2024-4-7 分享会

1. mini-rv32ima模拟器
2. 构建镜像文件
3. 升级nemu/npc踩坑点

mini-rv32ima模拟器

github地址 https://github.com/cnlohr/mini-rv32ima

一个非常简短的riscv32模拟器实现，(去除不重要的内容，大概700行左右)，可以启动linux，很快就能看完

仓库除了提供了模拟器，还提供了linux-nommu的buildroot构建脚本，一些示例应用等等

通过阅读这个模拟器，可以了解到运行linux-nommu的最低要求

linux-nommu最低要求

相较于ysyx要求的nemu/npc，还需要完成下面的功能

- 实现M拓展
- 实现A拓展
- 实现完整Zicsr拓展
- 实现时钟中断
- M和U两个特权状态(不需要特权保护)

还有一些指令需要实现

- Zifence指令拓展
- WFI指令等

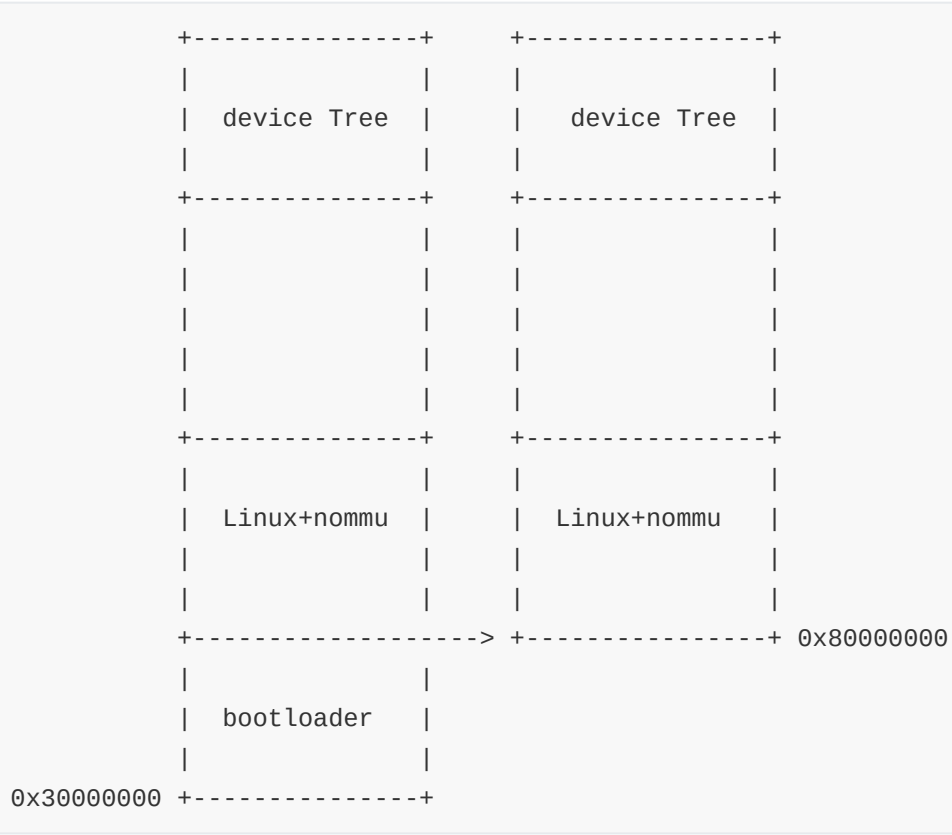
但是对于ysyx的nemu/npc而言，全部当作`nop`指令即可

构建可供nemu/npc运行的linux镜像

mini-rv32ima仓库提供了buildroot构建脚本，可以方便的构建出linux镜像

但是想要用于nemu/npc运行，还要添加一个bootloader和设备树文件

我构建的linux镜像如下



构建可供nemu/npc运行的linux镜像

除此之外，还需要修改设备树文件，将串口设备和CLINT设备地址改为`ysyx-soc`的地址

```
        clint@2000000 {
            interrupts-extended = <0x02 0x03 0x02 0x07>;
            reg = <0x00 0x02000000 0x00 0x10000>;
            compatible = "sifive,clint0\0riscv,clint0";
        };

        uart@10000000 {
            clock-frequency = <0x1000000>;
            reg = <0x00 0x10000000 0x00 0x100>;
            compatible = "ns16850";
        };
```

同时在`bootloader`运行结束之时，将寄存器`x10`,`x11`进行设置(核心数和设备树地址)

构建可供nemu/npc运行的linux镜像

除了`mini-rv32ima`提供的一些二进制程序，也可以很方便的添加自己想要添加的c语言程序

`mini-rv32ima`仓库当中的`hello_linux`目录下提供了一些示例

```c
#include <stdio.h>

int main( int argc, char ** argv ) {
    printf( "Hello, world %08x\n", argc );
    float ft = 7.3f;                    // 甚至有浮点数，虽然我不知道是怎么做到的
    printf( "f: %f\n", ft );
}
```

构建可供nemu/npc运行的linux镜像

这个时候，就可以从flash当中加载并运行linux-nommu了

```
[    0.000000] printk: legacy bootconsole [uart8250] enabled
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x0000000080000000-0x00000000809fffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080000000-0x00000000809fffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080000000-0x00000000809fffff]
[    0.000000] riscv: base ISA extensions
[    0.000000] riscv: ELF capabilities
[    0.000000] Kernel command line: earlycon=uart8250,mmio,0x10000000,1000000 console=ttyS0
[    0.000000] Dentry cache hash table entries: 2048 (order: 1, 8192 bytes, linear)
[    0.000000] Inode-cache hash table entries: 1024 (order: 0, 4096 bytes, linear)
[    0.000000] Built 1 zonelists, mobility grouping off.  Total pages: 2540
[    0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[    0.000000] Memory: 7516K/10240K available (1253K kernel code, 276K rwdata, 136K rodata, 802K init, 92K bss, 2724K reserved, 0K cma-reserved)
[    0.000000] SLUB: HWalign=64, Order=0-1, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] NR_IRQS: 64, nr_irqs: 64, preallocated irqs: 0
[    0.000000] riscv-intc: 32 local interrupts mapped
[    0.000000] clint: clint@2000000: timer running at 1000000 Hz
[    0.000000] clocksource: clint_clocksource: mask: 0xffffffffffffffff max_cycles: 0x1d854df40, max_idle_ns: 3526361616960 ns
[    0.000393] sched_clock: 64 bits at 1000kHz, resolution 1000ns, wraps every 2199023255500ns
[    0.111564] Console: colour dummy device 80x25
[    0.134823] Calibrating delay loop (skipped), value calculated using timer frequency.. 2.00 BogoMIPS (lpj=10000)
[    0.160352] pid_max: default: 4096 minimum: 301
[    0.201452] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes, linear)
[    0.224705] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes, linear)
[    0.908890] devtmpfs: initialized
[    1.463267] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[    1.489712] futex hash table entries: 16 (order: -5, 192 bytes, linear)
[    1.837922] cpu0: Ratio of byte access time to unaligned word access is 4.56, unaligned accesses are fast
[    2.588204] clocksource: Switched to clocksource clint_clocksource
[    6.895627] workingset: timestamp_bits=30 max_order=11 bucket_order=0
[   22.091129] Serial: 8250/16550 driver, 1 ports, IRQ sharing disabled
[   22.832230] printk: legacy console [ttyS0] disabled
[   23.040073] 10000000.uart: ttyS0 at MMIO 0x10000000 (irq = 0, base_baud = 1048576) is a XR16850
m[   23.075084] printk: legacy console [ttyS0] enabled
[   23.075084] printk: legacy console [ttyS0] enabled
[   23.106100] printk: legacy bootconsole [uart8250] disabled
[   23.106100] printk: legacy bootconsole [uart8250] disabled
[   25.669888] clk: Disabling unused clocks
``m[   28.843914] Freeing unused kernel image (initmem) memory: 800K
[   28.859580] This architecture does not have kernel memory protection.
[   28.874627] Run /init as init process
Welcome to mini-rv32ima Linux
Jan  1 00:00:40 login[22]: root login on 'console'
~ # ls
coremark     hello_linux  pi
~ # ./hello_linux
Hello, world 00000001
f: 7.300000
~ #
```

将mini-rv32ima改造来作为difftest

mini-rv32ima的代码非常短，很快便可以将其改造并用于difftest

改造后，可以用于nemu/npc的调试，来相对方便的找出bug

这一部分可以参考nemu来实现

```
uint8_t flash[flash_end - flash_begin];
uint8_t ram[MINI_RV32_RAM_SIZE];
struct MiniRV32IMAState core;

__EXPORT void difftest_memcpy(uint32_t addr, void *buf, uint32_t n,
                              bool direction) {}

__EXPORT void difftest_regcpy(void *dut, bool direction) {}

__EXPORT void difftest_exec(uint64_t n) { MiniRV32IMAStep(&core, ram, 0, n); }

__EXPORT void difftest_raise_intr(uint64_t NO) { MiniRV32IMAStep(&core, ram, NO, 1); }

__EXPORT void difftest_init(int port) {}
```

升级nemu/npc时的踩坑点

M指令拓展

注意除法指令和取模指令的除零和溢出情况的特殊处理

```
    INSTPAT("0000001 ????? ????? 000 ????? 01100 11", mul, R, R(rd) = ((int64_t)(sword_t)src1 *
(int64_t)(sword_t)src2));
    INSTPAT("0000001 ????? ????? 001 ????? 01100 11", mulh, R, R(rd) = ((int64_t)(sword_t)src1 *
(int64_t)(sword_t)src2) >> 32);
    INSTPAT("0000001 ????? ????? 010 ????? 01100 11", mulsu, R, R(rd) = ((int64_t)(sword_t)src1 *
(uint64_t)src2) >> 32);
    INSTPAT("0000001 ????? ????? 011 ????? 01100 11", mulu, R, R(rd) = ((uint64_t)src1 *
(uint64_t)src2) >> 32);
    INSTPAT("0000001 ????? ????? 100 ????? 01100 11", div, R, {
        if(src2 == 0) R(rd) = -1;
        else R(rd) = ((int32_t)src1 == INT32_MIN && (int32_t)src2 == -1) ? src1 : ((int32_t)src1 /
(int32_t)src2);
    });
    INSTPAT("0000001 ????? ????? 101 ????? 01100 11", divu, R, {
        R(rd) = (src2 == 0) ? 0xffffffff : src1 / src2;
    });
    INSTPAT("0000001 ????? ????? 110 ????? 01100 11", rem, R, {
        if(src2 == 0) R(rd) = src1;
        else R(rd) = ((int32_t)src1 == INT32_MIN && (int32_t)src2 == -1) ? 0 : ((uint32_t)
((int32_t)src1 % (int32_t)src2));
    });
    INSTPAT("0000001 ????? ????? 111 ????? 01100 11", remu, R, {
        R(rd) = (src2 == 0) ? src1 : src1 % src2;
    });
```

升级nemu/npc时的踩坑点

串口的CLINT外设

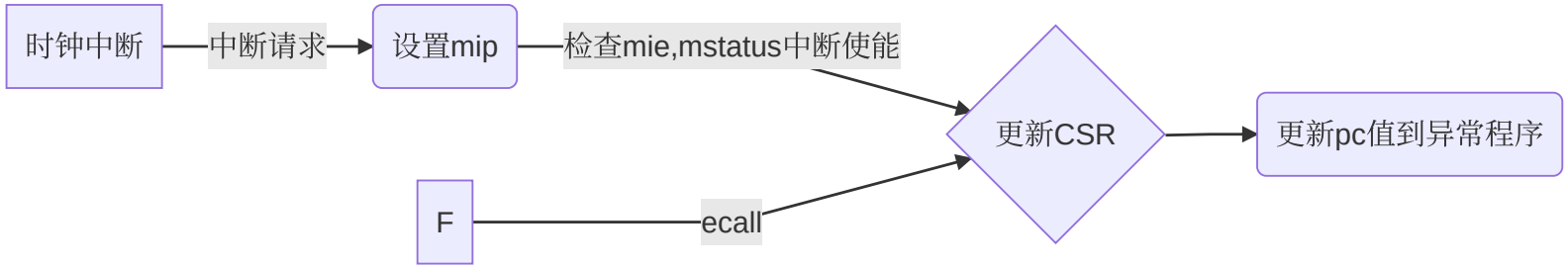
串口和CLINT的实现上可以参考mini-rv32ima模拟器的实现

~~当时因为串口的一个状态寄存器实现不正确导致bug调了4天~~

升级nemu/npc时的踩坑点

中断功能的实现

中断实现还是比较简单的

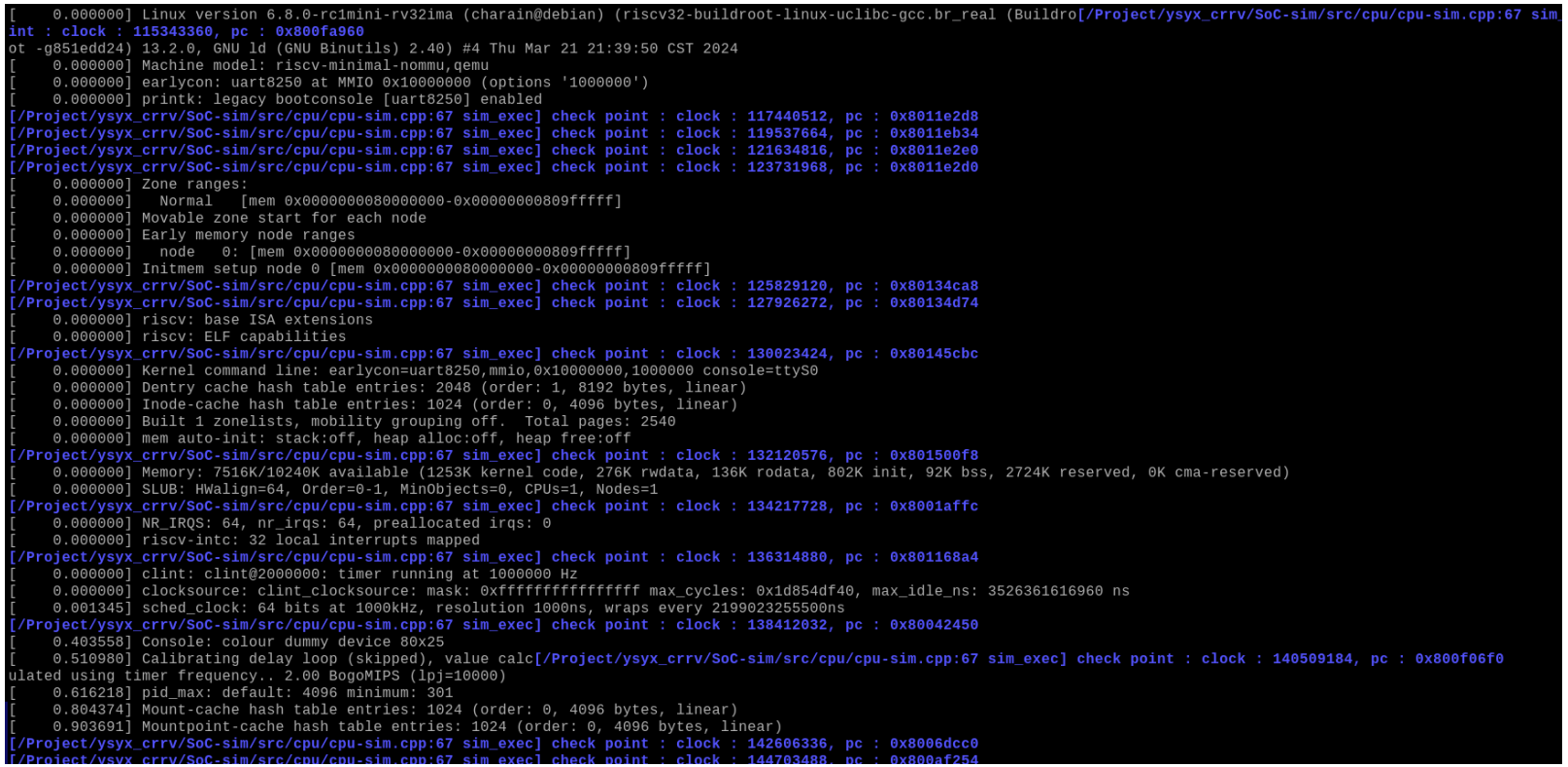


但是中断部分的difftest不太好做，尤其是npc这种

~~查到今天还在修npc的中断difftest bug~~

npc启动linux

很可惜花了一周多的时间，npc仍旧没有启动linux，(中断部分的difftest比较难处理)



```
[    4.260794] devtmpfs: initialized
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 159383552, pc : 0x800b7b84
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 161480704, pc : 0x800b8450
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 163577856, pc : 0x8011b200
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 165675008, pc : 0x8004bef8
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 167772160, pc : 0x80134abc
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 169869312, pc : 0x80027868
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 171966464, pc : 0x801227d0
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 174063616, pc : 0x80134d00
[    7.556728] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[    7.656314] futex hash table entries: 16 (order: -5, 19[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check po
2 bytes, linear)
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 178257920, pc : 0x800b7afc
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 180355072, pc : 0x800cbe48
[/Project/ysyx_crrv/SoC-sim/src/cpu/cpu-sim.cpp:67 sim_exec] check point : clock : 182452224, pc : 0x80134ac4
difftest fail at mstatus,NPC: 0x1888, REF: 0x1880

[/Project/ysyx_crrv/SoC-sim/src/cpu/difftest.cpp:95 difftest_checkregs] NPC reg:
pc : 0x80002434
x0 :$0    : 0x0          0            | x1 :ra    : 0x80002388   -2147474552
x2 :sp    : 0x80417e40   -2143191488  | x3 :gp    : 0x8026a710   -2144950512
x4 :tp    : 0x80414000   -2143207424  | x5 :t0    : 0x0          0
x6 :t1    : 0x0          0            | x7 :t2    : 0x0          0
x8 :s0    : 0x80417e90   -2143191408  | x9 :s1    : 0x80438001   -2143059967
x10:a0    : 0x80439f81   -2143051903  | x11:a1    : 0x8043bf83   -2143043709
x12:a2    : 0x1f80       8064         | x13:a3    : 0x8043bf83   -2143043709
x14:a4    : 0xffffffff   -1           | x15:a5    : 0xffff8c78   -29576
x16:a6    : 0x0          0            | x17:a7    : 0x0          0
x18:s2    : 0xcc0        3264         | x19:s3    : 0x8043a003   -2143051773
x20:s4    : 0x8026b000   -2144948224  | x21:s5    : 0x3          3
x22:s6    : 0x80224e28   -2145235416  | x23:s7    : 0x80225000   -2145234944
x24:s8    : 0xffff8c78   -29576       | x25:s9    : 0x8          8
x26:s10   : 0x8013a000   -2146197504  | x27:s11   : 0x0          0
x28:t3    : 0x0          0            | x29:t4    : 0x0          0
x30:t5    : 0x22         34           | x31:t6    : 0x80265144   -2144972476
[/Project/ysyx_crrv/SoC-sim/src/cpu/difftest.cpp:98 difftest_checkregs] REF reg:
pc : 0x8013a200
x0 :$0    : 0x0          0            | x1 :ra    : 0x80002388   -2147474552
x2 :sp    : 0x80417e40   -2143191488  | x3 :gp    : 0x8026a710   -2144950512
x4 :tp    : 0x80414000   -2143207424  | x5 :t0    : 0x0          0
x6 :t1    : 0x0          0            | x7 :t2    : 0x0          0
x8 :s0    : 0x80417e90   -2143191408  | x9 :s1    : 0x80438001   -2143059967
x10:a0    : 0x80439f81   -2143051903  | x11:a1    : 0x8043bf83   -2143043709
x12:a2    : 0x1f80       8064         | x13:a3    : 0x8043bf83   -2143043709
x14:a4    : 0xffffffff   -1           | x15:a5    : 0xffff8c78   -29576
x16:a6    : 0x0          0            | x17:a7    : 0x0          0
x18:s2    : 0xcc0        3264         | x19:s3    : 0x8043a003   -2143051773
```