Lab3 实验报告

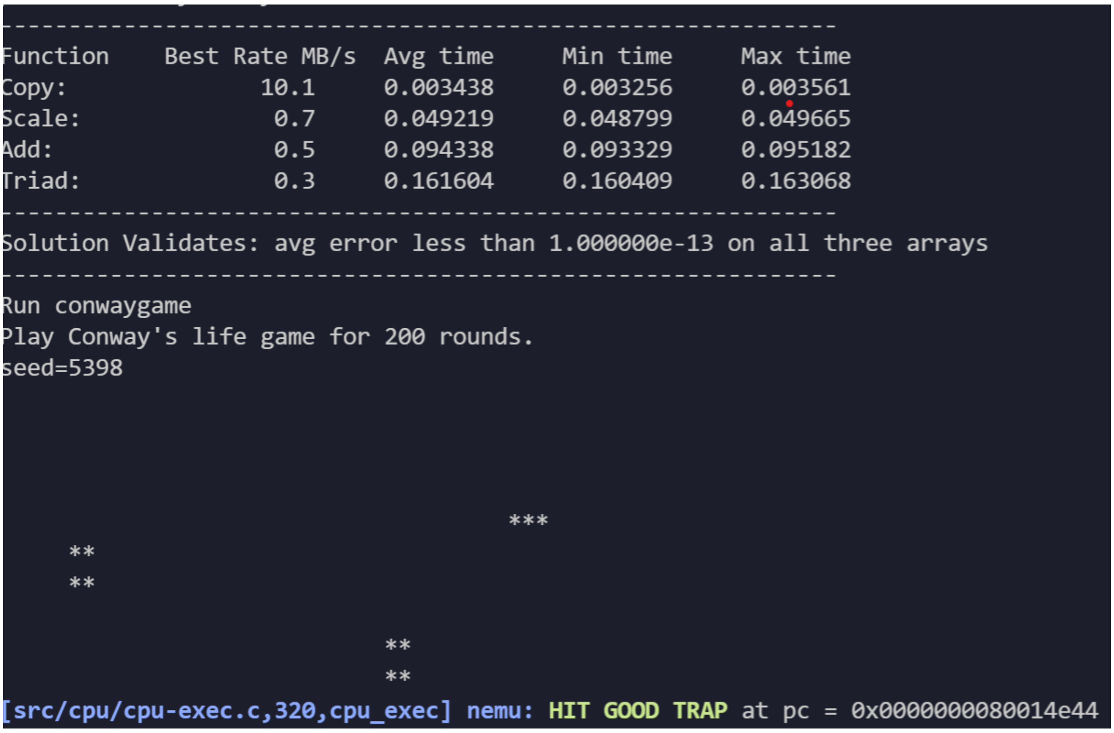
20307130013 黄栋豪

一．随机延迟的处理

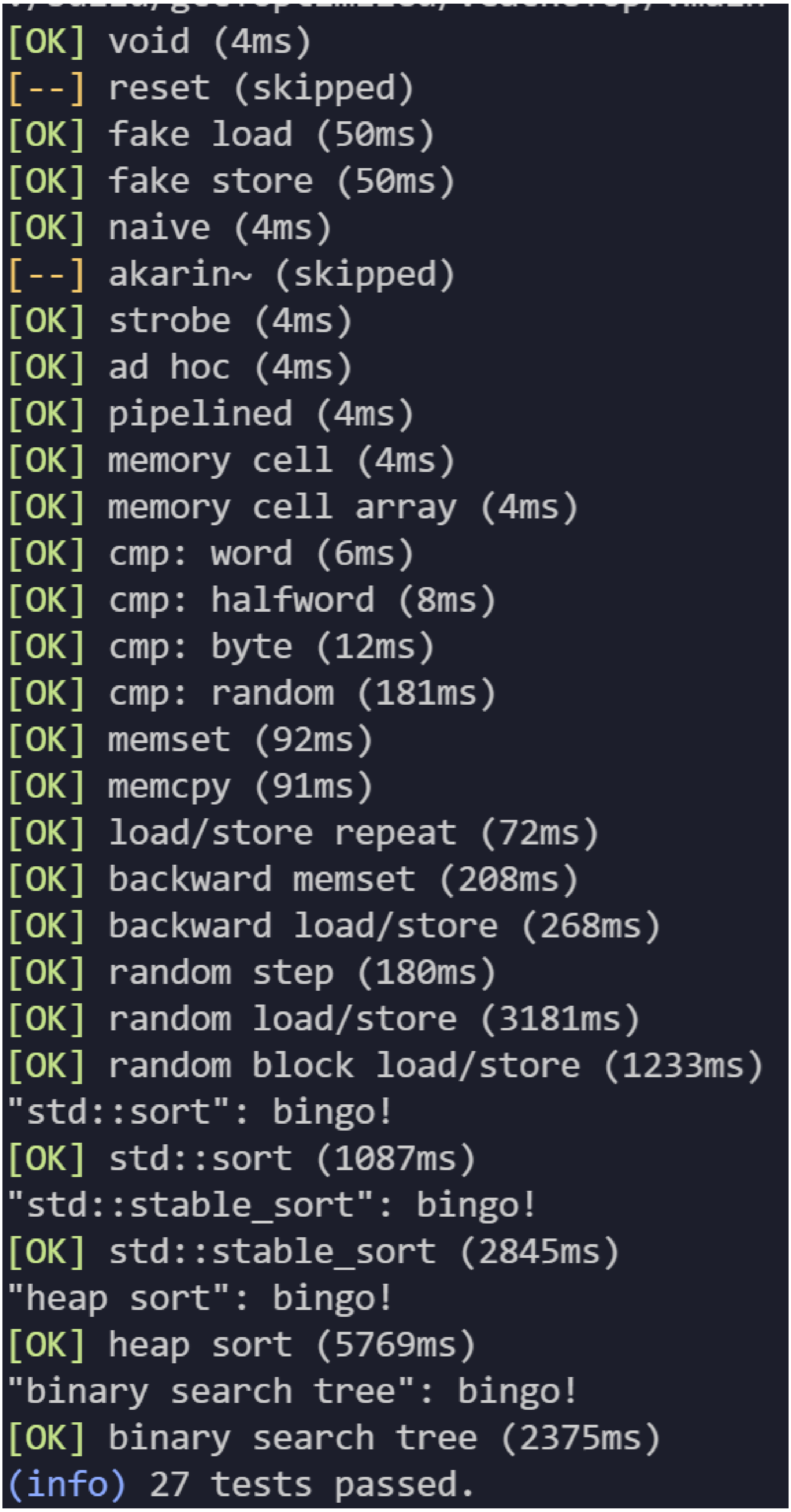
1.ALU 实现 64 位的乘除法器，在非除零（除零有特判，没有延迟）的情况下有固定的 64 周期延迟，　　因此需要阻塞住流水线，阻塞信号是 1 当且仅当乘除法器正在使用（valid），且未算完。将这个阻塞信号传给冲突控制模块，将全部流水段阻塞住，保证可能要转发的数据不丢失。

2.实现 hit0 周期延迟的 cache。由于其读取指令延迟不确定，因此对于 decode 阶段传出的 branch 信号和之前存下的 branch_nxt 都有可能被使用，这完全取决于有没有被 stall 住，且当任意一个被使用时，都需要 flush 之后的流水线寄存器。且当当前指令依赖前两个指令的数据，而上一条指令是 mem 相关，上上条是 load 时，可能会出现上一条指令读缓存需要 stall 而，此时上上条指令的数据已经被写到寄存器，但流水线寄存器已经不存在这个数据，无法转发的情况。此时需要在 decode 阶段阻塞一个周期，即可获得在寄存器内的正确数据。

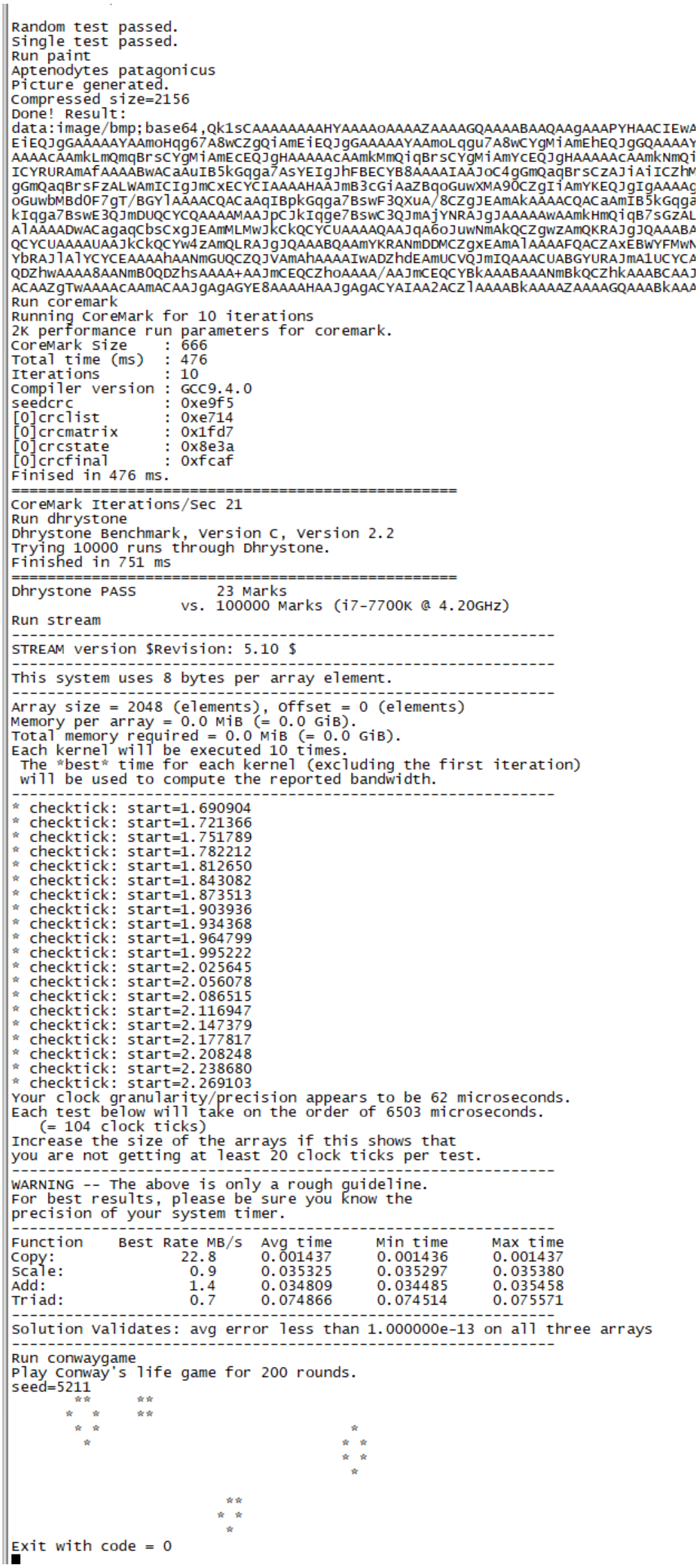
二．实验截图



```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:              10.1     0.003438     0.003256     0.003561
Scale:              0.7     0.049219     0.048799     0.049665
Add:                0.5     0.094338     0.093329     0.095182
Triad:              0.3     0.161604     0.160409     0.163068
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
Run conwaygame
Play Conway's life game for 200 rounds.
seed=5398




                                     ***
     **
     **

                            **
                            **
[src/cpu/cpu-exec.c,320,cpu_exec] nemu: HIT GOOD TRAP at pc = 0x0000000080014e44
```



```
[OK] void (4ms)
[--] reset (skipped)
[OK] fake load (50ms)
[OK] fake store (50ms)
[OK] naive (4ms)
[--] akarin~ (skipped)
[OK] strobe (4ms)
[OK] ad hoc (4ms)
[OK] pipelined (4ms)
[OK] memory cell (4ms)
[OK] memory cell array (4ms)
[OK] cmp: word (6ms)
[OK] cmp: halfword (8ms)
[OK] cmp: byte (12ms)
[OK] cmp: random (181ms)
[OK] memset (92ms)
[OK] memcpy (91ms)
[OK] load/store repeat (72ms)
[OK] backward memset (208ms)
[OK] backward load/store (268ms)
[OK] random step (180ms)
[OK] random load/store (3181ms)
[OK] random block load/store (1233ms)
"std::sort": bingo!
[OK] std::sort (1087ms)
"std::stable_sort": bingo!
[OK] std::stable_sort (2845ms)
"heap sort": bingo!
[OK] heap sort (5769ms)
"binary search tree": bingo!
[OK] binary search tree (2375ms)
(info) 27 tests passed.
```



```
Random test passed.
Single test passed.
Run paint
Aptenodytes patagonicus
Picture generated.
Compressed size=2156
Done! Result:
```

```
data:image/bmp;base64,Qk1sCAAAAAAAAHYAAAAoAAAAZAAAAGQAAAABAAQAAgAAAPYHAACIEwA
EiEQJgGAAAAAYAAmoHqg67A8wCZgQiAmEiEQJgGAAAAAYAAmoLqgu7A8wCYgMiAmEhEQJgGQAAAAY
AAAACAAmkLmQmqBrsCYgMiAmEcEQJgHAAAAACAAmkMmQiqBrsCYgMiAmYcEQJgHAAAAACAAmkNmQi
ICYRURAmAfAAAABwACaAuIB5kGqga7AsYEIgJhFBECYB8AAAAIAAJoC4gGmQaqBrsCzAJiAiICZhM
gGmQaqBrsFzALWAmICIgJmCxECYCIAAAAHAAJmB3cGiAaZBqoGuwXMA90CZgIiAmYKEQJgIgAAAAg
oGuwbMBd0F7gT/BGYlAAAACQACaAqIBpkGqga7BswF3QXuA/8CZgJEAmAkAAAACQACaAmIB5kGqga
kIqga7BswE3QJmDUQCYCQAAAAMAAJpCJkIqge7BswC3QJmAjYNRAJgJAAAAAwAAmkHmQiqB7sGzAL
AlAAAADwACagaqCbsCxgJEAmMLMwJkCkQCYCUAAAAQAAJqA6oJuwNmAkQCZgwzAmQKRAJgJQAAABA
QCYCUAAAAUAAJkCkQCYw4zAmQLRAJgJQAAABQAAmYKRANmDDMCZgxEAmAlAAAAFQACZAxEBWYFMwN
YbRAJlAlYCYCEAAAAhAANmGUQCZQJVAmAhAAAAIwADZhdEAmUCVQJmIQAAACUABGYURAJmA1UCYCA
QDZhwAAAA8AANmBOQDZhsAAAA+AAJmCEQCZhoAAAA/AAJmCEQCYBkAAABAAANmBkQCZhkAAABCAAJ
ACAAZgTwAAAACAAmACAAJgAgAGYE8AAAAHAAJgAgACYAIAA2ACZlAAAABkAAAAZAAAAGQAAABkAAA
```

```
Run coremark
Running CoreMark for 10 iterations
2K performance run parameters for coremark.
CoreMark Size     : 666
Total time (ms)   : 476
Iterations        : 10
Compiler version  : GCC9.4.0
seedcrc           : 0xe9f5
[0]crclist        : 0xe714
[0]crcmatrix      : 0x1fd7
[0]crcstate       : 0x8e3a
[0]crcfinal       : 0xfcaf
Finised in 476 ms.
===================================================
CoreMark Iterations/Sec 21
Run dhrystone
Dhrystone Benchmark, Version C, Version 2.2
Trying 10000 runs through Dhrystone.
Finished in 751 ms
===================================================
Dhrystone PASS         23 Marks
                   vs. 100000 Marks (i7-7700K @ 4.20GHz)
Run stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 2048 (elements), Offset = 0 (elements)
Memory per array = 0.0 MiB (= 0.0 GiB).
Total memory required = 0.0 MiB (= 0.0 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
* checktick: start=1.690904
* checktick: start=1.721366
* checktick: start=1.751789
* checktick: start=1.782212
* checktick: start=1.812650
* checktick: start=1.843082
* checktick: start=1.873513
* checktick: start=1.903936
* checktick: start=1.934368
* checktick: start=1.964799
* checktick: start=1.995222
* checktick: start=2.025645
* checktick: start=2.056078
* checktick: start=2.086515
* checktick: start=2.116947
* checktick: start=2.147379
* checktick: start=2.177817
* checktick: start=2.208248
* checktick: start=2.238680
* checktick: start=2.269103
Your clock granularity/precision appears to be 62 microseconds.
Each test below will take on the order of 6503 microseconds.
   (= 104 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:              22.8     0.001437     0.001436     0.001437
Scale:              0.9     0.035325     0.035297     0.035380
Add:                1.4     0.034809     0.034485     0.035458
Triad:              0.7     0.074866     0.074514     0.075571
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
Run conwaygame
Play Conway's life game for 200 rounds.
seed=5211
       **    **
      *  *   **
       * *                            *
        *                            * *
                                     * *
                                      *

                        **
                       * *
                        *
Exit with code = 0
█
```