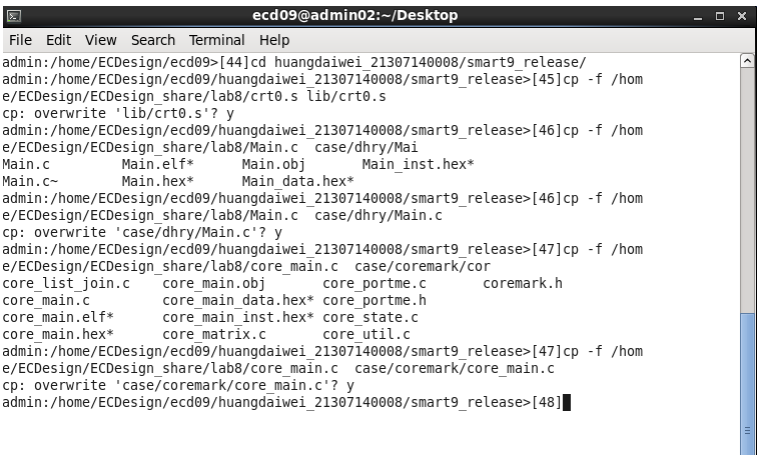
Pipeline 实验

一、实验目的

在 SMART 平台上通过调整分支预测配置情况观察测试程序程序的 CPI、分支预测准确率等，了解各种配置对 CPU 的性能的影响。

二、实验步骤（包括实验结果，数据记录，截图等）

（一）更改以及替换 SMART 平台内对应的文件，包括 crt0.s，dhrystone 程序和 coremark 程序。



（二）在启动文件 crt0.s 中选择分支预测的配置，并进行 dhrystone 程序和 coremark 程序的仿真。

- 用 gvim 打开 crt0.s

```
admin:/home/ECDesign/ecd01/smart9_release>[54]gvim lib/crt0.s
```

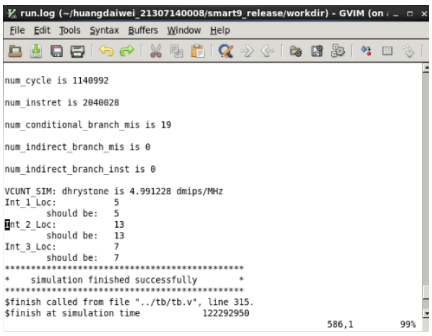
- 取消某配置的注释，并保留其他注释即表示对该配置进行测试。对以下 4 种分支预测分别进行 dhrystone 和 coremark 测试

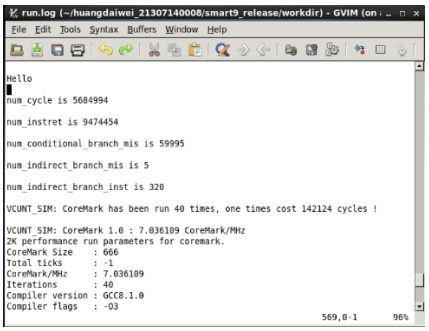
```
#-------------------------------------------------------------------------------
#-----------------select the branch prediction configuration here---------------
#-------------------------------------------------------------------------------
  # mhcr 4-RS, 5-BPE, 6-BTB, 7-IBPE, 12-L0BTB
  # reserve the low 3 bits asserted, select 1 of 4 at the same time

  li x3, 0x10f7     #all prediction on
  #li x3,0x0007      #all prediction off
  #li x3,0x00b7      #BTB,L0BTB off
  #li x3,0x10d7      #BPE off
```

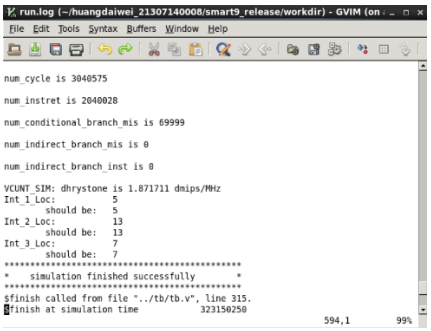
- 在 SMART 平台的主目录（smart9_release）下，执行 **source setup.csh**
  执行 cd workdir 进入 workdir 目录
  执行 run ../case/dhry/Main.c（对 dhrystone 进行测试），
  或 run ../case/coremark/core_main.c（对 coremark 进行测试）

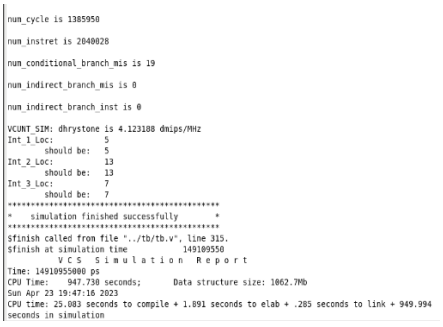
  以下分别为 dhrystone 和 coremark 结果。

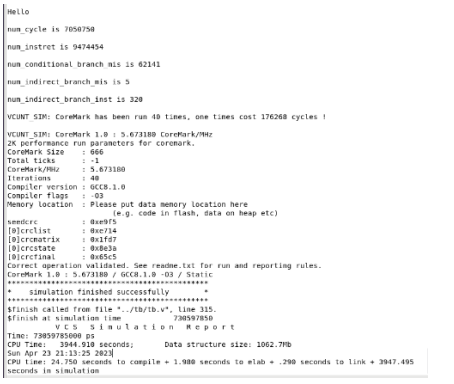
  dhrystone　　coremark

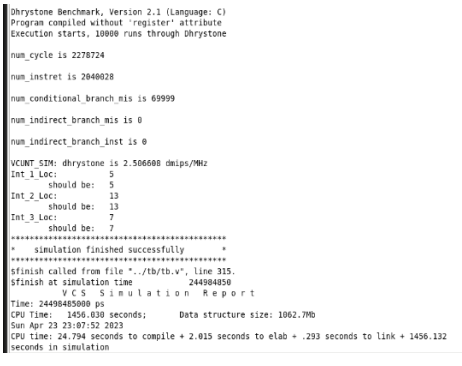


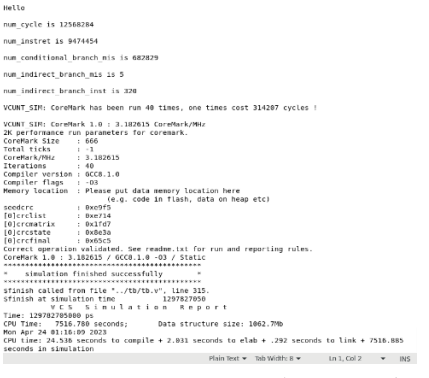












（三）完成分支预测配置下的 dhrystone 程序和 coremark 程序的仿真后，观察仿真结果，记录数据，汇总成上述的两张表格。

表格 1 dhrystone **测试**

	all prediction on	all prediction off	BTB,LOBTB off	BPE off
cycle	1140992	3040575	1385950	2278724
insts	2040028	2040028	2040028	2040028
CPI	0.599302	1.490457	0.679378	1.117006
conditional branch miss	19	69999	19	69999
indirect branch	0	0	0	0

miss				
indirect branch inst	0	0	0	0
DMIPS(dmips/MHz)	4.991228	1.871711	4.123188	2.506608

表格 2 coremark 测试

	all prediction on	all prediction off	BTB,LOBTB off	BPE off
cycle	5684994	12875906	7050750	12568284
insts	9474454	9474454	9474454	9474454
CPI	0.600034	1.359013	0.744185	1.326544
conditional branch miss	59995	682829	62141	682829
indirect branch miss	5	320	5	5
indirect branch inst	320	320	320	320
CoreMark point (CoreMark/MHz)	7.036109	3.106584	5.673180	3.182615

三、实验分析和总结

（一）实验分析

1.数据分析

- 针对以上表格我们可以看到：无论是 dhrystone 或 coremark 测试

  (1). 在相同的指令数下，四种配置所经历的周期数不同，从少到多（即 CPU 性能从强到弱）依次是 all prediction on < BTB, LOBTB off < BPE off < all prediction off;

  (2). 在相同的指令数下，四种配置的 CPI 不同，从小到大的顺序同上

  (3). 在相同的指令数下，四种配置的 DMPIS/CoreMark point 不同，从大到小顺序同上;

  以上说明开启分支预测时的性能要远优于不开启时的，符合我们的认知与预测。

- 另外，在 dhrystone 测试中：

  如果只将 BTB,LOBTB(分支目标预测)关闭，CPI 会从 0.599 变为 0.679 ；如果只将 BPE(允许预测跳转)关闭，CPI 会从 0.599 变为 1.117，并且条件分支的 miss 数会从 19 变为 69999，与预测全部关闭时的 miss 数相同。

  这说明 BPE 在性能提升和条件分支命中率上有着重要作用。

  在间接预测中，dhrystone 的数据均为 0，是与测试程序有关，其中并没有对间接分支目标地址的预测。

- 在 coremark 测试中：

  如果只将 BTB,LOBTB(分支目标预测)关闭,对 CPI 的影响不会太大,如果只将 BPE(允许预测跳转)关闭，CPI 会从 0.600 变为 1.327，接近分支预测全部关闭时的 1.359，并且条件分支的 miss 数与预测全部关闭时相同。

  这说明 BPE 仍在性能提升和条件分支命中率上占主要作用

而在间接分支预测命中率上，关闭BTB,LOBTB或者BPE都没有对miss数产生影响，因此对于间接分支的预测并非 BTB 或者 BPE 中的任意一个；

2.原因分析

(1) 现代CPU使用流水线的方式运行程序，使得程序运行效率显著提高。但是当遇到分支语句时，若没有分支预测器，后级需要等待前级流水线计算出分支的跳转位置与方式才可以进一步操作,则会导致流水线停顿，程序运行速度减慢。因此运用分支预测的方式，可以对分支语句进行预判断，在语句执行前判断哪一个分支将会被选中，减少程序判断跳转的延时，以达到加快程序运行，提高CPU性能的功能。

(2) 在C910中，总共存在四种分支预测器，分别是：分支历史表（BHT）、分支跳转目标预测器(BTB)、间接分支预测器和快速跳转目标预测器。它们的工作方式分别为：

- BHT：分支历史表，记录着分支历史信息，用指令的PC值的后12位进行索引，通过历史记录来预测本次是否跳转。当前BHT准确率可达百分之九十以上，但缺点是效率较低。
- BTB：用于记录一条指令及其跳转地址。特点是存储容量较小，但是一旦索引完成可迅速跳转，效率较高；
- 间接分支预测器：是指操作数不是要转去的下一条指令地址，而是一个存储位置（寄存器或者内存），这个存储位置的内容才是要转去的下一条指令地址。通过这种方法可以选择多于两条分支的执行路线。
- 快速跳转目标预测器：在发现指令可能引起连续跳转时记录连续跳转第二条指令的地址和跳转的目标地址。

(3) 分支预测器可以减少周期数，周期数越少，CPI 越小，单位时间内能够运行的dhrystone指令条数越多，运行coremark程序的时间越短（跑分越高）。并且成功预测的分支越多，减少的周期数越多。同时，由于BTB的容量远小于BHT，因此BHT检索命中率远高于BTB，BHT的存在对性能的提升更大。

(4) 同样，由于BHT的容量远大于BTB，BHT对于条件分支的影响也会更大。因此在相同的指令数下，条件分支语句的检索主要都在BHT中得以命中，关闭BHT将导致条件分支语句检索的大量失败(大量miss)

(5) 在dhrystone中,没有间接分支语句故对于间接分支语句的检索应该分析coremark可以看出，由于BHT与BTB都不支持间接分支检索，因此关闭其中的任意一个对于间接分支语句检索的命中率都没有太大的影响，只要间接分支预测期仍在运行，那么对于间接分支语句的命中率都是很高的

（二）实验总结

综上所述，分支预测期可以提高CPU的性能，具体表现在CPI的下降、周期数的减少，跑分的提高等。在BTB与BHT中，即使BTB的效率更高，但由于BHT的检索容量更大，因此命中率更高，对CPU性能的影响更高。另外，对于间接分支条件语句，主要由间接分支预测器负责预测，BTB或BHT对其没有直接影响。

从以上实验可以看出各种分支预测的分工以及优缺点。因此在现代工业中，出现了融合分支预测器（如Tournament，即整体局部自适应预测器），将局部预测与全局预测相结合，采用多个预测器相结合的方式，为特定的转移选取正确的预测器。同时，也可以将BHT与BTB相结合，采用BTB快速查找，BHT查漏补缺的方式，集合二者速度快与容量大的优点，达到性能的最优化。

四、实验收获、存在问题、改进措施或建议等

通过本次实验，我对于分支预测的原理与必要性有了更进一步的理解，也具体感受到了分支预测对于 CPU 性能的提升。同时通过对于不同配置下的 dhrystone 与 coremark 的实验，比较了不同分支预测器对性能不同的影响，也进一步了解了当下 CPU 的分支预测配置。