

实验报告

黄文稼 211240032

October 19, 2022

实验进度

我完成了全部内容

1. cpu-tests 以及其他 c 程序所需指令
2. string.c 优化以及 printf 和 sprintf 实现
3. itrace、iringbuf、mtrace、ftrace、dtrace 简单实现
4. 时钟、键盘、vga

感想

1. 在实现指令初期，顺手将测试指令以外其他相似指令一并实现，同时未标注，导致在移位指令出错时无法高效定位错误位置，盲目排错。
2. 最初实现移位指令时，错将 0b11111 值写作 63 导致 bug。进行按位操作时尽量使用二进制或十六进制。
3. 由于系统原因导致指令识别速度过慢，在怀疑 gcc 对指令识别部分存在优化时仍不检查优化后代码，仍然盲目进行代码优化，浪费时间。

必答题

YEMU 的执行过程

使用 $(R[0], R[1], M[5], M[6], M[7])$ 表示。则可以表示如下
 $(33, 0, 0, 16, 33, 0) \rightarrow (33, 33, 0, 16, 33, 0) \rightarrow (16, 33, 0, 16, 33, 0) \rightarrow (49, 33, 0, 16, 33, 0) \rightarrow$
 $(49, 33, 0, 16, 33, 49) \rightarrow (98, 33, 0, 16, 33, 49)$

NEMU 的指令执行过程

1. 输入镜像文件路径
2. `monitor.c` 中读入镜像文件
3. `cpu_exec` \rightarrow `execute`
4. `execute` 按 `n` 调用 `exec_once` 执行指令
5. `exec_once` 初始化 `Decode s`, 调用 `isa_exec_once`
6. `isa_exec_once` 使用 `inst_fetch` 获取指令 `value`, 并更新 `snpc`
7. `isa_exec_once` 调用 `decode_exec`
8. `decode_exec` 匹配指令, 匹配所用的 `key`, `mask` 和 `shift` 由 `gcc` 优化为常数
9. `decode_operand` 从指令 `value` 中读出目标寄存器、源寄存器与立即数
10. 通过 `__VA_ARGS__` 执行指令, 将 `$0` 寄存器归零
11. 返回至 `exec_once` 使用 `dnpc` 更新 `pc`

Typing-Game 命中效果

程序

1. 帧更新内, 先通过AM获取键盘输入, 调用`check_hit`判断命中
2. `check_hit` 中寻找与输入匹配的字符, 将最后找到的速度`v`取负
3. 在 `render`中通过AM更新画面, 速度为负的字符采用GREEN图案渲染
4. 再次更新帧, 调用`game_logic_update`进行字符位置更新, 命中的字符上移

AM

1. 访问指定NEMU为device准备的内存位置, 读写内容

ISA

1. 程序, AM与运行环境均被ISA 翻译为相应的指令等待NEMU执行

NEMU

1. 读取ISA翻译提供的指令, 依次执行。

编译与链接

static 与 inline

仅去除 inline 或 static 均无错误发生，二者均去除出现重定义错误。

去除 inline inline 是将函数的代码在调用处像 define 展开，因而仅去除 inline 无影响

去除 static 由于 inline 的存在，该函数不会被多次定义，但可能出现函数定义但未使用的警告

去除 static inline 由于头文件会被多次 include，会在链接时出现多重定义的错误。

volatile, static 与强弱符号

1. 35 个。在 nemu/build/obj-\$ISA-nemu-interpreter 下执行 `grep -r -c 'dummy'|grep '\.o.*:1'|wc -l`。注意此处""应当手动改为半角单引号。
2. 仍为 35 个，二者均未初始化被视为弱符号。
3. 多重定义。当两个均初始化后，被视为强符号，强符号不能被多次定义。

了解 Makefile

工作方式 读取参数，构造编译用的参数，执行系统命令进行操作

编译链接过程 通过依次将目标目录下的.c 文件 (hello.c)，依赖库 (AM、klib) 编译为.o 与.a 文件，再将二者链接成.elf 文件，再利用 objdump 生成反汇编文件，用 objcopy 生成 nemu 读取执行的.bin 文件