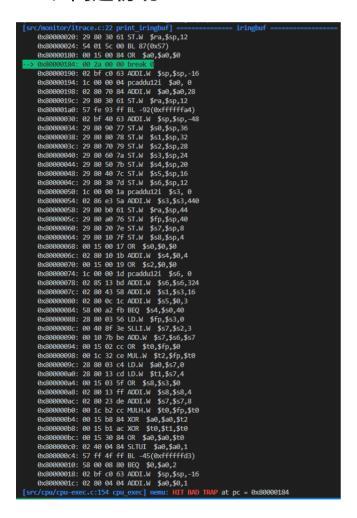
## la32-QEMU bug分析报告

### 前言

在使用 **qemu-system-loongarch32** 作为 nemu 差分测试的 Ref 时,发现LA32-qemu的 mulh.w 指令行为疑似存在问题,以下为该结论的详细分析过程。

## 一、nemu-la32r的 cpu-tests 测试

#### 1.1、问题初现



本组队员在做指令功能测试的过程中发现,nemu 运行测试程序 mul-longlong 的时候,在没有触发 difftest报错信息的情况下,在ebreak处异常退出。

因为是 32位指令集 测试的 longlong,第一反应这个 bug应该是由于一个寄存器没法完整的存储一个64位 数,所以在分开计算、存储时出现了问题。

这个问题比较的常见。指令实现过程中,处理进行各种类型的隐式转换时,很容易出现这个bug。

#### 2.1、difftest "消失"的疑云

Ebreak 这条调试指令在有两个作用:

- 1. 程序运行结束的终止符。
- 2. 返回 "机器" nemu的运行状态。(非0为异常)

在没有开启difftest的情况下,测试程序若执行到ebreak指令并触发 HIT BAD TRAP ,意味着前面的指令实现存在问题,触发了 am运行时环境的 halt(1)。(如下图)

```
attribute ((noinline))
       void check(bool cond) {
       if (!cond) halt(1);
 10
 11
 12
       #endif
 13
            ×
C trm.c
abstract-machine > am > src > platform > nemu > 🕒 trm.c > 🕎 halt(int)
       void putch(char ch) {
       void halt(int code) {
 17
         nemu_trap(code);
         // should not reach here
 20
         while (1);
 21
 22
```

这种情况下,difftest通常作为强有力的调试神器,在进入到halt之前,本应该能成功对比出了 Dut 和 Ref 的状态差异,从而输出报错信息。

但这里difftest并没有报错,证明作为 Ref 的 la32-QEMU到目前为止的寄存器状态和作为Dut 的nemu 是一致的。

因la32-qemu是由官方提供,故此时还并没有怀疑是qemu的问题,只当是我们组的工程里还有哪里的bug并被没有发现。

## 二、mul-longlong源码及汇编分析

既然调试工具失效,那么只能用最为朴素的方法——分析测试程序源码及汇编代码,人工验证指令行为 及机器状态。

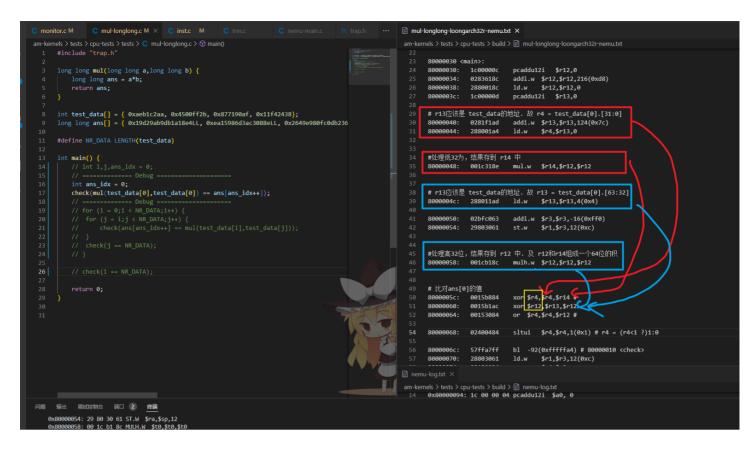
#### 2.1、C源码

从源码可知,mul-longlong的测试逻辑就是通过循环比对 test\_data的乘积 和 ans 中的值,来验证 cpu 处理 长整型的乘法是否存在问题。

```
am-kernels > tests > cpu-tests > tests > € mul-longlong.c > ♦ main()
       long long mul(long long a, long long b) {
          long long ans = a*b;
           return ans;
       int test_data[] = { 0xaeb1c2aa, 0x4500ff2b, 0x877190af, 0x11f42438};
       long long ans[] = { 0x19d29ab9db1a18e4LL, 0xea15986d3ac3088eLL, 0x2649e980fc0db236LL, 0xfa4c43da0a4a7d30LL, 0x1299898e2c56b139L
       #define NR_DATA LENGTH(test_data)
       int main() {
           int i,j,ans_idx = 0;
           // ===== Debug
           for (i = 0;i < NR_DATA;i++) {
                for (i = i:i < NR DATA:i++)
                  check(ans[ans_idx++] == mul(test_data[i],test_data[j]));
                check(j == NR_DATA);
           check(i == NF_DATA);
           return 0;
 29
                                                               C trm.c
am-kernels > tests > cpu-tests > include > h trap.h > 😯 check(bool)
                                                                abstract-machine > am > src > platform > nemu > € trm.c > ♦ putch(char)
       #ifndef __TRAP_H_
#define __TRAP_H_
                                                                       #define MAINARGS
                                                                       static const char mainargs[] = MAINARGS;
       #include <ar.h>
#include <krib.h>
                                                                        void putch(char ch) {
       #include <k lib-macros.h>
                                                                        outb(SERIAL_PORT, ch);
       __attribute__((noinline))
void check(bool cond) {
                                                                        void halt(int code) {
 10
      if (!cond) halt(1);
                                                                         nemu_trap(code);
                                                                         oid trm init()
```

#### 2.2、汇编源码分析

为了降低分析难度,对mul-longlong进行了一定修改。c源码及汇编代码、汇编分析如下图:



由上图的汇编代码,可以做出以下程序行为预判:

- 低32位,即: r4 和 r14的值,在执行xor指令前,应该都是 db1a18e4。
- 高32位,即: r13 和 r12的值,在执行xor指令前,应该都是19d29ab9。
- 当测试程序运行到上图中右边汇编代码第54行时,正确的运行行为应该是: r4 = (0<1?)1:0;

#### 2.3、nemu & qemu 比对

在以上预判的基础上,在mul.w、mulh.w、xor、sltui 处插入监视点,观测指令行为,可以得出以下结果:

# 2.3.1、符合人工分析预期的mulh.w实现:

可以看到下图中,mulh.w执行后,dut的t0寄存器 (即:r12寄存器)的值符合。但触发了difftest报错 ——与Ref的寄存器状态不同。

#### 2.3.2、符合difftest行为的实现:

下图则为符合difftest的预期,但违背了测试 程序的预期情况。

```
Description of the following process of the first of the second process of the first of the first of the second process of the first of the first
```

```
| Sec/spidiffeet/dat.com init_diffeet) Differential testing: ON | Sec/spidiffeet/dat.com init_diffeets) in result of easy instruction will be compared with /home/yeys/Desktop/perj241-FA_for_IA/mems/ | Sec/spidiffeet/dat.com init_diffeets) in result of easy instruction will be compared with /home/yeys/Desktop/perj241-FA_for_IA/mems/ | Images |
```

## 四、结论及处理

以上信息应该可以充分证明 **qemu-system-loongarch32** 处下载的qemu,mulh.w指令的实现存在问题(debug和release两个版本都试过了)。

疑惑:但假设该qemu存在问题,为什么Linux启动会没有问题???

为了不影响项目推进,我们团队目前的做法是,暂时把difftest关掉并尝试阅读 la32r-QEMU源码,试图修正问题。