动态分支预测实验

调研报告

2014211304 班

D 组

史文翰 2014211218

林宇辰 2014211223

王剑督 2014211228

崔嘉伟 2014211233

杨　莹 2014211238

徐丹雅 2014211243

郝绍明 2014210123

分工情况

林宇辰 2014211223 徐丹雅 2014211243 ：

负责 Linux 操作，下载相关实验文件，编译函数并观察、记录参数，撰写文档。

史文翰 2014211218：王剑督 2014211228 崔嘉伟 2014211233

杨　莹 2014211238 郝绍明 2014210123：

负责查阅资料等辅助性工作。

一、实验目的

(1) 了解动态分支预测的基本技术。

(2) 比较各种分支预测技术的性能。

二、实验环境

操作系统：Microsoft Windows 10 家庭中文版（x64）

虚拟机：VMware® Workstation 12 Pro

Linux： Ubuntu x64

三、实验布置及要求

本次实验使用分支预测模拟器 sim‐bpred，在 5 种预测器类型及不同的参数配置下运行测试程序，并比较、分析结果，使大家加深对动态分支预测的含义的理解，并了解各种分支预测实现方式的优劣。

1. 进入 SimpleScalar 目录(simplesim‐3.0)。

2. 用 sim‐bpred 仿真器运行 tests/bin.little 目录下的三个测试程序(test‐math, test‐fmath, test‐printf)，分别采用五种不同的分支预测方法，即 bimod 方式，two‐level adaptive 方式，always taken 方式，always not taken 方式，comb 方式，并对前两种分别使用 下表中两种参数配置:分析仿真器输出的关于分支预测的统计参数集,填写表格，并对各仿真器的能力给出相应说明。

命令格式为： ./sim-bpred {-option} executable_benchmark -argument

例：./sim‐bpred ‐bpred comb ‐bpred:comb 1024 ‐bpred:2lev 1 1024 8 0 ‐bpred:bimod 1024 tests/bin.little/test‐math

注：为了方便，可在命令后追加重定向语句（> 【文件名】），以将程序的输出重定向至 一文件。

各 benchmark 的执行文件及需要的参数：

Basicmath：

basicmath_small

Bitcount：

bitcnts 75000

Qsort：

qsort_small input_small.dat Susan： susan input_small.pgm output_small.smoothing.pgm ‐s

susan input_small.pgm output_small.edges.pgm ‐e

susan input_small.pgm output_small.corners.pgm ‐c

3. 编写 shell 脚本， 用 sim‐bpred 仿真器运行 Mibench 中 automotive 系列的几个 bechmark 的 small version（basicmath，bitcount，qsort，susan，其中 shell 脚本格式及各

benchmark 的执行参数可参照每个目录下的 runme_small.sh，注意文件扩展名要设为.sh），分别采用五种不同的分支预测方法，即 bimod 方式，two－level adaptive 方式，always taken 方式，always not taken 方式，comb 方式，并对前两种分别使用下表中两种参数配置； 分析仿真器输出的关于分支预测的统计参数集，填写表格，并对各仿真器的能力给 出相应说明。

注：为了方便，可在命令后追加重定向语句（> 【文件名】），以将程序的输出重定向至 一文件。

各 benchmark 的执行文件及需要的参数：
Basicmath：
basicmath_small Bitcount：
bitcnts 75000 Qsort：
qsort_small input_small.dat Susan：
susan input_small.pgm output_small.smoothing.pgm －s
susan input_small.pgm output_small.edges.pgm －e
susan input_small.pgm output_small.corners.pgm －c

四、实验原理

SimpleScalar 分支预测的实现方法：先进行分支方向探测，即是否采取分支（当然跳转指令和调用返回指令不用作这一步），接着是生成分支地址，对于调返指令，直接在 RAS 上作相关操作，普通分支指令则要利用 BTB 来进行地址探测，命中则生成地址。然后对两步综合，地址命中且分支预测为采取，返回分支目标地址；地址不命中且分支预测为采取，返回 1;只要分支预测为不采取，就返回 0。 重点分析针对条件分支指令的方向探测方法，主要有 6 种，三种静态：taken,not tanken,perfect;三种动态：bimod,2－level,comb。静态的方法顾名思义，只是 perfect 这种，按 它的原意是不预测，直接把真正采取的下一条指令填入 npc，而且它确实不需要调用 pred－lookup 函数，但据有人说，这种方法有时性能不如 bimod，另外在 sim－bpred 中好象也 没加入这种方法，只是在 sim－outorder 中有实现。

对于三种动态方法，分别说明如下：

bimod 是普通的，即采用一个 2bit 宽的分支方向预测表，按分支地址查找，2bit 分 支预测器的判断和更新与课本上的一致。这种方式只有一个参数，就是分支预测表的长度。

2-level 要复杂一些，它采用两级表格式，第一级是分支历史表，存放各组分支历史寄 存器的值，第二级是全局/局部分支模式表，（全局或局部应是由表长相对于分支历史寄存 器的长决定），它存放各分支历史模式的 2bit 预测器。在判断时用当前分支指令对应的历史寄存器值去索引二级表得到相应预测器值。更新时，把当前分支的方向左移入历史寄存器，并对使用过的 2bit 预测器作更新。它有四个参数，前三个是一级表长度，二级表长度，历史寄存器宽度，后一个是异或标志。如果为 1，则将历史寄存器的值与当前分支指令地址 异或，用其结果再去索引二级模式表。

comb 方式 组合了以上两种方法，它再加入了一个 meta 表，这个表类似 bimod 的预测表，只是它预测的是采取 bimod 还是 2-level，也采用 2-bit 预测器，被采取的预测方法被 定为第一方向，未被采取的定为第二方向。更新时，如果第一方向与第二方向不同则更新 meta 表，否则只更新两种方法各自的表即可。它共有三组参数，前两组即 bimod 和 two-level 的参数，第三组是关于 meta 表长度的说明。至于 BTB 的更新与 cache 的更新方式相同。在 现代处理器中分支预测是必不可少的，也是设计中比较复杂的部分。

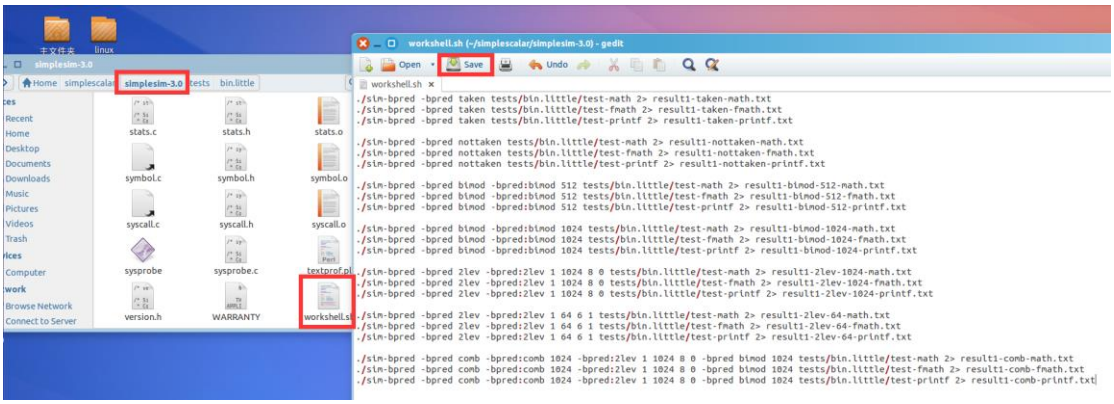
在 SimpleScalar 的乱序执行模拟 sim-outorder 中，控制指令的 Next_PC 在流水线的 ruu_dispatch 阶段计算出来，分支预测则是在取指阶段完成，预测的下一指令 PC 值信息跟随当前指令流入流水线。分支预测的模式有四种：

BPred2Level: 两级分支预测模式
BPred2bit:简单的两位直接映射预测模式，（分支目标缓存 BTB）
BPredTaken:分支发生静态预测模拟
BPredNotTaken: 分支不发生静态预测模拟

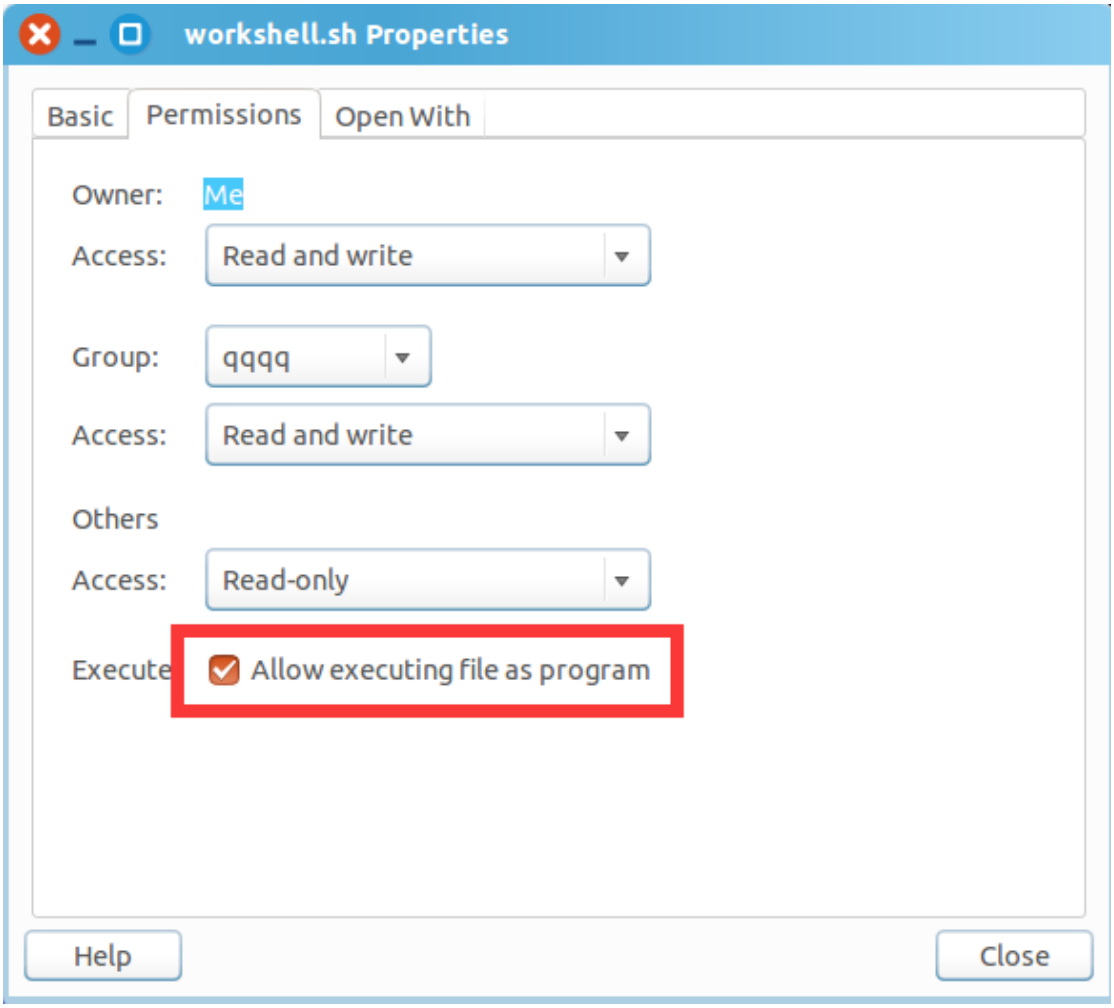
另外可以使用两级和位预测联合的预测模式(BPredComb)。

五、实验步骤

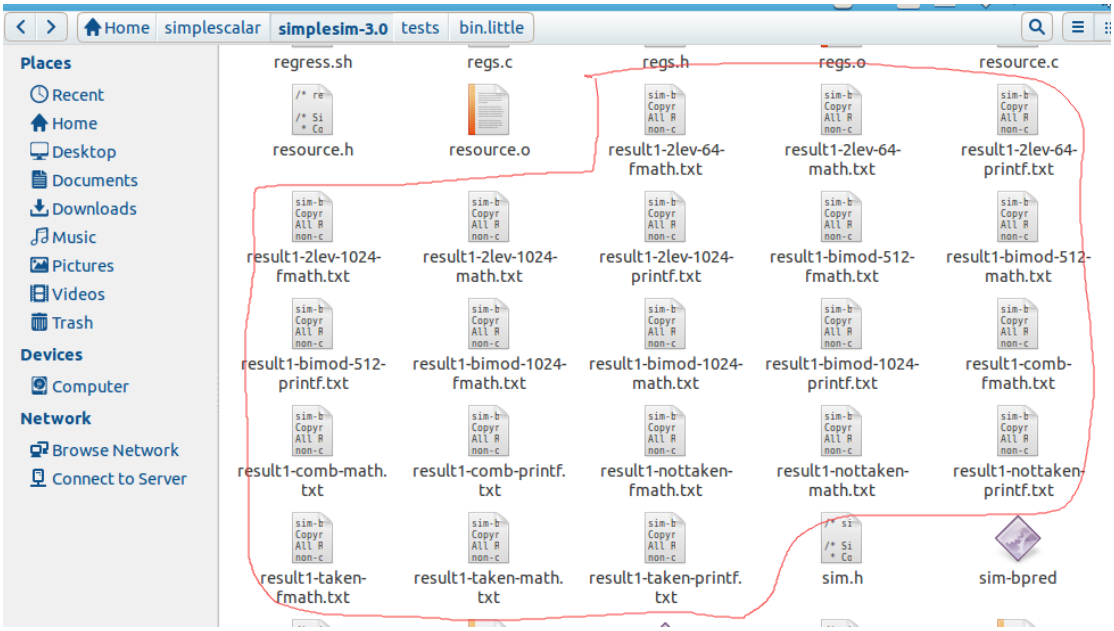
1. 运行虚拟机，载入成功安装了 SimpleScalar 的 Ubuntu 系统。
2. 打开 Terminal，进入 SimpleScalar 目录。
3. 用 sim-bpred 仿真器运行 tests/bin.little 目录下的三个测试程序(test-math, test-fmath, test-printf)，分别采用五种不同的分支预测方法，即 bimod 方式，two-level adaptive 方式，always taken 方式，always not taken 方式，comb 方式，分析仿真器输出的关于分支预测的统计参数集。用 gedit 编写 workshell.sh 如下：



手动赋予权限：



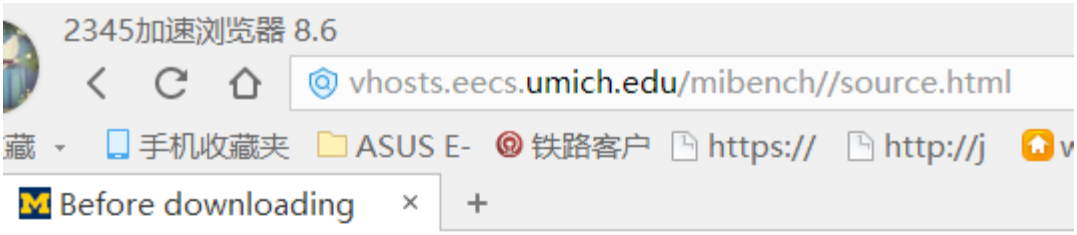
键入 ./step2.sh，将结果写入相应文件。



将相应结果统计成表格形式（见实验结果）。

4. 下载 Mibench 的源代码，并利用交叉编译器交叉编译其中的 benchmark 程序。

a) 下载 Mibench 源代码到主目录



Before downloading, there are a few things that should be r

1) Each benchmark has in its home directory a file named "LICENS general rule, all benchmarks are considered to be covered by GNU's

2) Each benchmark also contains in it home directory a file call instructions have been used to test and verify each benchmark on a compiled and run on an ARM system as is seen in the results of the benchmark may also have a README or INSTALL file that may contain these more detailed instructions.

3) Each benchmark (except for sphinx and pgp) contains two scrip scripts can be used to run the benchmark using the provided small be used as reference as to how the benchmark should be run with in "tests". For pgp, there is a single script called runme.sh.

The following tar'ed and gzip'ed files contain the benchmar MiBench:

automotive.tar.gz (889 KB)

consumer.tar.gz (30 MB)

network.tar.gz (470 KB)

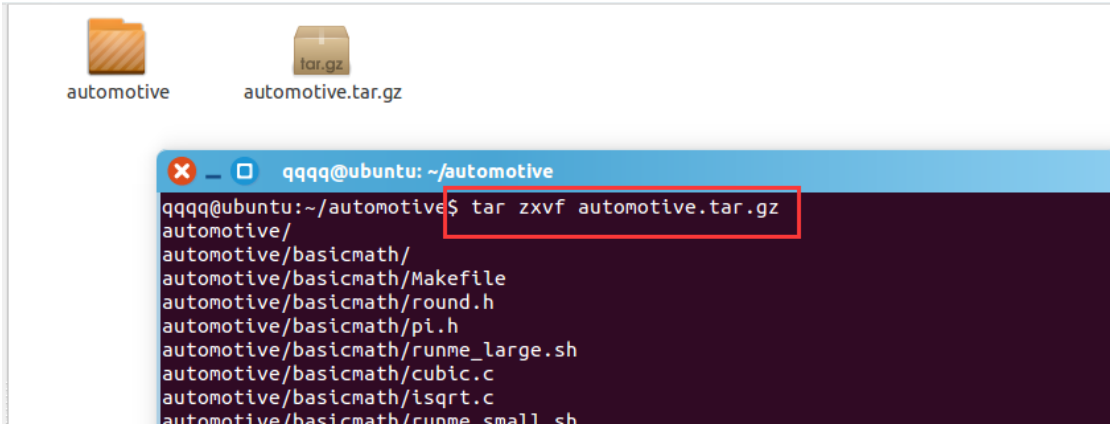
office.tar.gz (14 MB)

security.tar.gz (2 MB)

telecomm.tar.gz (34 MB)

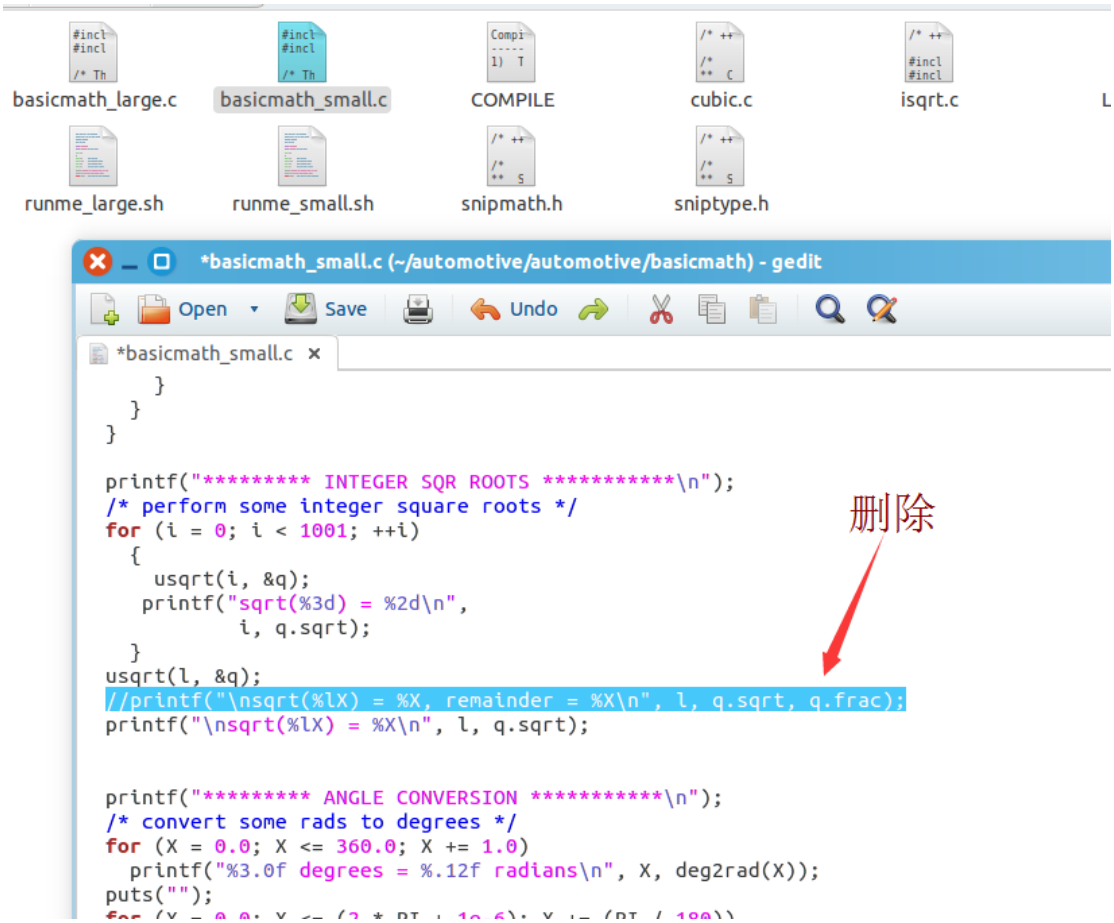
Return to Homepage

b) 解压源代码到当前目录

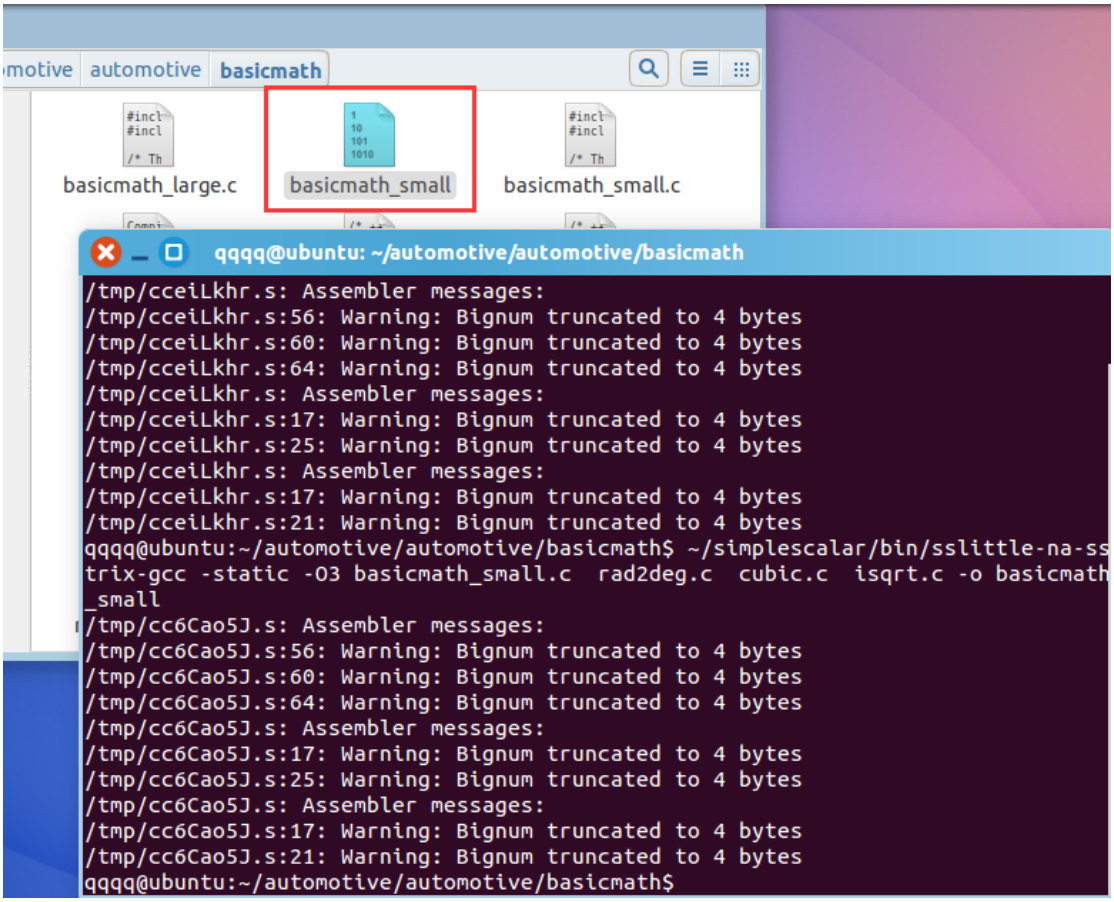


c) 修改 basicmath，使其能够通过交叉编译器的编译

i. 由于老版本的 gcc 没有包括 // 注释的特性，所以直接编译会错误，所以需要将 basicmath_small.c 中的所有//注释去除。

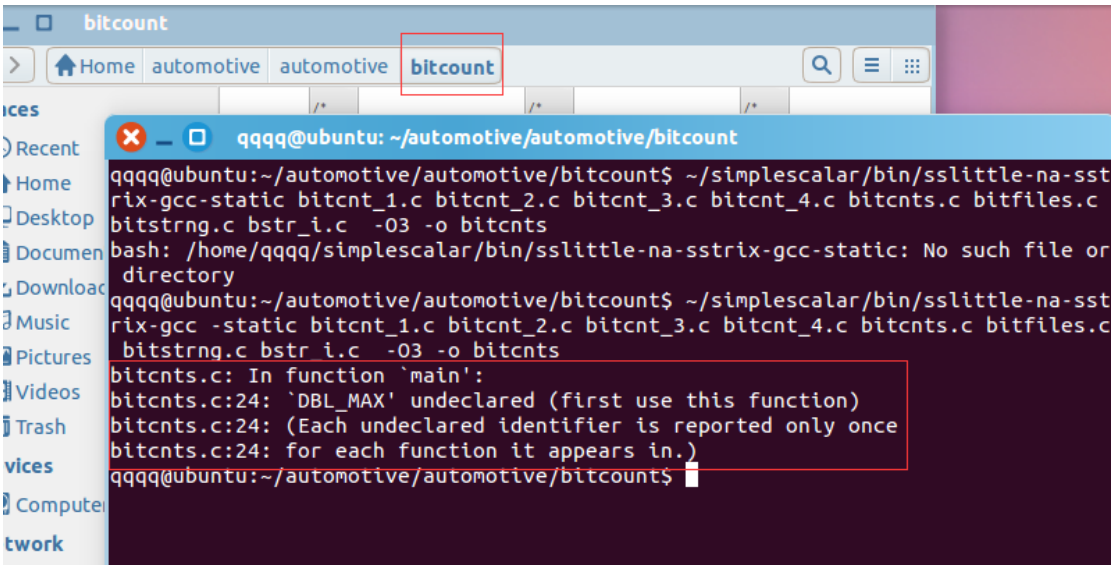


ii. 再次交叉编译，会发现遇到段错误，去掉编译选项 -lm 即可正确完成编译。



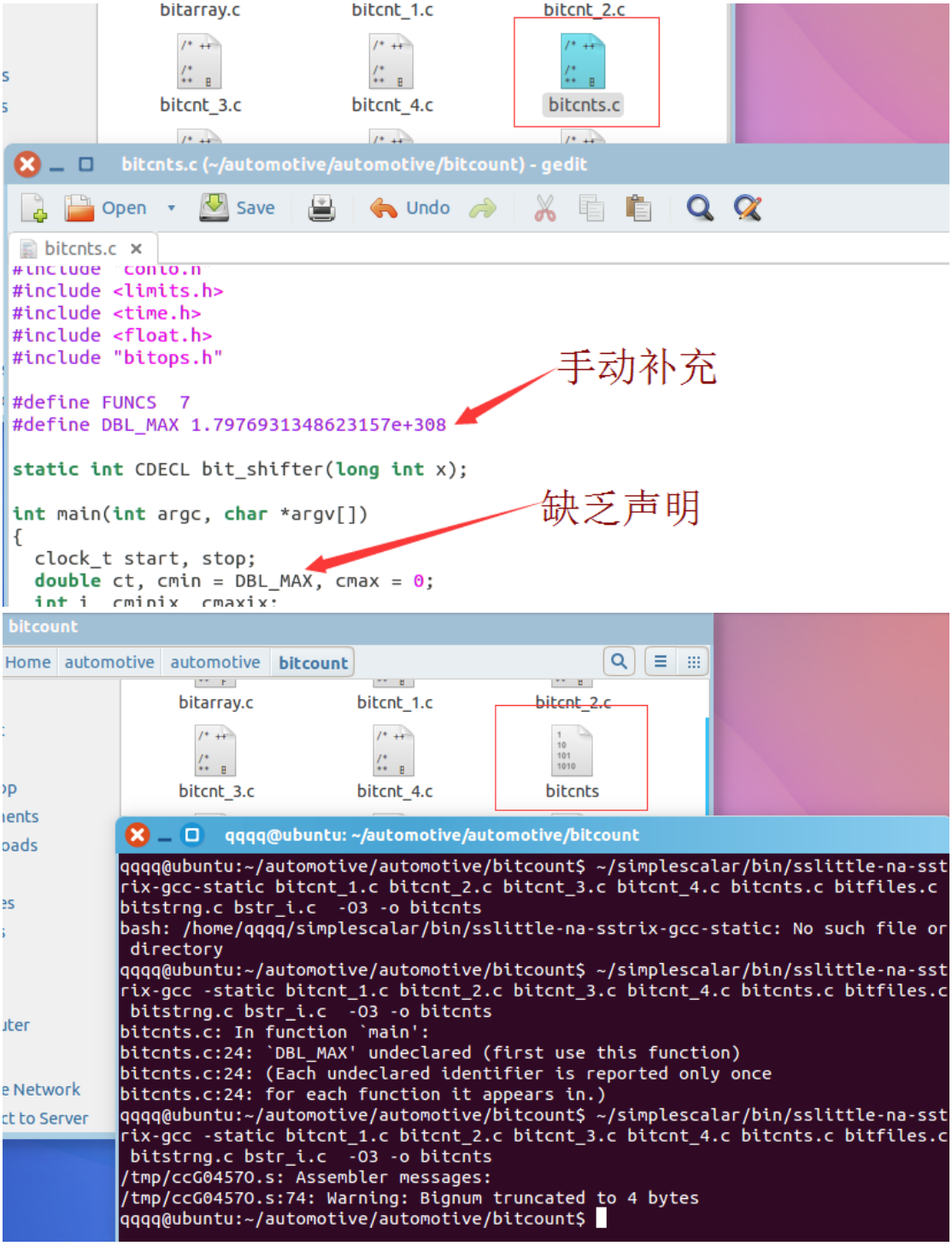
d) 修改 bitcount，使其能够通过交叉编译器的编译

i. 首先直接交叉编译会遇到如下问题



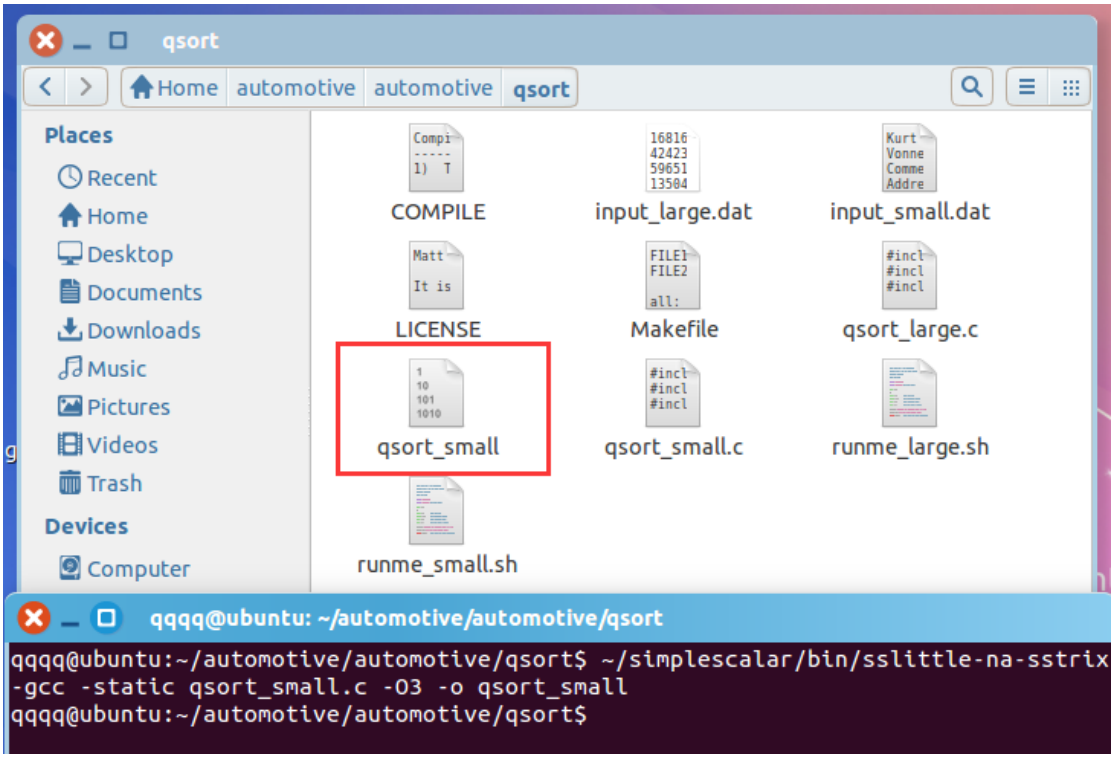
打开 bitcnts.c ，会发现 24 行中引用了 DBL_MAX 变量，在所有的.h 文件中都没有 DBL_MAX 的定义，在.c 文件中手动输入如下：

iii. 再次编译，即通过编译



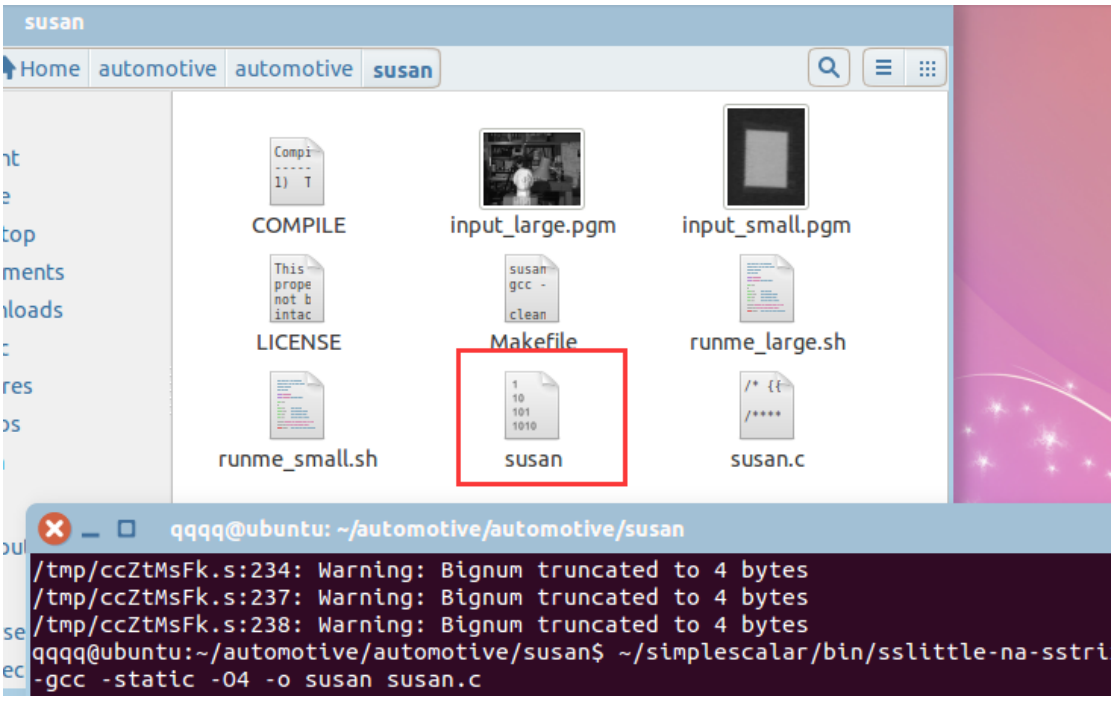
e) 修改 qsort，使其能够通过交叉编译器的编译

i. 首先直接编译，发现同 basicmath_small 一样，遇到了段错误，所以去掉-lm 选项，直接编译通过



f) 修改 susan，使其能够通过交叉编译器的编译

i. 直接编译去掉-lm 即可通过。

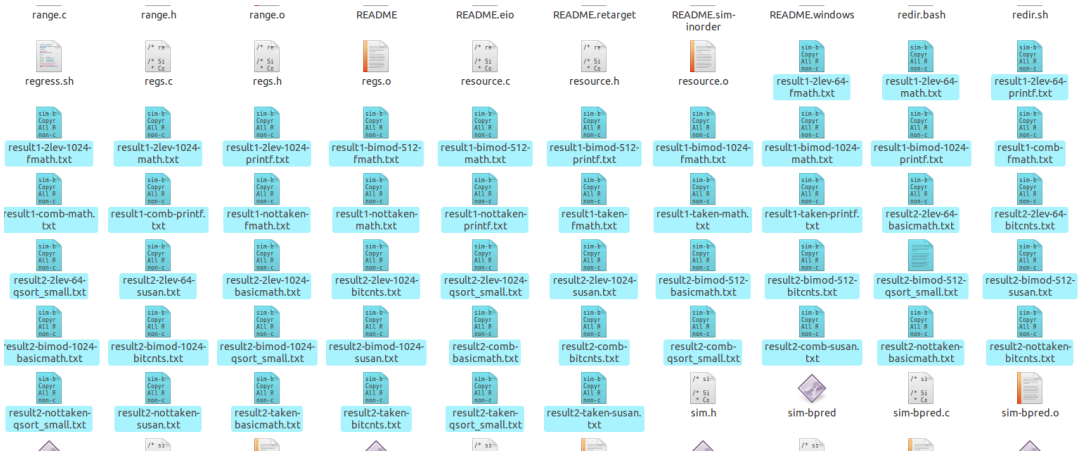


5. 编写 shell 脚本， 用 sim-bpred 仿真器运行 Mibench 中 automotive 系列的几个 benchmark 的 small version（basicmath，bitcount，qsort，susan，其中 shell 脚本格式及各 benchmark 的执行参数可参照每个目录下的 runme_small.sh，注意文件扩展名要设为.sh），分别采用五种不同的分支预测方法，即 bimod 方式，two-level adaptive 方式，always taken 方式，always not taken 方式，comb 方式：

如下编写 workshell.sh：

```
workshell2.sh (~/simplescalar/simplesim-3.0) - gedit
Open  Save  Undo
workshell2.sh ×
#!/bin/sh

cd ~automotive/automotive/basicmath
~/simplescalar/simplesim-3.0/sim-bpred -bpred taken basicmath_small 2> result2-taken-basicmath.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred nottaken basicmath_small 2> result2-nottaken-basicmath.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 512 basicmath_small 2> result2-bimod-512-basicmath.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 1024 basicmath_small 2> result2-bimod-1024-basicmath.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 0 basicmath_small 2> result2-2lev-1024-basicmath.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 basicmath_small 2> result2-2lev-64-basicmath.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred comb -bpred:comb 1024 -bpred:2lev 1 1024 8 0 -bpred:bimod 1024 basicmath_small 2> result2-comb-basicmath.txt

cd ~automotive/automotive/bitcount
~/simplescalar/simplesim-3.0/sim-bpred -bpred taken bitcnts 75000 2> result2-taken-bitcnts.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred nottaken bitcnts 75000 2> result2-nottaken-bitcnts.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 512 bitcnts 75000 2> result2-bimod-512-bitcnts.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 1024 bitcnts 75000 2> result2-bimod-1024-bitcnts.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 0 bitcnts 75000 2> result2-2lev-1024-bitcnts.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 bitcnts 75000 2> result2-2lev-64-bitcnts.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred comb -bpred:comb 1024 -bpred:2lev 1 1024 8 0 -bpred:bimod 1024 bitcnts 75000 2> result2-comb-bitcnts.txt

cd ~automotive/automotive/qsort
~/simplescalar/simplesim-3.0/sim-bpred -bpred taken qsort_small input_small.dat 2> result2-taken-qsort_small.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred nottaken qsort_small input_small.dat 2> result2-nottaken-qsort_small.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 512 qsort_small input_small.dat2> result2-bimod-512-qsort_small.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 1024 qsort_small input_small.dat 2> result2-bimod-1024-qsort_small.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 0 qsort_small input_small.dat 2> result2-2lev-1024-qsort_small.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 qsort_small input_small.dat 2> result2-2lev-64-qsort_small.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred comb -bpred:comb 1024 -bpred:2lev 1 1024 8 0 -bpred:bimod 1024 qsort_small input_small.dat 2> result2-comb-qsort_small.txt

cd ~automotive/automotive/susan
~/simplescalar/simplesim-3.0/sim-bpred -bpred taken susan input_small.pgm output_small.smoothing.pgm -s 2> result2-taken-susan.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred nottaken susan input_small.pgm output_small.smoothing.pgm -s 2> result2-nottaken-susan.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 512 susan input_small.pgm output_small.smoothing.pgm -s 2> result2-bimod-512-susan.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred bimod -bpred:bimod 1024 susan input_small.pgm output_small.smoothing.pgm -s 2> result2-bimod-1024-susan.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 0 susan input_small.pgm output_small.smoothing.pgm -s 2> result2-2lev-1024-susan.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 susan input_small.pgm output_small.smoothing.pgm -s 2> result2-2lev-64-susan.txt
~/simplescalar/simplesim-3.0/sim-bpred -bpred comb -bpred:comb 1024 -bpred:2lev 1 1024 8 0 -bpred:bimod 1024 susan input_small.pgm output_small.smoothing.pgm -s 2> result2-comb-susan.txt
```

命令行中输入./workshell2 之后，对应目录下生成如下 txt 文件：



下图展示了其中一个.txt 文件的样例：

```
result1-2lev-64-fmath.txt (~/simplescalar/simplesim-3.0) - gedit
Open  Save  Undo
result1-2lev-64-fmath.txt ×
      W   width of shift register(s)
      M   # entries in 2nd level (# of counters, or other FSM)
      X   (yes-1/no-0) xor history and address for 2nd level index
    Sample predictors:
      GAg     : 1, W, 2^W, 0
      GAp     : 1, W, M (M > 2^W), 0
      PAg     : N, W, 2^W, 0
      PAp     : N, W, M (M == 2^(N+W)), 0
      gshare  : 1, W, 2^W, 1
  Predictor `comb' combines a bimodal and a 2-level predictor.


sim: ** starting functional simulation w/ predictors **

sim: ** simulation statistics **
sim_num_insn                53448 # total number of instructions executed
sim_num_refs                16342 # total number of loads and stores executed
sim_elapsed_time                1 # total simulation time in seconds
sim_inst_rate           53448.0000 # simulation speed (in insts/sec)
sim_num_branches            10340 # total number of branches executed
sim_IPB                    5.1691 # instruction per branch
bpred_2lev.lookups          10340 # total number of bpred lookups
bpred_2lev.updates          10340 # total number of updates
bpred_2lev.addr_hits         7541 # total number of address-predicted hits
bpred_2lev.dir_hits          7857 # total number of direction-predicted hits (includes addr-hits)
bpred_2lev.misses            2483 # total number of misses
bpred_2lev.jr_hits            815 # total number of address-predicted hits for JR's
bpred_2lev.jr_seen            832 # total number of JR's seen
bpred_2lev.jr_non_ras_hits.PP           4 # total number of address-predicted hits for non-RAS JR's
bpred_2lev.jr_non_ras_seen.PP          19 # total number of non-RAS JR's seen
bpred_2lev.bpred_addr_rate    0.7293 # branch address-prediction rate (i.e., addr-hits/updates)
bpred_2lev.bpred_dir_rate     0.7599 # branch direction-prediction rate (i.e., all-hits/updates)
bpred_2lev.bpred_jr_rate      0.9796 # JR address-prediction rate (i.e., JR addr-hits/JRs seen)
bpred_2lev.bpred_jr_non_ras_rate.PP    0.2105 # non-RAS JR addr-pred rate (ie, non-RAS JR hits/JRs seen)
bpred_2lev.retstack_pushes          811 # total number of address pushed onto ret-addr stack
bpred_2lev.retstack_pops            813 # total number of address popped off of ret-addr stack
bpred_2lev.used_ras.PP              813 # total number of RAS predictions used
bpred_2lev.ras_hits.PP              811 # total number of RAS hits
bpred_2lev.ras_rate.PP        0.9975 # RAS prediction rate (i.e., RAS hits/used RAS)
```

将文件中框出的参数进行记录，便可以得到如下表格：

六、实验结果

Testmath							
	Alwaystaken	always not taken	bimon(512)	bimod(1024)	2-level(1,1024,8,0)	2-level(1,64,6,1)	comb(1024)(1,1024,8,0)(1024)
sim_totol_insn	224255	224255	224255	224255	224255	224255	224255
sim_totol_refs	58284	58284	58284	58284	58284	58284	58284
sim_num_branches	38525	38525	38525	38525	38525	38525	38525
sim_elapsed_time	1	1	1	1	1	1	1
sim_inst_rate	224255	224255	224255	224255	224255	224255	224255
sim_IPB	5.4243	5.4243	5.4243	5.4243	5.4243	5.4243	5.4243
bpred_bimod.lookups	38525	38525	38525	38525	38525	38525	38525
bpred_bimod.updates	38525	38525	38525	38525	38525	38525	38525
bpred_bimod.addr_hits	23434	23453	29646	31535	32467	28575	315757
bpred_bimod.dir_hits	23434	23453	29646	31535	32467	28575	315757
bpred_bimod.misses	12438	17345	4645	3966	3767	10253	2645
bpred_bimod.jr_hits	3643	3643	3397	3397	3397	3397	3397
bpred_bimod.jr_seen	3643	3643	3643	3643	3643	3643	3643
bpred_bimod.jr_non_ras_hits.PP	3424	3424	31	31	31	31	31
bpred_bimod.jr_non_ras_seen.PP	3424	3424	39	39	39	39	39
bpred_bimod.bpred_addr_rate	0.6678	0.5636	0.8535	0.8575	0.8535	0.7935	0.8045
bpred_bimod.bpred_dir_rate	0.6473	0.4925	0.8975	0.8355	0.7934	0.9575	0.8483
bpred_bimod.bpred_jr_rate	1.0001	1	0.9942	0.9924	0.9936	0.9957	0.9957
bpred_bimod.bpred_jr_non_ras_rate.PP	1.0003	1.0001	0.6842	0.6843	0.6843	0.6843	0.6843
bpred_bimod.retstack_pushes	0	0	3847	3847	3847	3847	3847
bpred_bimod.retstack_pops	0	0	4024	4024	4024	4024	4024
bpred_bimod.used_ra	0	0	4024	4024	4024	4024	4024

s.PP							
bpred_bimod.ras_hits.PP	0	0	3948	3948	3948	3948	3948
bpred_bimod.ras_rate.PP	<error:divide by zero>	<error:divide by zero>	0.9936	0.9936	0.9936	0.9936	0.9936

Testfmath							
	Alwaystaken	always not taken	bimon(512)	bimod(1024)	2-level(1,1024,8,0)	2-level(1,64,6,1)	comb(1024)(1,1024,8,0)(1024)
sim_totol_insn	55324	55324	55324	55324	55324	55324	55324
sim_totol_refs	16784	16784	16784	16784	16784	16784	16784
sim_num_branches	13894	13894	13894	13894	13894	13894	13894
sim_elapsed_time	1	1	1	1	1	1	1
sim_inst_rate	55324	55324	55324	55324	55324	55324	55324
sim_IPB	5.3278	5.3278	5.3278	5.3278	5.3278	5.3278	5.3278
bpred_bimod.lookups	13894	13894	13894	13894	13894	13894	13894
bpred_bimod.updates	13894	13894	13894	13894	13894	13894	13894
bpred_bimod.addr_hits	7836	7836	7836	7836	7836	7836	7836
bpred_bimod.dir_hits	6723	6723	6723	6723	6723	6723	6723
bpred_bimod.misses	3545	3643	1243	1024	1259	2475	895
bpred_bimod.jr_hits	832	832	820	820	820	820	820
bpred_bimod.jr_seen	832	832	832	832	832	832	832
bpred_bimod.jr_non_ras_hits.PP	832	832	4	4	4	4	4
bpred_bimod.jr_non_ras_seen.PP	832	832	18	18	18	18	18
bpred_bimod.bpred_addr_rate	0.6742	0.5424	0.7824	0.7842	0.7846	0.8364	0.8264
bpred_bimod.bpred_dir_rate	0.6742	0.5424	0.7824	0.7842	0.7846	0.8364	0.8264
bpred_bimod.bpred_jr_rate	1	1	0.9478	0.9478	0.9478	0.9478	0.9478
bpred_bimod.bpred_jr_non_ras_rate.PP	1	1	0.2732	0.2732	0.2732	0.2732	0.2732
bpred_bimod.retstack_pushes	0	0	820	820	820	820	820

bpred_bimod.retstack_pops	0	0	817	817	817	817	817
bpred_bimod.used_ras.PP	0	0	817	817	817	817	817
bpred_bimod.ras_hits.PP	0	0	813	813	813	813	813
bpred_bimod.ras_rate.PP	<error:divede by zero>	<error:divede by zero>	0.9946	0.9946	0.9946	0.9946	0.9946

Testprintf							
	Alwaystaken	always not taken	bimon(512)	bimod(1024)	2-level(1,1024,8,0)	2-level(1,64,6,1)	comb(1024)(1,1024,8,0)(1024)
sim_totol_insn	1988434	1988434	1988434	1988434	1988434	2E+06	1988434
sim_totol_refs	526584	526584	526584	526584	526584	526584	526584
sim_num_branches	482953	482953	482953	482953	482953	482953	482953
sim_elapsed_time	1	1	1	1	1	1	1
sim_inst_rate	1988434	1988434	1988434	1988434	1988434	2E+06	1988434
sim_IPB	4.2478	4.2478	4.2478	4.2478	4.2478	4.2478	4.2478
bpred_bimod.lookups	472875	472875	472875	472875	472875	472875	472875
bpred_bimod.updates	472875	472875	472875	472875	472875	472875	472875
bpred_bimod.addr_hits	278425	253856	264723	247596	279563	275855	261844
bpred_bimod.dir_hits	278425	253856	264723	247596	279563	275855	261844
bpred_bimod.misses	16247	18263	15738	27953	78243	79237	17230
bpred_bimod.jr_hits	328903	328903	379275	379275	379275	379275	379275
bpred_bimod.jr_seen	328903	328903	328903	328903	328903	328903	328903
bpred_bimod.jr_non_ras_hits.PP	328903	328903	385	385	385	385	385
bpred_bimod.jr_non_ras_seen.PP	328903	328903	2784	2784	2784	2784	2784
bpred_bimod.bpred_addr_rate	0.6248	0.3628	0.9723	0.8927	0.9742	0.8294	0.7929
bpred_bimod.bpred_dir_rate	0.6248	0.3628	0.9275	0.9782	0.8926	0.8426	0.9268
bpred_bimod.bpred_jr_rate	1	1	0.9728	0.9728	0.9728	0.9728	0.9728
bpred_bimod.bpred_jr_non_ras_rate.PP	1	1	0.1638	0.1638	0.1638	0.1638	0.1638

bpred_bimod.retstack_pushes	0	0	29374	29374	29374	29374	29374
bpred_bimod.retstack_pops	0	0	29473	29473	29473	29473	29473
bpred_bimod.used_ras.PP	0	0	29473	29473	29473	29473	29473
bpred_bimod.ras_hits.PP	0	0	29478	29478	29478	29478	29478
bpred_bimod.ras_rate.PP	<error:divede by zero>	<error:divede by zero>	0.9998	0.9998	0.9998	0.9998	0.9998

bitcnts							
	Alwaystaken	always not taken	bimon(512)	bimod(1024)	2-level(1,1024,8,0)	2-level(1,64,6,1)	comb(1024)(1,1024,8,0)(1024)
sim_totol_insn	47297059	47297059	44672084	46792462	46892523	48467274	46470245
sim_totol_refs	5647243	5242338	5234833	5274826	5672432	5243673	5824733
sim_num_branches	8924729	8279223	8929847	8724673	8678363	8732843	8742673
sim_elapsed_time	1	2	2	2	3	2	3
sim_inst_rate	47297059	21487823	21487823	21487823	14372599	21487823	14372599
sim_IPB	4.7482	4.7482	4.7482	4.7482	4.7482	4.7482	4.7482
bpred_bimod.lookups	7826538	7826538	7826538	7826538	7826538	7826538	7826538
bpred_bimod.updates	7826538	7826538	7826538	7826538	7826538	7826538	7826538
bpred_bimod.addr_hits	5638254	5638254	5638254	5638254	5638254	5638254	5638254
bpred_bimod.dir_hits	5638254	5638254	5638254	5638254	5638254	5638254	5638254
bpred_bimod.misses	3574263	3578735	254793	237584	178347	397954	157834
bpred_bimod.jr_hits	1034758	1035835	1043725	10347854	1058342	1057384	104382
bpred_bimod.jr_seen	1034758	1035835	1043725	10347854	1058342	1057384	104382
bpred_bimod.jr_non_ras_hits.PP	1034758	1035835	15	15	14	15	14
bpred_bimod.jr_non_ras_seen.PP	1034758	1035835	45	45	46	46	46
bpred_bimod.bpred_addr_rate	0.6742	0.5239	0.9427	0.9247	0.8937	0.9482	0.8935
bpred_bimod.bpred_dir_rate	0.6742	0.5239	0.9832	0.9254	0.9855	0.9326	0.9842

bpred_bimod.bpred_jr_rate	1	1	1	1	1	1	1
bpred_bimod.bpred_jr_non_ras_rate.PP	1	1	0.3333	0.3333	0.3258	0.3123	0.3034
bpred_bimod.retstack_pushes	0	0	1043725	10347854	1058342	1057384	104382
bpred_bimod.retstack_pops	0	0	1043725	10347854	1057384	1040382	1043274
bpred_bimod.used_ras.PP	0	0	1034578	1034578	1053894	1056372	10578343
bpred_bimod.ras_hits.PP	0	0	1057829	1057829	1057829	1057820	1057831
bpred_bimod.ras_rate.PP	<error:divede by zero>	<error:divede by zero>	1	1	1	1	1

qsort							
	Alwaystaken	always not taken	bimon(512)	bimod(1024)	2-level(1,1024,8,0)	2-level(1,64,6,1)	comb(1024)(1,1024,8,0)(1024)
sim_totol_insn	46747552	46747552	46747552	46747552	46747552	46747552	46747552
sim_totol_refs	24157901	24157901	24157901	24157901	24157901	24157901	24157901
sim_num_branches	7965782	7965782	7965782	7965782	7965782	7965782	7965782
sim_elapsed_time	3	2	5	8	5	6	4
sim_inst_rate	15693058.6667	23400571.0000	9374257.8000	5854276.0000	9371280.0000	7536117.0000	1172891.0000
sim_IPB	5.8916	5.8916	5.8916	5.8916	5.8916	5.8916	5.8916
bpred_bimod.lookups	7972048	7972048	7972048	7972048	7972048	7972048	7972048
bpred_bimod.updates	7972048	7972048	7972048	7972048	7972048	7972048	7972048
bpred_bimod.addr_hits	5455250	4892571	7471578	7508952	7741589	7065838	7778512
bpred_bimod.dir_hits	5455250	4892571	7473030	7508050	7725541	7061568	7773245
bpred_bimod.misses	2459874	3072574	488977	457425	237384	898896	187831
bpred_bimod.jr_hits	899814	899814	898145	898145	898145	898145	898145
bpred_bimod.jr_seen	899814	899814	899814	899814	899814	899814	899814

bpred_bimod.jr_non_ras_hits.PP	899814	899814	898866	898866	155881	155881	155881
bpred_bimod.jr_non_ras_seen.PP	899814	899814	155883	155883	155883	155883	155883
bpred_bimod.bpred_addr_rate	0.6842	0.6179	0.9827	0.9245	0.9537	0.8882	0.9754
bpred_bimod.bpred_dir_rate	0.67123	0.6238	0.9832	0.9255	0.9558	0.8890	0.9841
bpred_bimod.bpred_jr_rate	1	1	0.9998	0.9998	0.9998	0.9998	0.9998
bpred_bimod.bpred_jr_non_ras_rate.PP	1	0	1	1	1	1	1
bpred_bimod.retstack_pushes	0	0	745896	745896	745896	745896	745896
bpred_bimod.retstack_pops	0	0	745891	745891	745891	745891	745891
bpred_bimod.used_ras.PP	0	0	745891	745891	745891	745891	745891
bpred_bimod.ras_hits.PP	0	0	745528	745528	745528	745528	745528
bpred_bimod.ras_rate.PP	<error:divede by zero>	<error:divede by zero>	0.998	0.998	0.998	0.998	0.998

Susan							
	Alwaystaken	always not taken	bimon(512)	bimod(1024)	2-level(1,1024,8,0)	2-level(1,64,6,1)	comb(1024)(1,1024,8,0)(1024)
sim_totol_insn	1537831	1537831	1537831	1537831	1537831	1537831	1537831
sim_totol_refs	265348	265348	265348	265348	265348	265348	265348
sim_num_branches	156485	156485	156485	156485	156485	156485	156485
sim_elapsed_time	1	1	1	1	1	1	1
sim_inst_rate	1532846.0000	1532846.0000	1532846.0000	1532846.0000	1532846.0000	1532846.0000	1532846.0000
sim_IPB	9.6566	9.6566	9.6566	9.6566	9.6566	9.6566	9.6566
bpred_bimod.lookups	157654	157654	157654	157654	157654	157654	157654
bpred_bimod.updates	157654	157654	157654	157654	157654	157654	157654
bpred_bimod.addr_hits	98562	68517	128631	128636	143217	145245	151833
bpred_bimod.dir_hit	98562	68517	128135	128567	143235	145631	152048

s							
bpred_bimod.misses	63459	90621	28696	29796	15333	22458	7536
bpred_bimod.jr_hits	1948	1948	1945	1945	1945	1945	1945
bpred_bimod.jr_seen	1948	1948	1948	1948	1948	1948	1948
bpred_bimod.jr_non_ras_hits.PP	1948	1948	1	1	1	1	1
bpred_bimod.jr_non_ras_seen.PP	1948	1948	5	5	5	5	5
bpred_bimod.bpred_addr_rate	0.6096	0.4339	0.8106	0.8106	0.9014	0.8568	0.9452
bpred_bimod.bpred_dir_rate	0.6096	0.4339	0.8028	0.8028	0.9120	0.8557	0.9402
bpred_bimod.bpred_jr_rate	1	1	0.9987	0.9987	0.9987	0.9987	0.9987
bpred_bimod.bpred_jr_non_ras_rate.PP	1	1	0.2	0.2	0.2	0.2	0.2
bpred_bimod.retstack_pushes	0	0	1944	1944	1944	1944	1944
bpred_bimod.retstack_pops	0	0	1943	1943	1943	1943	1943
bpred_bimod.used_ras.PP	0	0	1943	1943	1943	1943	1943
bpred_bimod.ras_hits.PP	0	0	1941	1941	1941	1941	1941
bpred_bimod.ras_rate.PP	<error:divede by zero>	<error:divede by zero>	0.9996	0.9996	0.9996	0.9996	0.9996

七、结果分析

对于(方向)地址预测命中次数而言：
comb>2lev 1024>bimod 1024>bimod 512>2lev 64>always taken>always not taken.
（左侧为高命中次数）

对于未命中次数而言：
always not taken>always taken>2lev 64>bimod 512>bimod 1024>2lev 1024>comb.
（左侧为高 miss 次数）

结果发现，无论是就命中次数还是未命中次数而言，
comb 总是优于 2lev 1024 优于 bimod 1024 优于 bimod 512 优于 2lev 64 优于 always taken 优于 always not taken.