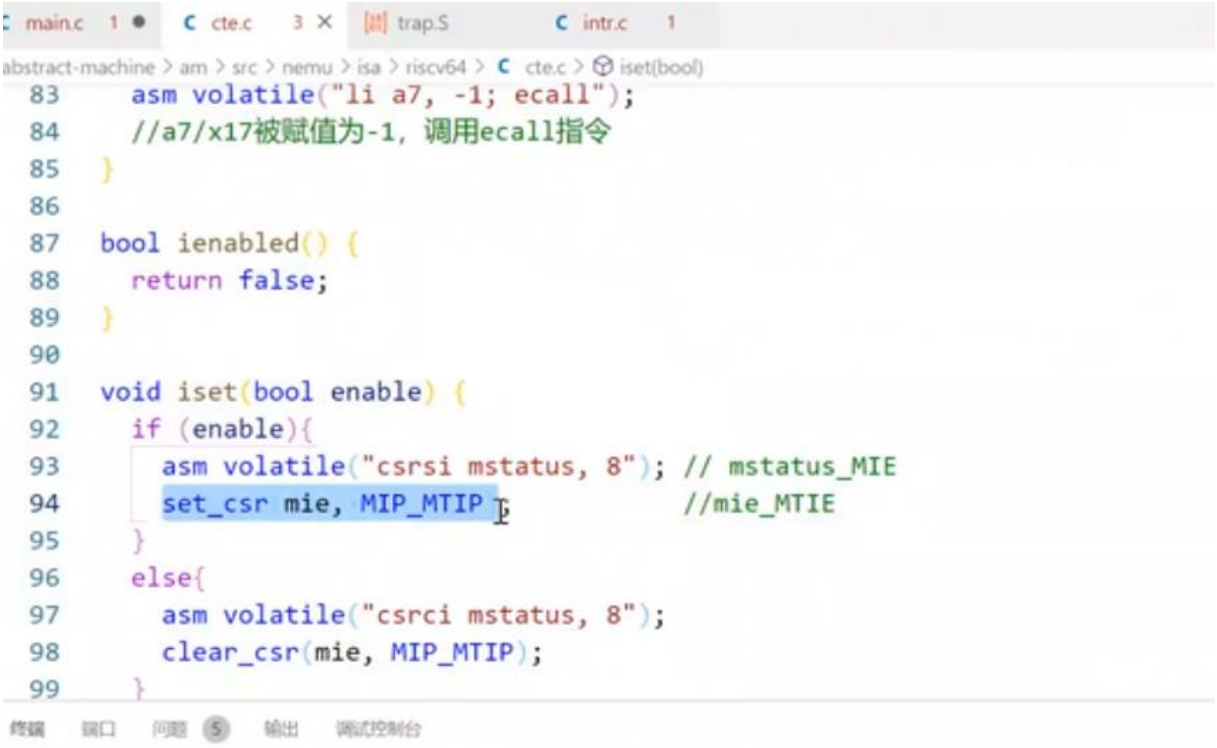
1. 一开始使用 alu_res[32-i] 来进行操作，但是在循环中好像直接对某一位的 i 赋值不可取，vivado 最后综合出来都是 x
2. 后来改用状态机，但是结果不对，在状态机中 A <= A-B，A<=A<<1,因为是并行，无法考虑当前 A 的值替换结果后再左移的操作，左移是对上一个时钟 A 左移的结果，由于时序一直差一个时钟，之前考虑保留当前的 A 在下一个时钟左移，但是在运算后左移的 A 好像会覆盖掉运算的 A 值，或许可以尝试 next_A <= A-B; A <= next_A<< 1'b1;
3. 由于状态机时序搞得太累，上网查了一下别人除法器的写法，发现例化模块之后一切都很简单，具体见代码，只需要在初始化和 generatefor 循环中例化就可以了

异常中断

进入中断-硬件

- CPU捕获计时器中断
- 保存/更新CSRs
  - mepc
    - 中断：下一条指令的PC值
  - mcause
    - 记录当前的异常原因：0x8000000000000007
  - mstatus
    - MPIE设为MIE的值，MIE设为0
- 跳转到mtvec的异常入口地址

中断软件置位



直接在 id 阶段进行中断？

进入中断-硬件

University of Chinese Academy of S

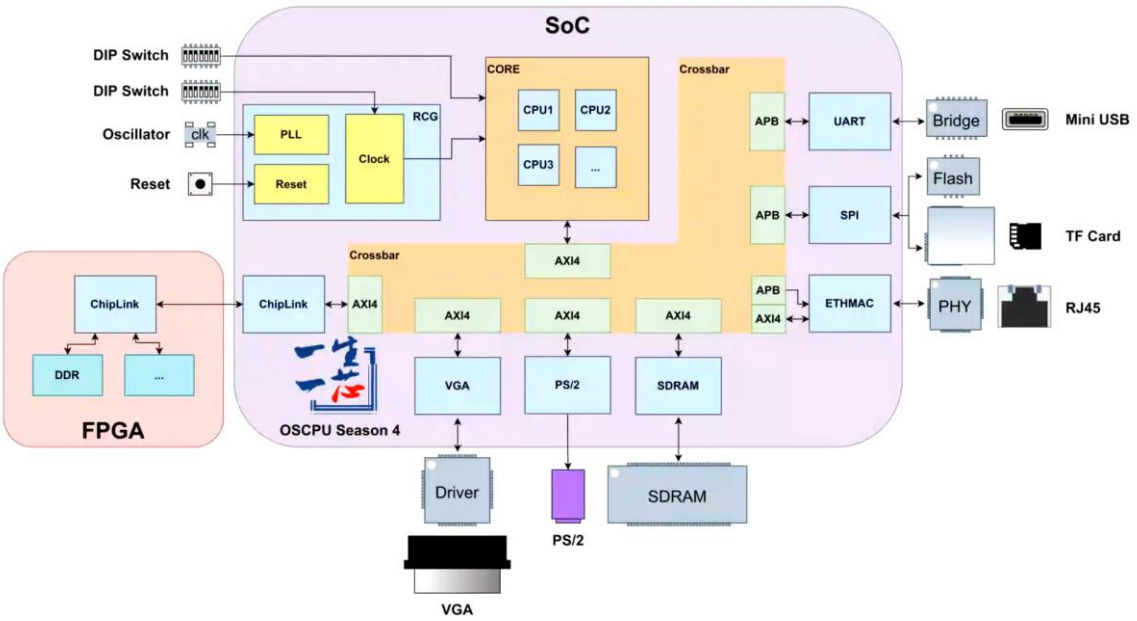
- CPU捕获计时器中断
- 保存/更新CSRs
  - mepc
    - 中断：下一条指令的PC值
  - mcause
    - 记录当前的异常原因：0x8000000000000007
  - mstatus
    - MPIE设为MIE的值，MIE设为0
- 跳转到mtvec的异常入口地址

接入外设

abstract-machine > scripts > M riscv64-npc.mk

```
include $(AM_HOME)/scripts/isa/riscv64.mk

AM_SRCS := riscv/npc/start.S \
           riscv/npc/trm.c \
           riscv/npc/ioe.c \
           riscv/npc/timer.c \
           riscv/npc/input.c \
           riscv/npc/cte.c \
           riscv/npc/trap.S \
           platform/dummy/vme.c \
           platform/dummy/mpe.c \
           riscv/npc/gpu.c \
           riscv/npc/disk.c


CFLAGS    += -fdata-sections -ffunction-sections
LDFLAGS   += -T $(AM_HOME)/scripts/linker.ld --defsym=_pmem_start=0x8000
```

SOC


SoC
DIP Switch
DIP Switch
Oscillator
clk
Reset
PLL
Reset
Clock
RCG
CORE
CPU1
CPU2
CPU3
...
Crossbar
APB
UART
Bridge
Mini USB
Flash
APB
SPI
TF Card
Crossbar
AXI4
APB
AXI4
ETHMAC
PHY
RJ45
ChipLink
ChipLink
AXI4
AXI4
AXI4
AXI4
DDR
...
FPGA
OSCPU Season 4
VGA
PS/2
SDRAM
Driver
PS/2
SDRAM
VGA