

**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 3:动态分支预测

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、实验布置及要求

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

参数用法详情可查阅 users\_guide\_v2.pdf。

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

三、实验环境

操作系统：Ubuntu 17.04

四、实验结果

(1) test-math

./sim-bpred -bpred taken tests/bin.little/test-math

./sim-bpred -bpred nottaken tests/bin.little/test-math

./sim-bpred -bpred bimod -bpred:bimod 512 tests/bin.little/test-math

./sim-bpred -bpred bimod -bpred:bimod 1024 tests/bin.little/test-math

./sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 01 tests/bin.little/test-math

./sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 tests/bin.little/test-math

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | always taken | always not taken | bimod  (512) | bimod  (1024) | 2-level  (1,1024,8,0) | 2-level  (1,64,6,1) | comb  (1024)  (1,1024,8,0)  (1024) |
| sim\_total\_insn | 213688 | 213688 | 213688 | 213688 | 213688 | 213688 | 213688 |
| sim\_total\_refs | 56897 | 56897 | 56897 | 56897 | 56897 | 56897 | 56897 |
| sim\_num\_branches | 38591 | 38591 | 38591 | 38591 | 38591 | 38591 | 38591 |
| sim\_elapsed\_time | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| sim\_inst\_rate | 213688 | 213688 | 213688 | 213688 | 213688 | 213688 | 213688 |
| sim\_IPB | 5.5372 | 5.5372 | 5.5372 | 5.5372 | 5.5372 | 5.5372 | 5.5372 |
| bpred\_bimod.lookups | 38591 | 38591 | 38591 | 38591 | 38591 | 38591 | 38591 |
| bpred\_bimod.updates | 38591 | 38591 | 38591 | 38591 | 38591 | 38591 | 38591 |
| bpred\_bimod.addr\_hits | 25661 | 21938 | 33692 | 34237 | 34242 | 27844 | 35570 |
| bpred\_bimod.dir\_hits | 25661 | 21938 | 34195 | 34732 | 34794 | 28387 | 36011 |
| bpred\_bimod.misses | 12930 | 16653 | 4396 | 3859 | 3797 | 10204 | 2580 |
| bpred\_bimod.jr\_hits | 3543 | 3543 | 3528 | 3528 | 3528 | 3528 | 3528 |
| bpred\_bimod.jr\_seen | 3543 | 3543 | 3543 | 3543 | 3543 | 3543 | 3543 |
| bpred\_bimod.jr\_non\_ras\_hits.PP | 3543 | 3543 | 28 | 28 | 28 | 28 | 28 |
| bpred\_bimod.jr\_non\_ras\_seen.PP | 3543 | 3543 | 41 | 41 | 41 | 41 | 41 |
| bpred\_bimod.bpred\_addr\_rate | 0.6649 | 0.5685 | 0.8731 | 0.8872 | 0.8873 | 0.7215 | 0.9217 |
| bpred\_bimod.bpred\_dir\_rate | 0.6649 | 0.5685 | 0.8861 | 0.9000 | 0.9016 | 0.7356 | 0.9331 |
| bpred\_bimod.bpred\_jr\_rate | 1.0000 | 1.0000 | 0.9958 | 0.9958 | 0.9958 | 0.9985 | 0.9958 |
| bpred\_bimod.bpred\_jr\_non\_ras\_rate.PP | 1.0000 | 1.0000 | 0.6829 | 0.6829 | 0.6829 | 0.6829 | 0.6829 |
| bpred\_bimod.retstack\_pushes | 0 | 0 | 3504 | 3504 | 3504 | 3504 | 3504 |
| bpred\_bimod.retstack\_pops | 0 | 0 | 3502 | 3502 | 3502 | 3502 | 3502 |
| bpred\_bimod.used\_ras.PP | 0 | 0 | 3502 | 3502 | 3502 | 3502 | 3502 |
| bpred\_bimod.ras\_hits.PP | 0 | 0 | 3500 | 3500 | 3500 | 3500 | 3500 |
| bpred\_bimod.ras\_rate.PP | Error:divide by zero | Error:divid by zero | 0.9994 | 0.9994 | 0.9994 | 0.9994 | 0.9994 |

(2) test-fmath

./sim-bpred -bpred taken tests/bin.little/test-fmath

./sim-bpred -bpred nottaken tests/bin.little/test-fmath

./sim-bpred -bpred bimod -bpred:bimod 512 tests/bin.little/test-fmath

./sim-bpred -bpred bimod -bpred:bimod 1024 tests/bin.little/test-fmath

./sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 01 tests/bin.little/test-fmath

./sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 tests/bin.little/test-fmath

./sim-bpred -bpred comb -bpred:comb 1024 -bpred:2lev 1 1024 8 0 -bpred:bimod 1024 tests/bin.little/test-fmath

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | always taken | always not taken | bimod  (512) | bimod  (1024) | 2-level  (1,1024,8,0) | 2-level  (1,64,6,1) | comb  (1024)  (1,1024,8,0)  (1024) |
| sim\_total\_insn | 53448 | 53448 | 53448 | 53448 | 53448 | 53448 | 53448 |
| sim\_total\_refs | 16342 | 16342 | 16342 | 16342 | 16342 | 16342 | 16342 |
| sim\_num\_branches | 10340 | 10340 | 10340 | 10340 | 10340 | 10340 | 10340 |
| sim\_elapsed\_time | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| sim\_inst\_rate | 53448.000 | 53448 | 53448 | 53448 | 53448 | 53448 | 53448 |
| sim\_IPB | 5.1691 | 5.1691 | 5.1691 | 5.1691 | 5.1691 | 5.1691 | 5.1691 |
| bpred\_bimod.lookups | 10340 | 10340 | 10340 | 10340 | 10340 | 10340 | 10340 |
| bpred\_bimod.updates | 10340 | 10340 | 10340 | 10340 | 10340 | 10340 | 10340 |
| bpred\_bimod.addr\_hits | 6839 | 5768 | 8825 | 9021 | 8814 | 7541 | 9226 |
| bpred\_bimod.dir\_hits | 6839 | 5768 | 9128 | 9315 | 9125 | 7857 | 9475 |
| bpred\_bimod.misses | 3501 | 4572 | 1212 | 1025 | 1215 | 2483 | 865 |
| bpred\_bimod.jr\_hits | 832 | 832 | 815 | 815 | 815 | 815 | 815 |
| 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 | 19 | 19 | 19 | 19 | 19 |
| bpred\_bimod.bpred\_addr\_rate | 0.6614 | 0.5578 | 0.8535 | 0.8724 | 0.8524 | 0.7293 | 0.8923 |
| bpred\_bimod.bpred\_dir\_rate | 0.6614 | 0.5578 | 0.8828 | 0.9009 | 0.8825 | 0.7599 | 0.9163 |
| bpred\_bimod.bpred\_jr\_rate | 1.0000 | 1.0000 | 0.9796 | 0.9796 | 0.9796 | 0.9796 | 0.9796 |
| bpred\_bimod.bpred\_jr\_non\_ras\_rate.PP | 1.0000 | 1.0000 | 0.2105 | 0.2105 | 0.2105 | 0.2105 | 0.2105 |
| bpred\_bimod.retstack\_pushes | 0 | 0 | 815 | 815 | 85 | 815 | 815 |
| bpred\_bimod.retstack\_pops | 0 | 0 | 813 | 813 | 813 | 813 | 813 |
| bpred\_bimod.used\_ras.PP | 0 | 0 | 813 | 813 | 813 | 813 | 813 |
| bpred\_bimod.ras\_hits.PP | 0 | 0 | 811 | 811 | 811 | 811 | 811 |
| bpred\_bimod.ras\_rate.PP | Error:divide by zero | Error:divid byzero | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |

(3) test-printf

./sim-bpred -bpred taken tests/bin.little/test-printf

./sim-bpred -bpred nottaken tests/bin.little/test-printf

./sim-bpred -bpred bimod -bpred:bimod 512 tests/bin.little/test-printf

./sim-bpred -bpred bimod -bpred:bimod 1024 tests/bin.little/test-printf

./sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 01 tests/bin.little/test-printf

./sim-bpred -bpred 2lev -bpred:2lev 1 64 6 1 tests/bin.little/test-printf

./sim-bpred -bpred comb -bpred:comb 1024 -bpred:2lev 1 1024 8 0 -bpred:bimod 1024 tests/bin.little/test-printf

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | always taken | always not taken | bimod  (512) | bimod  (1024) | 2-level  (1,1024,8,0) | 2-level  (1,64,6,1) | comb  (1024)  (1,1024,8,0)  (1024) |
| sim\_total\_insn | 1813877 | 1813877 | 1813877 | 1813877 | 1813877 | 1816877 | 1813877 |
| sim\_total\_refs | 516941 | 516941 | 516941 | 516941 | 516941 | 516941 | 516941 |
| sim\_num\_branches | 401609 | 401609 | 401690 | 401609 | 401690 | 401609 | 401609 |
| sim\_elapsed\_time | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| sim\_inst\_rate | 1813877 | 1813877 | 1813877 | 1813877 | 1813877 | 1813877 | 1813877 |
| sim\_IPB | 4.5165 | 4.5165 | 4.5165 | 4.5165 | 4.5165 | 405165 | 4.5165 |
| bpred\_bimod.lookups | 401609 | 401609 | 401690 | 401690 | 401690 | 401690 | 401609 |
| bpred\_bimod.updates | 401609 | 401609 | 401690 | 401690 | 401690 | 401690 | 401609 |
| bpred\_bimod.addr\_hits | 261363 | 233361 | 372591 | 377458 | 372700 | 323623 | 384151 |
| bpred\_bimod.dir\_hits | 261363 | 233361 | 374901 | 379761 | 375009 | 325966 | 386385 |
| bpred\_bimod.misses | 140246 | 168248 | 26708 | 21848 | 26600 | 75643 | 15224 |
| bpred\_bimod.jr\_hits | 31998 | 31998 | 30152 | 30152 | 30152 | 30152 | 30152 |
| bpred\_bimod.jr\_seen | 31998 | 31998 | 31998 | 31998 | 31998 | 31998 | 31998 |
| bpred\_bimod.jr\_non\_ras\_hits.PP | 31998 | 31998 | 363 | 363 | 363 | 363 | 363 |
| bpred\_bimod.jr\_non\_ras\_seen.PP | 31998 | 31998 | 2198 | 2198 | 2198 | 2198 | 2198 |
| bpred\_bimod.bpred\_addr\_rate | 0.6508 | 0.5811 | 0.9277 | 0.9399 | 0.9280 | 0.8058 | 0.9565 |
| bpred\_bimod.bpred\_dir\_rate | 0.6508 | 0.5811 | 0.9335 | 0.9456 | 0.9338 | 0.8117 | 0.9621 |
| bpred\_bimod.bpred\_jr\_rate | 1.0000 | 1.0000 | 0.9423 | 0.9423 | 0.9423 | 0.9423 | 0.9423 |
| bpred\_bimod.bpred\_jr\_non\_ras\_rate.PP | 1.0000 | 1.0000 | 0.1652 | 0.1652 | 0.1652 | 0.1652 | 0.1652 |
| bpred\_bimod.retstack\_pushes | 0 | 0 | 29802 | 29802 | 29802 | 29802 | 29802 |
| bpred\_bimod.retstack\_pops | 0 | 0 | 29800 | 29800 | 29800 | 29800 | 29800 |
| bpred\_bimod.used\_ras.PP | 0 | 0 | 29800 | 29789 | 29800 | 29800 | 29800 |
| bpred\_bimod.ras\_hits.PP | 0 | 0 | 29789 | 29789 | 29789 | 29789 | 29789 |
| bpred\_bimod.ras\_rate.PP | Error:divide by zero | Error:divide by zero | 0.9996 | 0.9996 | 0.9996 | 0.9996 | 0.9996 |

根据得到的统计数据得知，在上述的分支预测器配置中，复合式的地址预测命中率与分支方向预测命中率最高。

五、代码分析

在sim-bpred.c中检查了输入的动态预测方法选项，并创建了相应的分支预测器。

sim\_check\_options(struct opt\_odb\_t \*odb, int argc, char \*\*argv)函数中：

 if (!mystricmp(pred\_type, "taken"))

  {//创建预测器实例

**bpred = bpred\_create(BPredTaken, 0, 0, 0, 0, 0, 0, 0, 0, 0);**

  }

在sim\_main函数中：

 while (TRUE){

  regs.regs\_R[MD\_REG\_ZERO] = 0;

#ifdef TARGET\_ALPHA

    regs.regs\_F.d[MD\_REG\_ZERO] = 0.0;

#endif

   /\* 获取下一条指令\*/

   MD\_FETCH\_INST(inst, mem, regs.regs\_PC);

     /\* 指令计数 \*/

   sim\_num\_insn++;

   /\* set default reference address and access mode \*/

   addr = 0; is\_write = FALSE;

   /\* set default fault - none \*/

   fault = md\_fault\_none;

   /\* 对指令译码 \*/

   MD\_SET\_OPCODE(op, inst);

     /\* 执行指令 \*/

   switch (op){

#define DEFINST(OP,MSK,NAME,OPFORM,RES,FLAGS,O1,O2,I1,I2,I3)

     case OP:

         SYMCAT(OP,\_IMPL);

         break;

#define DEFLINK(OP,MSK,NAME,MASK,SHIFT)

     case OP:

         panic("attempted to execute a linking opcode");

#define CONNECT(OP)

#define DECLARE\_FAULT(FAULT)

         { fault = (FAULT); break; }

#include "machine.def"

         default:

         panic("attempted to execute a bogus opcode");

}

    if (fault != md\_fault\_none)

fatal("fault (%d) detected @ 0x%08p", fault, regs.regs\_PC);

    if (MD\_OP\_FLAGS(op) & F\_MEM){

     sim\_num\_refs++;

     if (MD\_OP\_FLAGS(op) & F\_STORE)

       is\_write = TRUE;

    }

    if (MD\_OP\_FLAGS(op) & F\_CTRL){

      md\_addr\_t pred\_PC;

      struct bpred\_update\_t update\_rec;

  sim\_num\_branches++;

if (pred)// 如果分支预测器创建成功

      {

       /\* 获取预测的下条指令的地址 \*/

         pred\_PC = **bpred\_lookup**(pred,

                                         /\* 分支地址\*/regs.regs\_PC,

                                          /\*目的地址 \*/target\_PC,

                                          /\* 指令操作码 \*/op,

                                          /\* call? \*/MD\_IS\_CALL(op),

                                          /\* return? \*/MD\_IS\_RETURN(op),

                                          /\* stash an update ptr \*/&update\_rec,

                                          /\* stash return stack ptr \*/&stack\_idx);

   /\* 判断从分支预测器返回的下条指令地址是否合法 \*/

   if (!pred\_PC)// 不合法,(当返回0时,表示采用分支不转移预测)

   {

    /\* 分支不转移,pc直接加一 \*/

    pred\_PC = regs.regs\_PC + sizeof(md\_inst\_t);

   }

   /\* 根据指令执行的实际结果,来更新分支预测器\*/

    bpred\_update(pred,

  /\* 分支地址 \*/regs.regs\_PC,

/\* resolved branch target \*/regs.regs\_NPC,

  /\* 分支是否转移 \*/regs.regs\_NPC != (regs.regs\_PC +sizeof(md\_inst\_t)),

  /\* pred taken? \*/pred\_PC != (regs.regs\_PC +sizeof(md\_inst\_t)),

  /\* correct pred? \*/pred\_PC == regs.regs\_NPC,

  /\* opcode \*/op,

  /\* predictor update pointer \*/&update\_rec);

   }

 }

 /\* check for DLite debugger entry condition \*/

 if (dlite\_check\_break(regs.regs\_NPC, is\_write ? ACCESS\_WRITE : ACCESS\_READ,

 addr, sim\_num\_insn, sim\_num\_insn))

   dlite\_main(regs.regs\_PC, regs.regs\_NPC, sim\_num\_insn, &regs, mem);

      /\* go to the next instruction \*/

      regs.regs\_PC = regs.regs\_NPC;

      regs.regs\_NPC += sizeof(md\_inst\_t);

      /\* finish early? \*/

      if (max\_insts && sim\_num\_insn >= max\_insts)

return;

    }

六、实验总结

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,加深了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。

此次实验中遇到了许多问题，但最后都成功解决，锻炼了我解决问题的能力及动手能力。