

**课程名称**  \_ \_\_计算机体系结构 \_\_\_

**学院** \_ 计算机学院 \_\_\_\_\_\_\_\_

**专业** \_ 计算机科学与技术\_\_\_

**班级**  \_ 2014211306\_\_\_\_\_\_\_\_\_

**学号** \_ 2014211292\_\_\_\_\_\_\_\_\_

**姓名** \_ 张有杰\_\_\_\_\_\_\_\_\_\_\_\_\_

实验 2:5 级超标量流水线

一、实验目的

1. 通过实验,加深对流水线技术的理解。

2. 初步掌握 SimpleScalar 的使用。

二、实验布置及要求

1. 利用 SimpleScalar 中的模拟器 sim‐outorder,运行其自带的测试程序:

./tests‐pisa/bin.little 目录下的 test‐math, test‐fmath 及 test‐printf。

2. 记录结果及生成的统计量,注意观察流水线相关的统计量。

3. 使用 gcc‐2.7.2.3 交叉编译器,构建 Mibench benchmark(基准测试程序)的 automotive

包(今后的实验中会用到)。在此过程中,需要将各文件夹下 Makefile 中的“gcc”替换为

SimpleScalar 中 的 交 叉 编 译 器 的 绝 对 路 径 ( 即 : SimpleScalar 安 装 目 录

/home/student/simplescalar/sslittle-na-sstrix/bin/gcc)。

4. 修改各文件夹下的 runme\_small.sh,使其能够调用 sim‐outorder 模拟器以完成 benchmark

的运行,并观察输出结果。

例:将 basicmath/runme\_small.sh 中:

basicmath\_small > output\_small.txt

变为:

/home/student/simplescalar/simplesim-3.0/sim-outorder basicmath\_small > output\_small.txt

三、实验环境

操作系统：Ubuntu 17.04

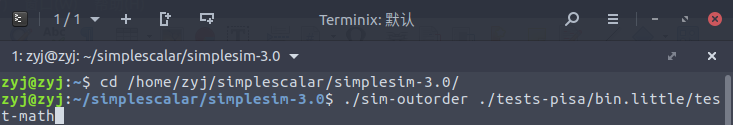
四、实验过程

1. 利用 SimpleScalar 中的模拟器 sim‐outorder,运行其自带的测试程序:

./tests‐pisa/bin.little 目录下的 test‐math, test‐fmath 及 test‐printf。

(1) 执行过程如下：首先使用 cd 命令切换工作目录至 $IDIR/simplesim-3.0/ 目录下，然后使用模拟器 sim-outorder 运行自带的测试程序。以test-math 为例，命令如下：

另外两个测试程序可以以类似方式完成。



2. 使用模拟器 sim-outorder 执行测试程序后，得到如下结果

将结果及统计量复制导出到外部文件中进行保存。

(1)test-math 的执行结果

sim: \*\* starting performance simulation \*\*

pow(12.0, 2.0) == 144.000000

pow(10.0, 3.0) == 1000.000000

pow(10.0, -3.0) == 0.001000

str: 123.456

x: 123.000000

str: 123.456

x: 123.456000

str: 123.456

x: 123.456000

123.456 123.456000 123 1000

sinh(2.0) = 3.62686

sinh(3.0) = 10.01787

h=3.60555

atan2(3,2) = 0.98279

pow(3.60555,4.0) = 169

169 / exp(0.98279 \* 5) = 1.24102

3.93117 + 5\*log(3.60555) = 10.34355

cos(10.34355) = -0.6068, sin(10.34355) = -0.79486

x 0.5x

x0.5 x

x 0.5x

-1e-17 != -1e-17 Worked!

sim: \*\* simulation statistics \*\*

sim\_num\_insn 213745 # total number of instructions committed

sim\_num\_refs 56902 # total number of loads and stores committed

sim\_num\_loads 34108 # total number of loads committed

sim\_num\_stores 22794.0000 # total number of stores committed

sim\_num\_branches 38605 # total number of branches committed

sim\_elapsed\_time 1 # total simulation time in seconds

sim\_inst\_rate 213745.0000 # simulation speed (in insts/sec)

sim\_total\_insn 233115 # total number of instructions executed

sim\_total\_refs 61930 # total number of loads and stores executed

sim\_total\_loads 37548 # total number of loads executed

sim\_total\_stores 24382.0000 # total number of stores executed

sim\_total\_branches 42799 # total number of branches executed

sim\_cycle 224375 # total simulation time in cycles

sim\_IPC 0.9526 # instructions per cycle

sim\_CPI 1.0497 # cycles per instruction

sim\_exec\_BW 1.0390 # total instructions (mis-spec + committed) per cycle

sim\_IPB 5.5367 # instruction per branch

IFQ\_count 352297 # cumulative IFQ occupancy

IFQ\_fcount 74037 # cumulative IFQ full count

ifq\_occupancy 1.5701 # avg IFQ occupancy (insn's)

ifq\_rate 1.0390 # avg IFQ dispatch rate (insn/cycle)

ifq\_latency 1.5113 # avg IFQ occupant latency (cycle's)

ifq\_full 0.3300 # fraction of time (cycle's) IFQ was full

RUU\_count 1440715 # cumulative RUU occupancy

RUU\_fcount 45204 # cumulative RUU full count

ruu\_occupancy 6.4210 # avg RUU occupancy (insn's)

ruu\_rate 1.0390 # avg RUU dispatch rate (insn/cycle)

ruu\_latency 6.1803 # avg RUU occupant latency (cycle's)

ruu\_full 0.2015 # fraction of time (cycle's) RUU was full

LSQ\_count 315312 # cumulative LSQ occupancy

LSQ\_fcount 5107 # cumulative LSQ full count

lsq\_occupancy 1.4053 # avg LSQ occupancy (insn's)

lsq\_rate 1.0390 # avg LSQ dispatch rate (insn/cycle)

lsq\_latency 1.3526 # avg LSQ occupant latency (cycle's)

lsq\_full 0.0228 # fraction of time (cycle's) LSQ was full

sim\_slip 1965692 # total number of slip cycles

avg\_sim\_slip 9.1964 # the average slip between issue and retirement

bpred\_bimod.lookups 44203 # total number of bpred lookups

bpred\_bimod.updates 38605 # total number of updates

bpred\_bimod.addr\_hits 34569 # total number of address-predicted hits

bpred\_bimod.dir\_hits 35160 # total number of direction-predicted hits (includes addr-hits)

bpred\_bimod.misses 3445 # total number of misses

bpred\_bimod.jr\_hits 3424 # total number of address-predicted hits for JR's

bpred\_bimod.jr\_seen 3544 # total number of JR's seen

bpred\_bimod.jr\_non\_ras\_hits.PP 28 # total number of address-predicted hits for non-RAS JR's

bpred\_bimod.jr\_non\_ras\_seen.PP 41 # total number of non-RAS JR's seen

bpred\_bimod.bpred\_addr\_rate 0.8955 # branch address-prediction rate (i.e., addr-hits/updates)

bpred\_bimod.bpred\_dir\_rate 0.9108 # branch direction-prediction rate (i.e., all-hits/updates)

bpred\_bimod.bpred\_jr\_rate 0.9661 # JR address-prediction rate (i.e., JR addr-hits/JRs seen)

bpred\_bimod.bpred\_jr\_non\_ras\_rate.PP 0.6829 # non-RAS JR addr-pred rate (ie, non-RAS JR hits/JRs seen)

bpred\_bimod.retstack\_pushes 3909 # total number of address pushed onto ret-addr stack

bpred\_bimod.retstack\_pops 4536 # total number of address popped off of ret-addr stack

bpred\_bimod.used\_ras.PP 3503 # total number of RAS predictions used

bpred\_bimod.ras\_hits.PP 3396 # total number of RAS hits

bpred\_bimod.ras\_rate.PP 0.9695 # RAS prediction rate (i.e., RAS hits/used RAS)

il1.accesses 256050 # total number of accesses

il1.hits 239944 # total number of hits

il1.misses 16106 # total number of misses

il1.replacements 15595 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0629 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0609 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 56904 # total number of accesses

dl1.hits 56353 # total number of hits

dl1.misses 551 # total number of misses

dl1.replacements 66 # total number of replacements

dl1.writebacks 61 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0097 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0012 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0011 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ul2.accesses 16718 # total number of accesses

ul2.hits 15496 # total number of hits

ul2.misses 1222 # total number of misses

ul2.replacements 0 # total number of replacements

ul2.writebacks 0 # total number of writebacks

ul2.invalidations 0 # total number of invalidations

ul2.miss\_rate 0.0731 # miss rate (i.e., misses/ref)

ul2.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

ul2.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

ul2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

itlb.accesses 256050 # total number of accesses

itlb.hits 256027 # total number of hits

itlb.misses 23 # total number of misses

itlb.replacements 0 # total number of replacements

itlb.writebacks 0 # total number of writebacks

itlb.invalidations 0 # total number of invalidations

itlb.miss\_rate 0.0001 # miss rate (i.e., misses/ref)

itlb.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

itlb.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

itlb.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dtlb.accesses 57590 # total number of accesses

dtlb.hits 57580 # total number of hits

dtlb.misses 10 # total number of misses

dtlb.replacements 0 # total number of replacements

dtlb.writebacks 0 # total number of writebacks

dtlb.invalidations 0 # total number of invalidations

dtlb.miss\_rate 0.0002 # miss rate (i.e., misses/ref)

dtlb.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

dtlb.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

dtlb.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

sim\_invalid\_addrs 0 # total non-speculative bogus addresses seen (debug var)

ld\_text\_base 0x00400000 # program text (code) segment base

ld\_text\_size 91744 # program text (code) size in bytes

ld\_data\_base 0x10000000 # program initialized data segment base

ld\_data\_size 13028 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x7fffc000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x00400140 # program entry point (initial PC)

ld\_environ\_base 0x7fff8000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 33 # total number of pages allocated

mem.page\_mem 132k # total size of memory pages allocated

mem.ptab\_misses 40 # total first level page table misses

mem.ptab\_accesses 2021619 # total page table accesses

mem.ptab\_miss\_rate 0.0000 # first level page table miss rate

(2)test-fmath 的执行结果

sim: \*\* starting performance simulation \*\*

q=4 (int)x=12 (int)y=29

z=144

z=841

z=13

z=13

l=6

l=36

\*lp=216

z=144.000000

q=4 x=12.000000 (int)x=12 y=29.000000 (int)y=29

q = 16 x = 11.700001 y = 23.400000

sim: \*\* simulation statistics \*\*

sim\_num\_insn 53504 # total number of instructions committed

sim\_num\_refs 16347 # total number of loads and stores committed

sim\_num\_loads 8635 # total number of loads committed

sim\_num\_stores 7712.0000 # total number of stores committed

sim\_num\_branches 10353 # total number of branches committed

sim\_elapsed\_time 1 # total simulation time in seconds

sim\_inst\_rate 53504.0000 # simulation speed (in insts/sec)

sim\_total\_insn 58079 # total number of instructions executed

sim\_total\_refs 17481 # total number of loads and stores executed

sim\_total\_loads 9398 # total number of loads executed

sim\_total\_stores 8083.0000 # total number of stores executed

sim\_total\_branches 11298 # total number of branches executed

sim\_cycle 68393 # total simulation time in cycles

sim\_IPC 0.7823 # instructions per cycle

sim\_CPI 1.2783 # cycles per instruction

sim\_exec\_BW 0.8492 # total instructions (mis-spec + committed) per cycle

sim\_IPB 5.1680 # instruction per branch

IFQ\_count 89690 # cumulative IFQ occupancy

IFQ\_fcount 18943 # cumulative IFQ full count

ifq\_occupancy 1.3114 # avg IFQ occupancy (insn's)

ifq\_rate 0.8492 # avg IFQ dispatch rate (insn/cycle)

ifq\_latency 1.5443 # avg IFQ occupant latency (cycle's)

ifq\_full 0.2770 # fraction of time (cycle's) IFQ was full

RUU\_count 359927 # cumulative RUU occupancy

RUU\_fcount 10584 # cumulative RUU full count

ruu\_occupancy 5.2626 # avg RUU occupancy (insn's)

ruu\_rate 0.8492 # avg RUU dispatch rate (insn/cycle)

ruu\_latency 6.1972 # avg RUU occupant latency (cycle's)

ruu\_full 0.1548 # fraction of time (cycle's) RUU was full

LSQ\_count 85683 # cumulative LSQ occupancy

LSQ\_fcount 2148 # cumulative LSQ full count

lsq\_occupancy 1.2528 # avg LSQ occupancy (insn's)

lsq\_rate 0.8492 # avg LSQ dispatch rate (insn/cycle)

lsq\_latency 1.4753 # avg LSQ occupant latency (cycle's)

lsq\_full 0.0314 # fraction of time (cycle's) LSQ was full

sim\_slip 501428 # total number of slip cycles

avg\_sim\_slip 9.3718 # the average slip between issue and retirement

bpred\_bimod.lookups 11563 # total number of bpred lookups

bpred\_bimod.updates 10353 # total number of updates

bpred\_bimod.addr\_hits 9123 # total number of address-predicted hits

bpred\_bimod.dir\_hits 9434 # total number of direction-predicted hits (includes addr-hits)

bpred\_bimod.misses 919 # total number of misses

bpred\_bimod.jr\_hits 792 # total number of address-predicted hits for JR's

bpred\_bimod.jr\_seen 833 # total number of JR's seen

bpred\_bimod.jr\_non\_ras\_hits.PP 4 # total number of address-predicted hits for non-RAS JR's

bpred\_bimod.jr\_non\_ras\_seen.PP 19 # total number of non-RAS JR's seen

bpred\_bimod.bpred\_addr\_rate 0.8812 # branch address-prediction rate (i.e., addr-hits/updates)

bpred\_bimod.bpred\_dir\_rate 0.9112 # branch direction-prediction rate (i.e., all-hits/updates)

bpred\_bimod.bpred\_jr\_rate 0.9508 # JR address-prediction rate (i.e., JR addr-hits/JRs seen)

bpred\_bimod.bpred\_jr\_non\_ras\_rate.PP 0.2105 # non-RAS JR addr-pred rate (ie, non-RAS JR hits/JRs seen)

bpred\_bimod.retstack\_pushes 942 # total number of address pushed onto ret-addr stack

bpred\_bimod.retstack\_pops 930 # total number of address popped off of ret-addr stack

bpred\_bimod.used\_ras.PP 814 # total number of RAS predictions used

bpred\_bimod.ras\_hits.PP 788 # total number of RAS hits

bpred\_bimod.ras\_rate.PP 0.9681 # RAS prediction rate (i.e., RAS hits/used RAS)

il1.accesses 63612 # total number of accesses

il1.hits 59269 # total number of hits

il1.misses 4343 # total number of misses

il1.replacements 3847 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0683 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0605 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 16431 # total number of accesses

dl1.hits 15955 # total number of hits

dl1.misses 476 # total number of misses

dl1.replacements 28 # total number of replacements

dl1.writebacks 22 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0290 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0017 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0013 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ul2.accesses 4841 # total number of accesses

ul2.hits 3974 # total number of hits

ul2.misses 867 # total number of misses

ul2.replacements 0 # total number of replacements

ul2.writebacks 0 # total number of writebacks

ul2.invalidations 0 # total number of invalidations

ul2.miss\_rate 0.1791 # miss rate (i.e., misses/ref)

ul2.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

ul2.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

ul2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

itlb.accesses 63612 # total number of accesses

itlb.hits 63592 # total number of hits

itlb.misses 20 # total number of misses

itlb.replacements 0 # total number of replacements

itlb.writebacks 0 # total number of writebacks

itlb.invalidations 0 # total number of invalidations

itlb.miss\_rate 0.0003 # miss rate (i.e., misses/ref)

itlb.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

itlb.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

itlb.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dtlb.accesses 16595 # total number of accesses

dtlb.hits 16585 # total number of hits

dtlb.misses 10 # total number of misses

dtlb.replacements 0 # total number of replacements

dtlb.writebacks 0 # total number of writebacks

dtlb.invalidations 0 # total number of invalidations

dtlb.miss\_rate 0.0006 # miss rate (i.e., misses/ref)

dtlb.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

dtlb.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

dtlb.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

sim\_invalid\_addrs 0 # total non-speculative bogus addresses seen (debug var)

ld\_text\_base 0x00400000 # program text (code) segment base

ld\_text\_size 79920 # program text (code) size in bytes

ld\_data\_base 0x10000000 # program initialized data segment base

ld\_data\_size 12288 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x7fffc000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x00400140 # program entry point (initial PC)

ld\_environ\_base 0x7fff8000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 30 # total number of pages allocated

mem.page\_mem 120k # total size of memory pages allocated

mem.ptab\_misses 34 # total first level page table misses

mem.ptab\_accesses 879916 # total page table accesses

mem.ptab\_miss\_rate 0.0000 # first level page table miss rate

(3)test-printf 的执行结果

sim: \*\* starting performance simulation \*\*

%.4x: `0012'

%04x: `0012'

%4.4x: `0012'

%04.4x: `0012'

%4.3x: ` 012'

%04.3x: `0012'

%.\*x: `0012'

%0\*x: `0012'

%\*.\*x: `0012'

%0\*.\*x: `0012'

bad format: "%z"

nil pointer (padded): " (nil)"

decimal negative: "-2345"

octal negative: "37777773327"

hex negative: "fffff6d7"

long decimal number: "-123456"

long octal negative: "37777773327"

long unsigned decimal number: "4294843840"

zero-padded LDN: "-000123456"

left-adjusted ZLDN: "-123456 "

space-padded LDN: " -123456"

left-adjusted SLDN: "-123456 "

zero-padded string: " Hi, Z."

left-adjusted Z string: "Hi, Z. "

space-padded string: " Hi, Z."

left-adjusted S string: "Hi, Z. "

null string: "(null)"

e-style >= 1: "1.234000e+01"

e-style >= .1: "1.234000e-01"

e-style < .1: "1.234000e-03"

e-style big: "1.000000000000000000000000000000000000000000000000000000000000e+20"

e-style == .1: "1.000000e-01"

f-style >= 1: "12.340000"

f-style >= .1: "0.123400"

f-style < .1: "0.001234"

g-style >= 1: "12.34"

g-style >= .1: "0.1234"

g-style < .1: "0.00123"

g-style big: "100000000000000000000"

:new test: 99.85:

0.10000

0.10000

x0.5000x

0x1

| 0.0000| 0.0000e+00| 0|

| 1.0000| 1.0000e+00| 1|

| -1.0000| -1.0000e+00| -1|

| 100.0000| 1.0000e+02| 100|

| 1000.0000| 1.0000e+03| 1000|

| 10000.0000| 1.0000e+04| 1e+04|

| 12345.0000| 1.2345e+04| 1.235e+04|

| 100000.0000| 1.0000e+05| 1e+05|

| 123456.0000| 1.2346e+05| 1.235e+05|

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

Formatted output test

prefix 6d 6o 6x 6X 6u

%-+#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-+# |-123 |0377 |0xff |0XFF |4294967295 |

%-+0 |-123 |377 |ff |FF |4294967295 |

%-+ |-123 |377 |ff |FF |4294967295 |

%-#0 |-123 |0377 |0xff |0XFF |4294967295 |

%-# |-123 |0377 |0xff |0XFF |4294967295 |

%-0 |-123 |377 |ff |FF |4294967295 |

%- |-123 |377 |ff |FF |4294967295 |

%+#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%+# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%+0 |-00123 |000377 |0000ff |0000FF |4294967295 |

%+ | -123 | 377 | ff | FF |4294967295 |

%#0 |-00123 |000377 |0x00ff |0X00FF |4294967295 |

%# | -123 | 0377 | 0xff | 0XFF |4294967295 |

%0 |-00123 |000377 |0000ff |0000FF |4294967295 |

% | -123 | 377 | ff | FF |4294967295 |

(null)

(null)

1.234568e+06 should be 1.234568e+06

1234567.800000 should be 1234567.800000

1.23457e+06 should be 1.23457e+06

123.456 should be 123.456

1e+06 should be 1e+06

10 should be 10

0.02 should be 0.02

testing parametric fields: 0.7000

sim: \*\* simulation statistics \*\*

sim\_num\_insn 1813937 # total number of instructions committed

sim\_num\_refs 516949 # total number of loads and stores committed

sim\_num\_loads 348604 # total number of loads committed

sim\_num\_stores 168345.0000 # total number of stores committed

sim\_num\_branches 401630 # total number of branches committed

sim\_elapsed\_time 1 # total simulation time in seconds

sim\_inst\_rate 1813937.0000 # simulation speed (in insts/sec)

sim\_total\_insn 1984229 # total number of instructions executed

sim\_total\_refs 560618 # total number of loads and stores executed

sim\_total\_loads 379039 # total number of loads executed

sim\_total\_stores 181579.0000 # total number of stores executed

sim\_total\_branches 450155 # total number of branches executed

sim\_cycle 1395643 # total simulation time in cycles

sim\_IPC 1.2997 # instructions per cycle

sim\_CPI 0.7694 # cycles per instruction

sim\_exec\_BW 1.4217 # total instructions (mis-spec + committed) per cycle

sim\_IPB 4.5164 # instruction per branch

IFQ\_count 3424768 # cumulative IFQ occupancy

IFQ\_fcount 724224 # cumulative IFQ full count

ifq\_occupancy 2.4539 # avg IFQ occupancy (insn's)

ifq\_rate 1.4217 # avg IFQ dispatch rate (insn/cycle)

ifq\_latency 1.7260 # avg IFQ occupant latency (cycle's)

ifq\_full 0.5189 # fraction of time (cycle's) IFQ was full

RUU\_count 14062172 # cumulative RUU occupancy

RUU\_fcount 558697 # cumulative RUU full count

ruu\_occupancy 10.0758 # avg RUU occupancy (insn's)

ruu\_rate 1.4217 # avg RUU dispatch rate (insn/cycle)

ruu\_latency 7.0870 # avg RUU occupant latency (cycle's)

ruu\_full 0.4003 # fraction of time (cycle's) RUU was full

LSQ\_count 2777724 # cumulative LSQ occupancy

LSQ\_fcount 37082 # cumulative LSQ full count

lsq\_occupancy 1.9903 # avg LSQ occupancy (insn's)

lsq\_rate 1.4217 # avg LSQ dispatch rate (insn/cycle)

lsq\_latency 1.3999 # avg LSQ occupant latency (cycle's)

lsq\_full 0.0266 # fraction of time (cycle's) LSQ was full

sim\_slip 18576304 # total number of slip cycles

avg\_sim\_slip 10.2409 # the average slip between issue and retirement

bpred\_bimod.lookups 464071 # total number of bpred lookups

bpred\_bimod.updates 401630 # total number of updates

bpred\_bimod.addr\_hits 377941 # total number of address-predicted hits

bpred\_bimod.dir\_hits 380425 # total number of direction-predicted hits (includes addr-hits)

bpred\_bimod.misses 21205 # total number of misses

bpred\_bimod.jr\_hits 29956 # total number of address-predicted hits for JR's

bpred\_bimod.jr\_seen 31999 # total number of JR's seen

bpred\_bimod.jr\_non\_ras\_hits.PP 363 # total number of address-predicted hits for non-RAS JR's

bpred\_bimod.jr\_non\_ras\_seen.PP 2198 # total number of non-RAS JR's seen

bpred\_bimod.bpred\_addr\_rate 0.9410 # branch address-prediction rate (i.e., addr-hits/updates)

bpred\_bimod.bpred\_dir\_rate 0.9472 # branch direction-prediction rate (i.e., all-hits/updates)

bpred\_bimod.bpred\_jr\_rate 0.9362 # JR address-prediction rate (i.e., JR addr-hits/JRs seen)

bpred\_bimod.bpred\_jr\_non\_ras\_rate.PP 0.1652 # non-RAS JR addr-pred rate (ie, non-RAS JR hits/JRs seen)

bpred\_bimod.retstack\_pushes 34915 # total number of address pushed onto ret-addr stack

bpred\_bimod.retstack\_pops 31690 # total number of address popped off of ret-addr stack

bpred\_bimod.used\_ras.PP 29801 # total number of RAS predictions used

bpred\_bimod.ras\_hits.PP 29593 # total number of RAS hits

bpred\_bimod.ras\_rate.PP 0.9930 # RAS prediction rate (i.e., RAS hits/used RAS)

il1.accesses 2116704 # total number of accesses

il1.hits 2054684 # total number of hits

il1.misses 62020 # total number of misses

il1.replacements 61510 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0293 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0291 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 524334 # total number of accesses

dl1.hits 523762 # total number of hits

dl1.misses 572 # total number of misses

dl1.replacements 77 # total number of replacements

dl1.writebacks 73 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0011 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0001 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0001 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ul2.accesses 62665 # total number of accesses

ul2.hits 61523 # total number of hits

ul2.misses 1142 # total number of misses

ul2.replacements 0 # total number of replacements

ul2.writebacks 0 # total number of writebacks

ul2.invalidations 0 # total number of invalidations

ul2.miss\_rate 0.0182 # miss rate (i.e., misses/ref)

ul2.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

ul2.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

ul2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

itlb.accesses 2116704 # total number of accesses

itlb.hits 2116685 # total number of hits

itlb.misses 19 # total number of misses

itlb.replacements 0 # total number of replacements

itlb.writebacks 0 # total number of writebacks

itlb.invalidations 0 # total number of invalidations

itlb.miss\_rate 0.0000 # miss rate (i.e., misses/ref)

itlb.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

itlb.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

itlb.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dtlb.accesses 528808 # total number of accesses

dtlb.hits 528798 # total number of hits

dtlb.misses 10 # total number of misses

dtlb.replacements 0 # total number of replacements

dtlb.writebacks 0 # total number of writebacks

dtlb.invalidations 0 # total number of invalidations

dtlb.miss\_rate 0.0000 # miss rate (i.e., misses/ref)

dtlb.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

dtlb.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

dtlb.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

sim\_invalid\_addrs 0 # total non-speculative bogus addresses seen (debug var)

ld\_text\_base 0x00400000 # program text (code) segment base

ld\_text\_size 74640 # program text (code) size in bytes

ld\_data\_base 0x10000000 # program initialized data segment base

ld\_data\_size 13636 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x7fffc000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x00400140 # program entry point (initial PC)

ld\_environ\_base 0x7fff8000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 29 # total number of pages allocated

mem.page\_mem 116k # total size of memory pages allocated

mem.ptab\_misses 67 # total first level page table misses

mem.ptab\_accesses 12667210 # total page table accesses

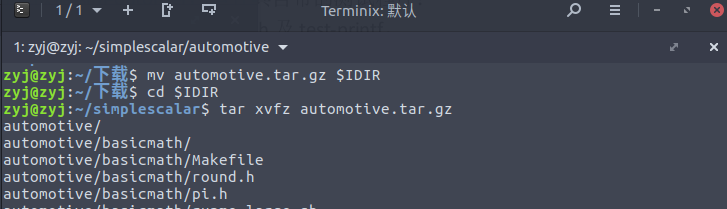
mem.ptab\_miss\_rate 0.0000 # first level page table miss rate

3. 使用 gcc‐2.7.2.3 交叉编译器,构建 Mibench benchmark(基准测试程序)的 automotive 包。在此过程中,需要将各文件夹下 Makefile 中的“gcc”替换为

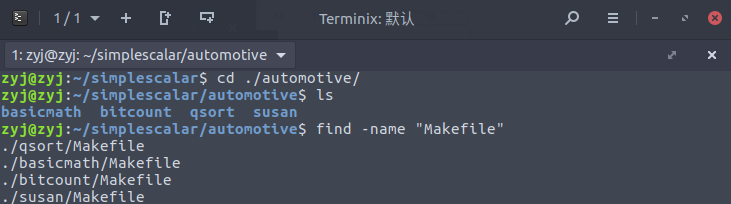
SimpleScalar 中 的 交 叉 编 译 器 的 绝 对 路 径 ( 即 : SimpleScalar 安 装 目 录

/home/student/simplescalar/sslittle-na-sstrix/bin/gcc)。

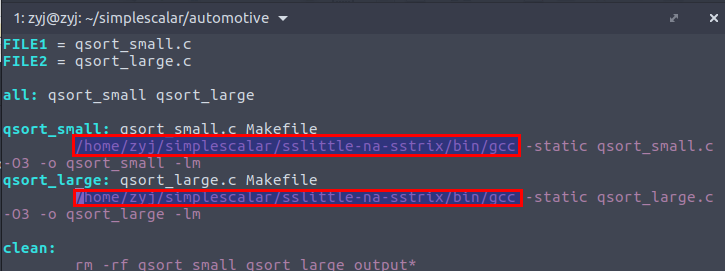
在 <http://vhosts.eecs.umich.edu/mibench/> 网页下载 Mibench benchmark 程序的 automotive 包，将其复制到 $IDIR 目录下后使用如下命令进行解压

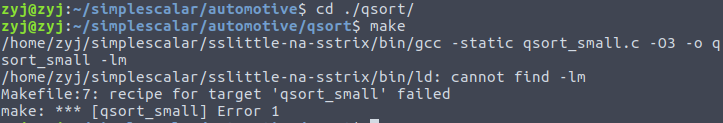


使用如下的命令找到 automotive 包中所有的 Makefile。



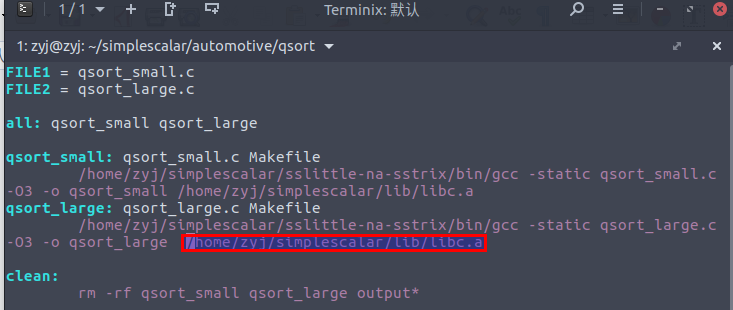
使用 vim 对以上 Makefile中的“gcc”替换为SimpleScalar 中 的 交 叉 编 译 器 的 绝 对 路 径 ( 即 : SimpleScalar 安 装 目 /home/student/simplescalar/sslittle-na-sstrix/bin/gcc)。

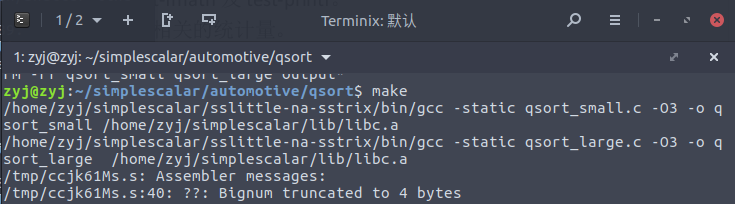
现在以 ./qsort/Makefile 为例,进行文件内容的替换。



查询资料得知这是由于静态库的链接错误导致的，修改 -lm 为 /home/zyj/simplescalar/lib/libc.a

新的 Makefile 如下

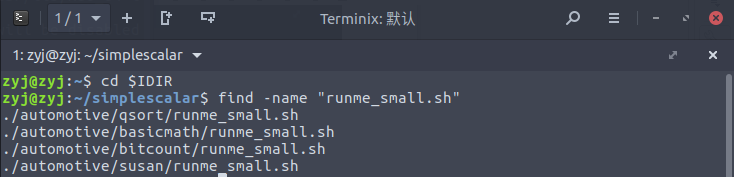


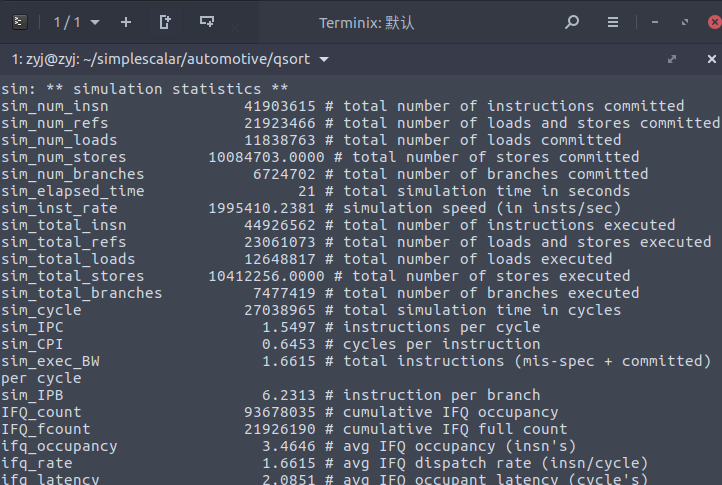
重新 make 后构建成功。

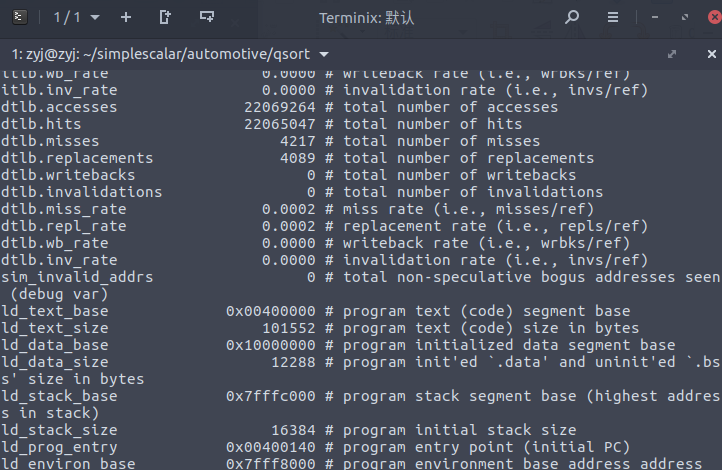
其余 basicmath 、bitcount、susan 的处理步骤相似，不再赘述。

4. 修改各文件夹下的 runme\_small.sh,使其能够调用 sim‐outorder 模拟器以完成 benchmark的运行,并观察输出结果。

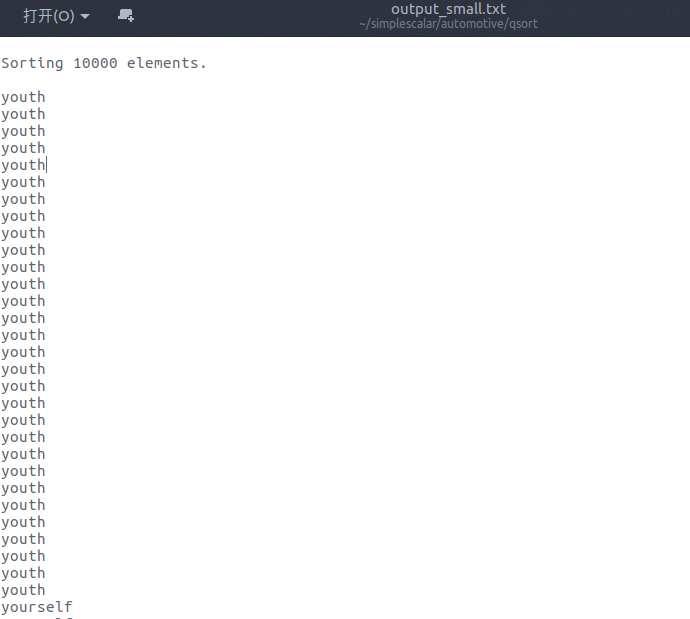
通过 find 命令查询得到所有的 runme\_small.sh 所在的位置。

以 basicmath 文件夹为例，按照实验指导书使用 vim 修改其中的 runme\_small.sh 。

(1)执行 ./qsort/runme\_small.sh 输出的部分统计结果如下：



输出结果output\_small.txt如下：



(2)执行 ./basicmath/runme\_small.sh 输出的部分输出结果如下。

\*\*\*\*\*\*\*\*\* CUBIC FUNCTIONS \*\*\*\*\*\*\*\*\*\*\*

Solutions: NaN

Solutions: NaN

Solutions: NaN

Solutions: NaN

\*\*\*\*\*\*\*\*\* INTEGER SQR ROOTS \*\*\*\*\*\*\*\*\*\*\*

sqrt( 0) = 0

sqrt( 1) = 65536

sqrt( 2) = 92681

sqrt( 3) = 113511

sqrt( 4) = 131072

sqrt( 5) = 146542

sqrt( 6) = 160529

sqrt( 7) = 173391

sqrt( 8) = 185363

sqrt( 9) = 196608

sqrt( 10) = 207243

sqrt( 11) = 217358

sqrt( 12) = 227023

sqrt( 13) = 236293

sqrt( 14) = 245213

sqrt( 15) = 253819

sqrt( 16) = 262144

sqrt( 17) = 270211

sqrt( 18) = 278045

sqrt( 19) = 285664

sqrt( 20) = 293085

sqrt( 21) = 300323

sqrt( 22) = 307391

sqrt( 23) = 314299

sqrt( 24) = 321059

sqrt( 25) = 327680

sqrt( 26) = 334169

sqrt( 27) = 340535

sqrt( 28) = 346783

sqrt( 29) = 352922

sqrt( 30) = 358955

sqrt( 31) = 364889

sqrt( 32) = 370727

sqrt( 33) = 376475

sqrt( 34) = 382137

sqrt( 35) = 387716

sqrt( 36) = 393216

sqrt( 37) = 398639

sqrt( 38) = 403991

sqrt( 39) = 409272

sqrt( 40) = 414486

sqrt( 41) = 419635

sqrt( 42) = 424721

sqrt( 43) = 429748

sqrt( 44) = 434716

sqrt( 45) = 439628

sqrt( 46) = 444486

sqrt( 47) = 449292

sqrt( 48) = 454046

sqrt( 49) = 458752

sqrt( 50) = 463409

sqrt( 51) = 468020

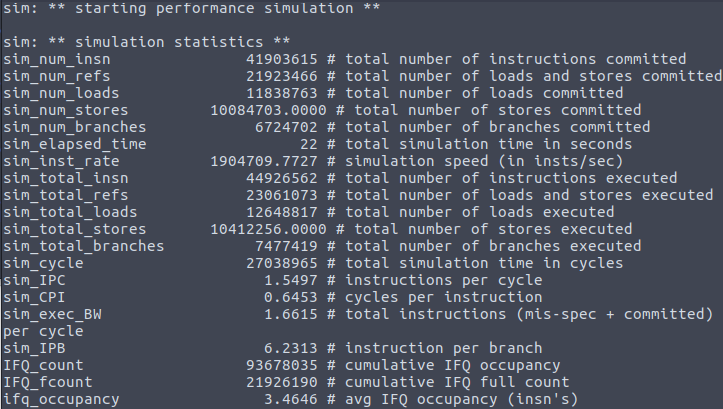
sqrt( 52) = 472586

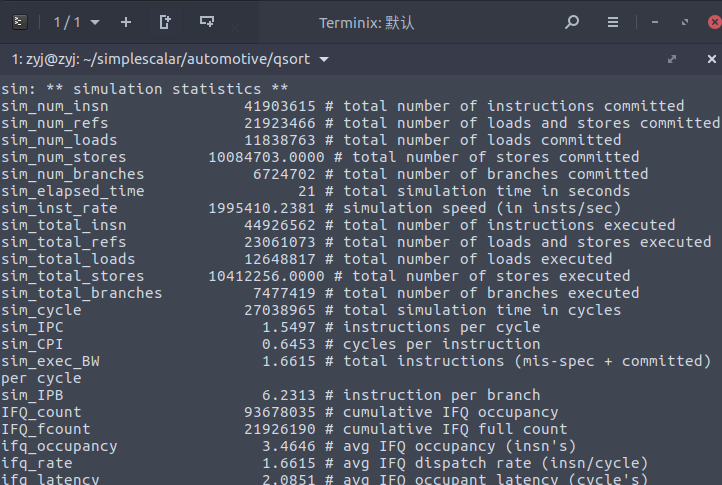
sqrt( 53) = 477109

sqrt( 54) = 481589

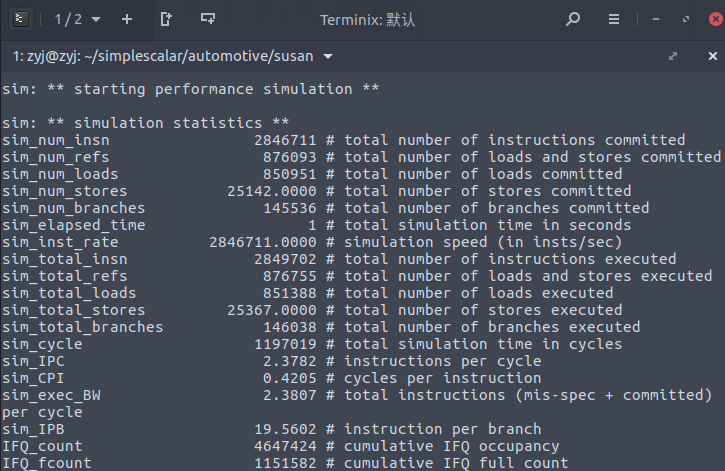
sqrt( 55) = 486027

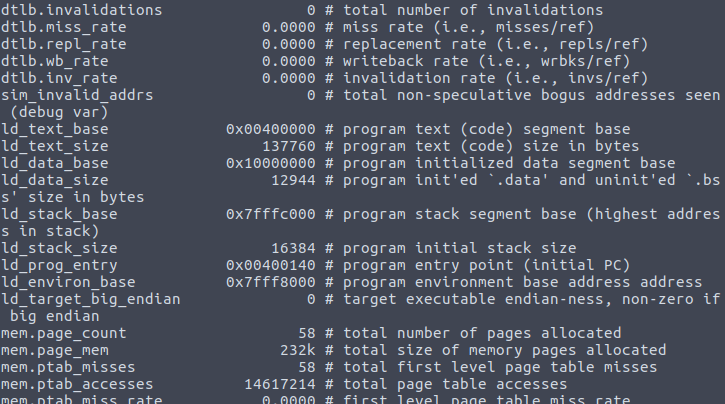
sqrt( 56) = 490426



(3)执行 ./bitcount/runme\_small.sh 输出的部分统计信息如下：

(4)执行 ./susan/runme\_small.sh 输出的部分统计信息如下。输出的 output\_small.\* 中都是乱码，故不再截图展示。





四、实验总结

在此次实验中我完成了 5 级超标量流水线的仿真实验，在观察每一次仿真的统计数据和输出结果的同时也熟悉了 Linux 的使用，这让我对超标量流水线的工作原理有了进一步的理解，同时也锻炼了我的动手能力。