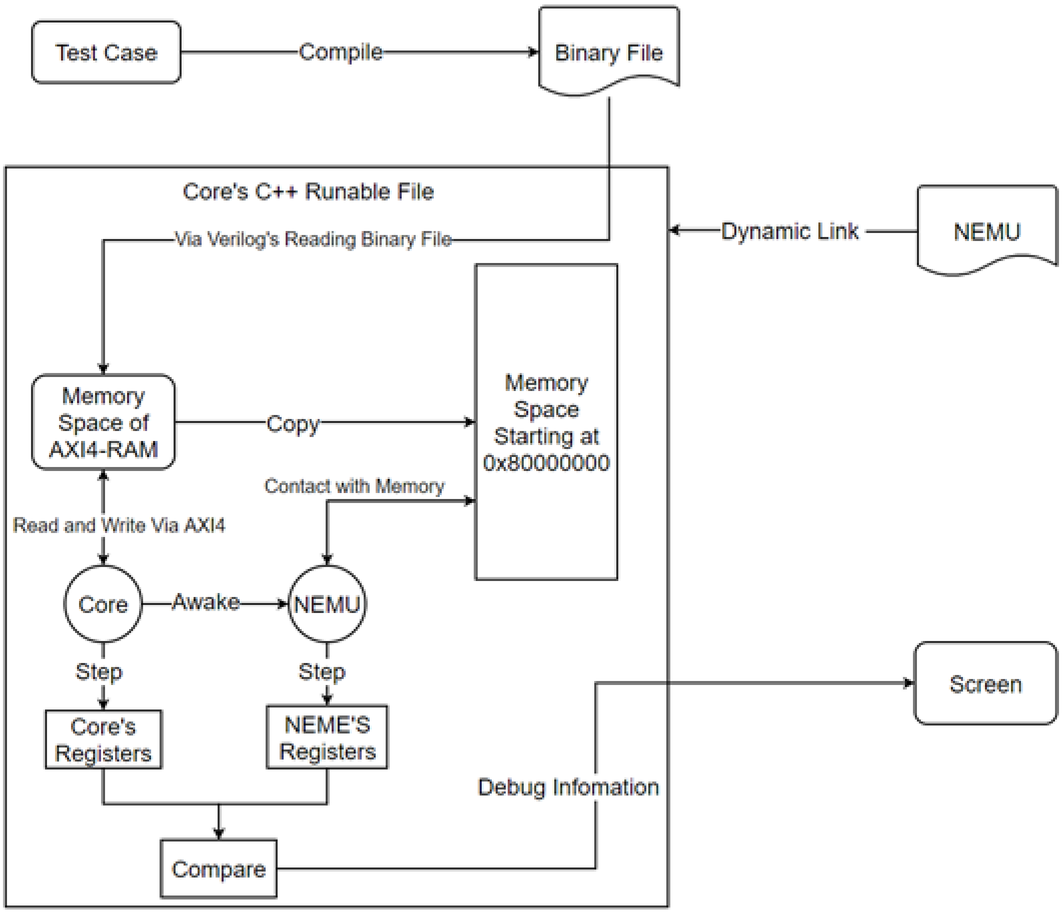
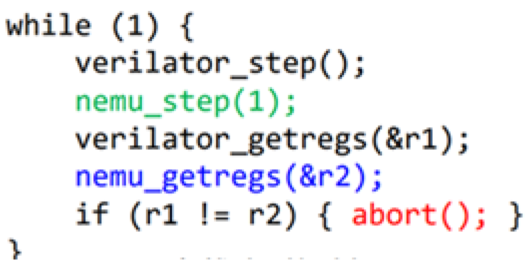
Functional testing and analysis

1.Test framework

The DiffTest framework was used as the test framework.The framework contains a simulator with correct behavior :NEMU, the processor core to be tested: NutShell, the simulated peripheral devices: SDRAM, UART, etc. After NutShell executes an instruction, NutShell will wake up NEMU to execute an instruction, and compare the states of the two, the state continues consistent, the inconsistency error, and the error field is printed.The schematic diagram is as follows:





```
while (1) {
    verilator_step();
    nemu_step(1);
    verilator_getregs(&r1);
    nemu_getregs(&r2);
    if (r1 != r2) { abort(); }
}
```

2.Method

- Before starting the test, an initial level 3 page table of SV39 (this page table translates the virtual address to the same physical address) is populated into the simulated RAM interacting with NutShell.
- Hard-coding set the initial position of the SATP register to point to the page table, and modify the condition for the start of address conversion, so that it can be turned on when it

is in M mode. After that, NutShell can directly perform address translation and access the page table when accessing the memory, without requiring the test program to maintain an additional page table, which reduces the difficulty of testing.As long as the test program can be run correctly, the behavior of the MMU is correct.

- In order to add support for the Svnapot extension test, we need to make some modifications to the initial populated page table, change the N bits of the page table entry to 1, and change the PPN[0][3:0] to 1000, so that all the leaf node page table entries are the page table entries of the Svnapot extension. `src/test/csrc/ram.cpp` is the directory in which this emulated RAM file resides.

First add a parameter to the addPageSv39 function to choose whether or not to use the Svnapot extended PTE

```
void addpageSv39(bool napot_on) {...}
```

For Svnapot extended PTE, set N to 1, PPN[0][3:0] to 1000

```
for(int outidx = 0; outidx < PTEMMIONUM; outidx++) {
    for(int inidx = 0; inidx < ENTRYNUM; inidx++) {
      if(!napot_on)
        ptemmio[outidx][inidx] = (((0x40000000 + outidx*PTEVOLUME + inidx*PAGESIZE) & 0xfffff000) >> 2) | 0xf;
      else
        ptemmio[outidx][inidx] = ((((0x40000000 + outidx*PTEVOLUME + inidx*PAGESIZE) & 0xffff0000) | 0x00008000) >> 2) | 0xf | 0x8000000000000000;
    }
 }
for(int outidx = 0; outidx < PTENUM; outidx++ ) {
    for(int inidx = 0; inidx < ENTRYNUM; inidx++ ) {
      if(!napot_on)
        pte[outidx][inidx] = (((0x80000000 + outidx*PTEVOLUME + inidx*PAGESIZE) & 0xfffff000)>>2) | 0xf;
      else
        pte[outidx][inidx] = ((((0x80000000 + outidx*PTEVOLUME + inidx*PAGESIZE) & 0xffff0000) | 0x00008000) >> 2) | 0xf | 0x8000000000000000;
    }
}
```

Then turn on the Svnapot extension when calling this function

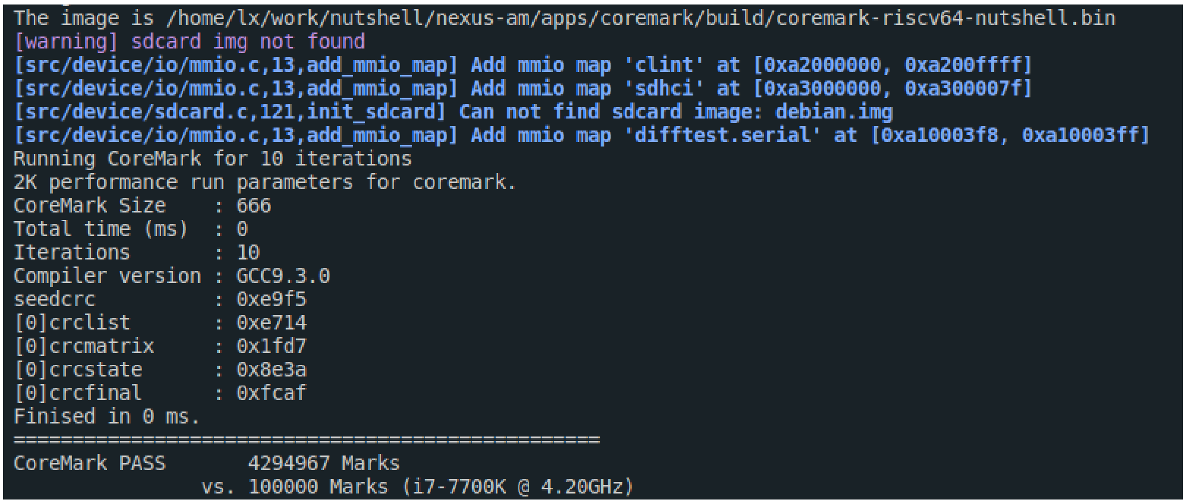
```
//new add
 addpageSv39(true);
//new end
```

3.Results

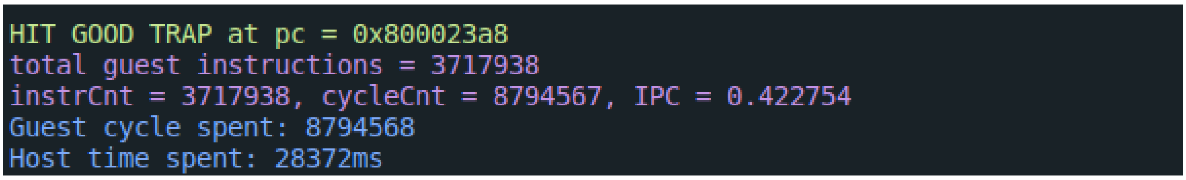
Note: All marks and times in the results are not trusted because the page table does not override the MMIO address of 0x3XXXXXXX, so I annotated the time-related functions in the test program.

3.1 CoreMark:

Svnapot enabled:



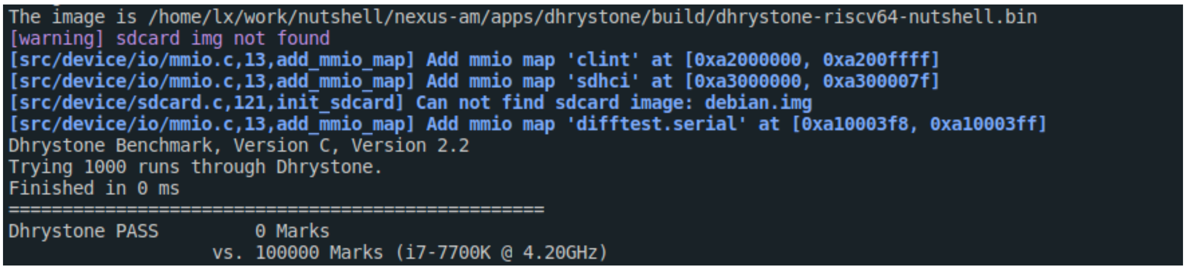
```
The image is /home/lx/work/nutshell/nexus-am/apps/coremark/build/coremark-riscv64-nutshell.bin
[warning] sdcard img not found
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'clint' at [0xa2000000, 0xa200ffff]
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'sdhci' at [0xa3000000, 0xa300007f]
[src/device/sdcard.c,121,init_sdcard] Can not find sdcard image: debian.img
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'difftest.serial' at [0xa10003f8, 0xa10003ff]
Running CoreMark for 10 iterations
2K performance run parameters for coremark.
CoreMark Size    : 666
Total time (ms)  : 0
Iterations       : 10
Compiler version : GCC9.3.0
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xfcaf
Finised in 0 ms.
==================================================
CoreMark PASS       4294967 Marks
                vs. 100000 Marks (i7-7700K @ 4.20GHz)
```



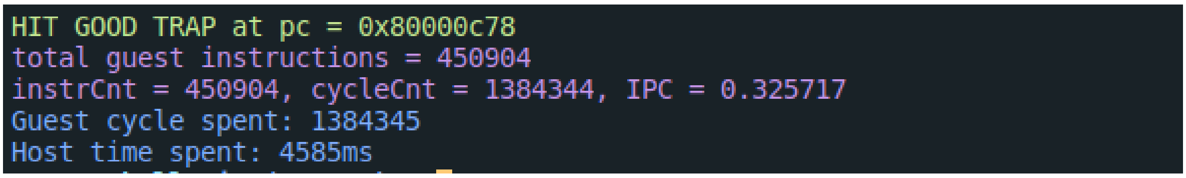
```
HIT GOOD TRAP at pc = 0x800023a8
total guest instructions = 3717938
instrCnt = 3717938, cycleCnt = 8794567, IPC = 0.422754
Guest cycle spent: 8794568
Host time spent: 28372ms
```

3.2 Dhrystone：

Svnapot enabled：



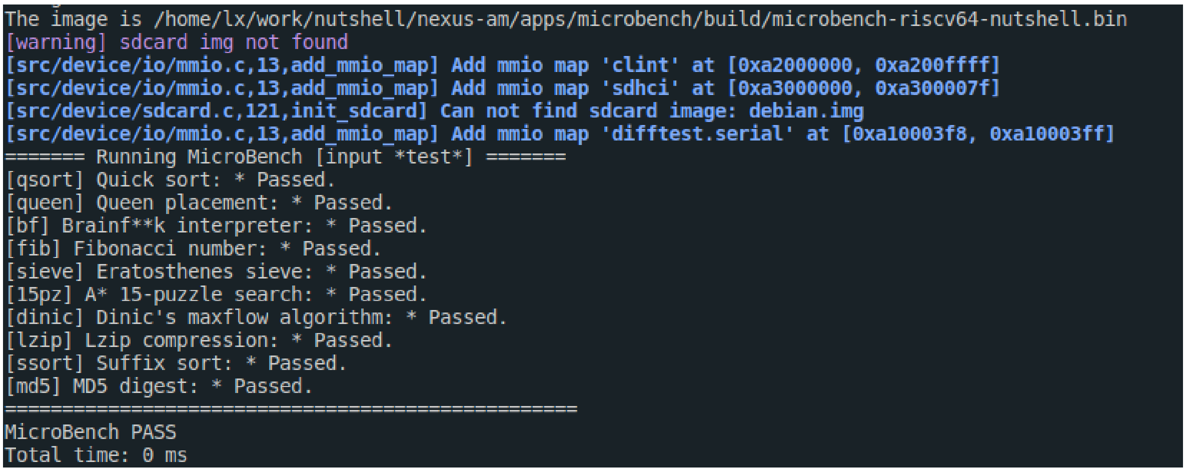
```
The image is /home/lx/work/nutshell/nexus-am/apps/dhrystone/build/dhrystone-riscv64-nutshell.bin
[warning] sdcard img not found
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'clint' at [0xa2000000, 0xa200ffff]
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'sdhci' at [0xa3000000, 0xa300007f]
[src/device/sdcard.c,121,init_sdcard] Can not find sdcard image: debian.img
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'difftest.serial' at [0xa10003f8, 0xa10003ff]
Dhrystone Benchmark, Version C, Version 2.2
Trying 1000 runs through Dhrystone.
Finished in 0 ms
==================================================
Dhrystone PASS         0 Marks
                 vs. 100000 Marks (i7-7700K @ 4.20GHz)
```



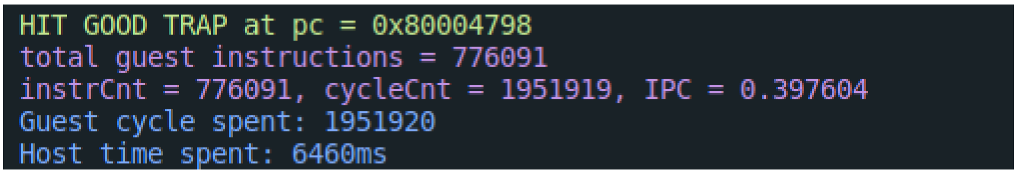
```
HIT GOOD TRAP at pc = 0x80000c78
total guest instructions = 450904
instrCnt = 450904, cycleCnt = 1384344, IPC = 0.325717
Guest cycle spent: 1384345
Host time spent: 4585ms
```

3.3 Microbench(test)：

Svnapot enabled：



```
The image is /home/lx/work/nutshell/nexus-am/apps/microbench/build/microbench-riscv64-nutshell.bin
[warning] sdcard img not found
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'clint' at [0xa2000000, 0xa200ffff]
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'sdhci' at [0xa3000000, 0xa300007f]
[src/device/sdcard.c,121,init_sdcard] Can not find sdcard image: debian.img
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'difftest.serial' at [0xa10003f8, 0xa10003ff]
======= Running MicroBench [input *test*] =======
[qsort] Quick sort: * Passed.
[queen] Queen placement: * Passed.
[bf] Brainf**k interpreter: * Passed.
[fib] Fibonacci number: * Passed.
[sieve] Eratosthenes sieve: * Passed.
[15pz] A* 15-puzzle search: * Passed.
[dinic] Dinic's maxflow algorithm: * Passed.
[lzip] Lzip compression: * Passed.
[ssort] Suffix sort: * Passed.
[md5] MD5 digest: * Passed.
==================================================
MicroBench PASS
Total time: 0 ms
```

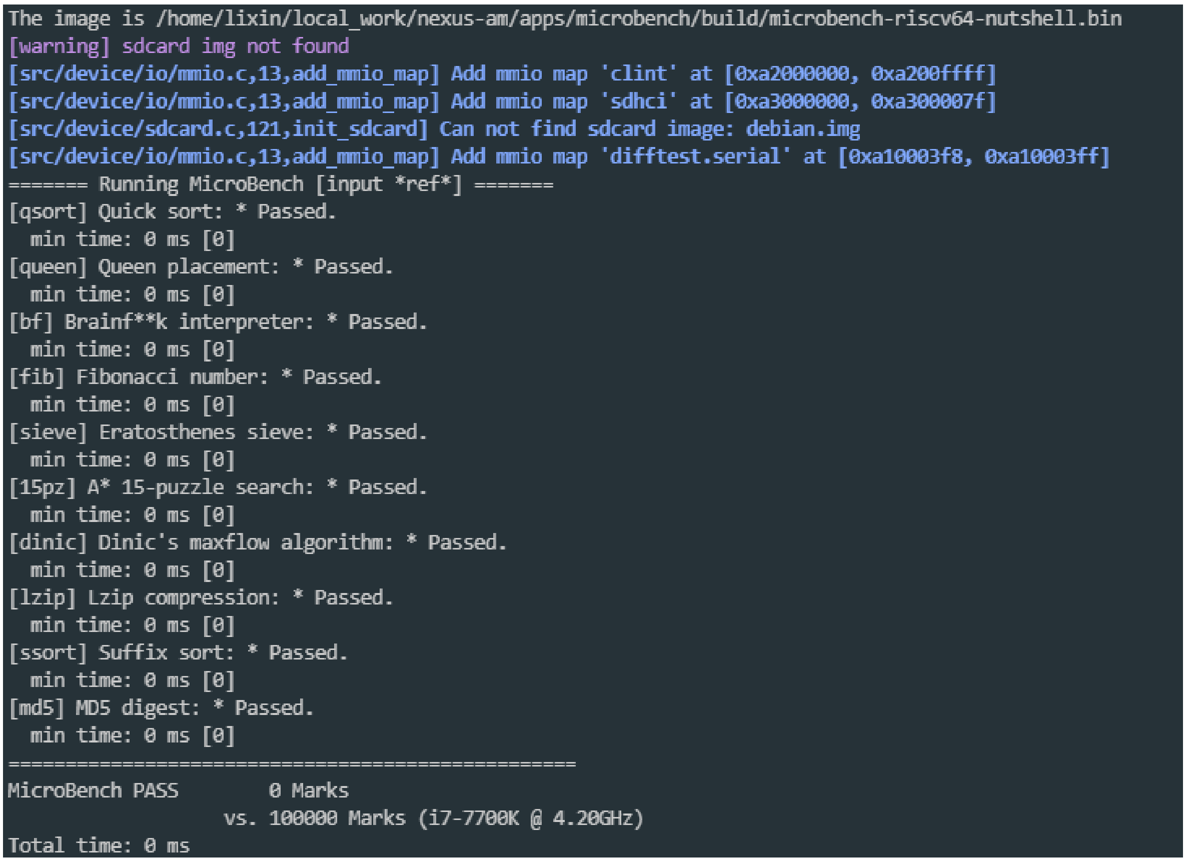


```
HIT GOOD TRAP at pc = 0x80004798
total guest instructions = 776091
instrCnt = 776091, cycleCnt = 1951919, IPC = 0.397604
Guest cycle spent: 1951920
Host time spent: 6460ms
```

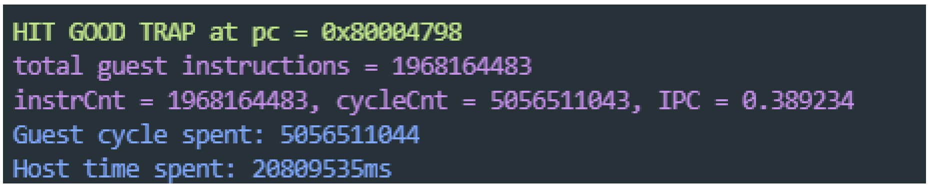
3.4 Microbench(ref):

This Microbench is more complex than the microbench(test)

Svnapot enabled:



```
The image is /home/lixin/local_work/nexus-am/apps/microbench/build/microbench-riscv64-nutshell.bin
[warning] sdcard img not found
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'clint' at [0xa2000000, 0xa200ffff]
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'sdhci' at [0xa3000000, 0xa300007f]
[src/device/sdcard.c,121,init_sdcard] Can not find sdcard image: debian.img
[src/device/io/mmio.c,13,add_mmio_map] Add mmio map 'difftest.serial' at [0xa10003f8, 0xa10003ff]
======= Running MicroBench [input *ref*] =======
[qsort] Quick sort: * Passed.
  min time: 0 ms [0]
[queen] Queen placement: * Passed.
  min time: 0 ms [0]
[bf] Brainf**k interpreter: * Passed.
  min time: 0 ms [0]
[fib] Fibonacci number: * Passed.
  min time: 0 ms [0]
[sieve] Eratosthenes sieve: * Passed.
  min time: 0 ms [0]
[15pz] A* 15-puzzle search: * Passed.
  min time: 0 ms [0]
[dinic] Dinic's maxflow algorithm: * Passed.
  min time: 0 ms [0]
[lzip] Lzip compression: * Passed.
  min time: 0 ms [0]
[ssort] Suffix sort: * Passed.
  min time: 0 ms [0]
[md5] MD5 digest: * Passed.
  min time: 0 ms [0]
==================================================
MicroBench PASS        0 Marks
                   vs. 100000 Marks (i7-7700K @ 4.20GHz)
Total time: 0 ms
```



```
HIT GOOD TRAP at pc = 0x80004798
total guest instructions = 1968164483
instrCnt = 1968164483, cycleCnt = 5056511043, IPC = 0.389234
Guest cycle spent: 5056511044
Host time spent: 20809535ms
```

4.Analysis

- In terms of function, Coremark Dhrystone and Microbench can run to the end correctly, indicating the correctness of function implementation.