Lab 4 Report

PB20111623 马子睿

实验内容

- 实现动态分支预测器，分别采用BTB的1bit感知和BTB+BHT的2bit感知进行分支预测
- 统计分支预测对流水线性能的影响，并做出定量分析

实验设计

- BTB表项的设计

  首先，我定义了如下几个表项：

  ```
  // BTB data table
  reg [31:0]              btb_predict_pc [0:BTB_SET-1];
  // BTB tag table
  reg [BTB_TAG_WIDTH-1:0] btb_branch_tag [0:BTB_SET-1];
  // BTB valid table
  reg                     btb_valid [0:BTB_SET-1];
  // BTB history table
  reg                     btb_history [0:BTB_SET-1];
  ```

  这几个表项分别用来记录**分支跳转目标地址、跳转指令所在地址的高位、有效位和最近一次跳转历史**。所有表项都会在**EX阶段进入一条分支指令**时更新，其中，历史位会写入这条跳转指令本次真实的跳转情况。

  在读取时，BTB会首先根据tag表来确定是否命中，如果命中，那么就会根据历史表进行预测，并给出对应的跳转地址。如不命中，那么会给出PC+4的值。

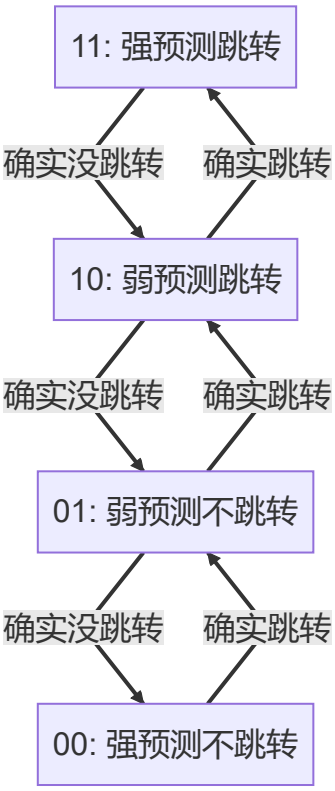
  在我的设计中，BTB表共计32项。

- BHT的设计

  在这里我采用了一个表来记录所有的状态机：

  ```
  reg [1:0] bht_state [0:BHT_SET-1];
  ```

  BHT表会使用PC的地址低位进行索引（低位指除去了最后两位的低位），每个感知器由4个状态构成：



除此以外，这里还有一个实用的技巧：**复位时将所有状态复位为01**，这样感知器可以迅速进入学习状态。

BHT同样会在一条分支指令进入EX阶段时更新对应表项。但原来进入PC级的是这条指令地址+4，这无法准确的更新表项，因此应该将EX段PC-4作为写地址。

- 冲突冒险处理：

  现在的实验设计中，我会将分支预测器给出的预测地址传入EX段，再从EX段和真实分支结果一起反馈回分支预测器，用分支预测器判断是否预测成功，并相应的生成pre_fail信号。这个信号将代替br信号，在Hazard里发起flush信号。

Branch History Table

BTB	BHT	REAL	NPC_PRED	flush	NPC_REAL	BTB_update
Y	Y	Y	BUF	N	BUF	N
Y	Y	N	BUF	Y	PC_EX+4	N
Y	N	Y	PC_IF+4	Y	BUF	N
Y	N	N	PC_IF+4	N	PC_EX+4	N
N	Y	Y	PC_IF+4	Y	BR_TARGET	Y
N	Y	N	PC_IF+4	N	PC_EX+4	Y
N	N	Y	PC_IF+4	Y	BR_TARGET	Y
N	N	N	PC_IF+4	N	PC_EX+4	Y

分支收益和分支代价

- BTB+1bit感知器预测：

    - 收益：记录下了跳转指令的跳转地址，从而可以根据最后一次的跳转历史来进行预测，对于循环的最后一天分支指令的预测非常有效
    - 代价：由于分支预测必须要当周期给出结果，因此所有的存储器都是异步读，在消耗资源的同时也更消耗了时序
- BTB+BHT 2bit感知器预测：
    - 收益：对于多重循环而言，内层循环的最后一条分支指令可以在多次进入循环时预测正确，改进了内层循环在1bit预测时的“抖动”。
    - 代价：需要实现一个状态机表，当出现地址低位冲突时，很有可能增大分支预测代价。

性能测试实验数据记录

QuickSort

预测方法	总周期数	总执行分支指令数	分支预测正确次数	分支预测正确率
不使用分支预测	112284	6981	5320	0.762
使用btb预测	112692	6981	5115	0.733
使用bht预测	111210	6981	5856	0.839

MatMul

预测方法	总周期数	总执行分支指令数	分支预测正确次数	分支预测正确率
不使用分支预测	350290	4624	274	0.059
使用btb预测	342684	4624	4076	0.881
使用bht预测	342144	4624	4346	0.940

btb

预测方法	总周期数	总执行分支指令数	分支预测正确次数	分支预测正确率
不使用分支预测	509	101	1	0.010
使用btb预测	311	101	99	0.980
使用bht预测	311	101	99	0.980

bht

预测方法	总周期数	总执行分支指令数	分支预测正确次数	分支预测正确率
不使用分支预测	535	110	11	0.100
使用btb预测	379	110	88	0.800
使用bht预测	361	110	97	0.882

实验分析

- 对于btb测试主要是使用单层循环进行测试。在这种情况下不使用分支预测自然会出现众多的错误，而使用1bit或2bit的效果一定是一致的，因为预测错误只有可能出现在第一次跳转和最后一次不跳转上
- 对于bht测试则不然，使用1bit来感知内层循环就会导致内层循环在上一次结束到下一次开始的两次分支预测一定会失败，但显然，上一次内层循环结束后，下一次依然应该预测这个指令是跳转的，而2bit感知器就解决了这个问题。可以看到，btb预测和bht预测相差9次，而这九次全部是由上文所述的问题造成的。这里也就可以清晰看出加入了2bit感知器可以很好地解决了多重循环预测的问题
- 多重循环和交替跳转的问题非常深刻地体现在了快速排序中。快速排序是典型的循环嵌套，而且其中Partition找位置的分支变化的非常频繁，指针经常没有移位几次就找到了对应的位置，因此1bit的分支预测效果已经差到了极点——交替跳转甚至令其命中率不及不预测的分支预测器！但此时，2bit感知器非常好地解决了多重循环和交替跳转的问题，因此获得了极好的命中效果。
- 对于矩阵乘法，由于其中的大循环循环次数非常多，因此有分支预测一定强于没有分支预测。但同时，由于矩阵乘法依然存在循环嵌套，故2bit感知器的效果也一定好于1bit感知器。
- 综上所述：造成2bit感知器和1bit感知器性能区别的主要体现在了多重循环和交替跳转上，而1bit预测和不预测的性能区别主要取决于程序中交替跳转情况发生的多寡上。

实验总结

通过本次实验，我有了以下收获：

- 全面认识了简单的分支预测策略，并能够结合具体测试分析不同策略的优劣和性能影响
- 熟悉了几种简单的分支预测器的设计方法