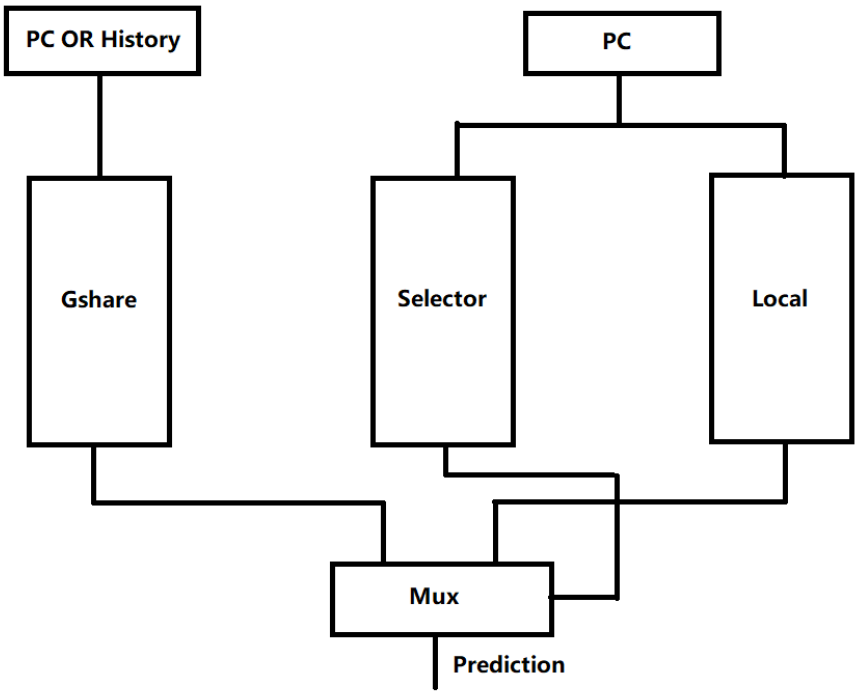
Mips分支预测实验报告

实验目的

为Mips流水线处理器实现分支预测部件Predictor，对分支跳转语句进行预测，降低CPI

Predictor结构



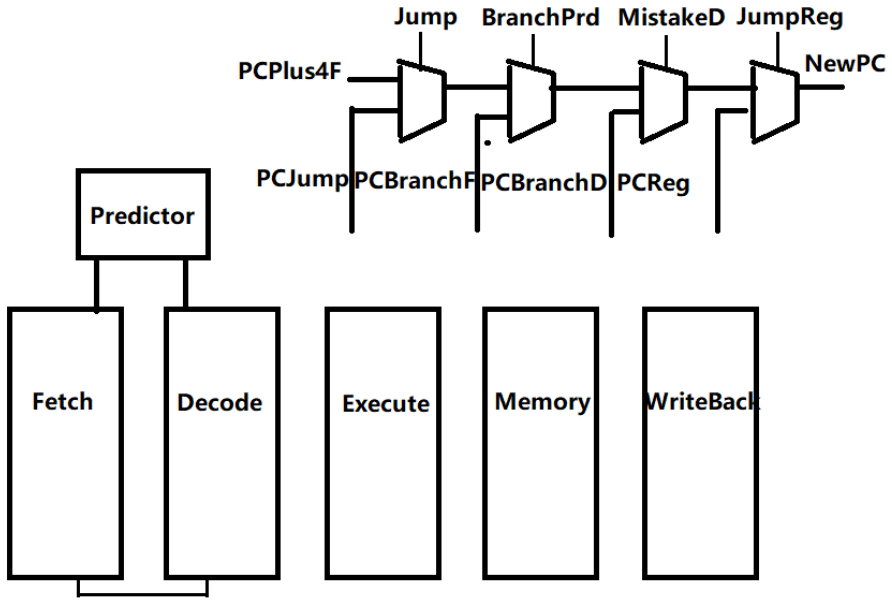
主要由三部分组成：基于当前PC的分支预测Local，基于当前PC和历史信息的分支预测Gshare和基于是否预测成功的选择器Selector

作用机制

输入信号：Fetch器件的PC值PCF、Decode器件的PC值PCD、Fetch器件的指令是否是分支跳转isBranchF、Decode器件的指令是否是分支跳转isBranchD、Decode器件得到的跳转位置real_addr、Decode器件得到的分支是否执行real_taken、Decode器件得到的分支预测是否错误MistakeD

输出信号：Fetch器件的指令是否需要跳转prd、Fetch器件的指令跳转的位置prd_addr

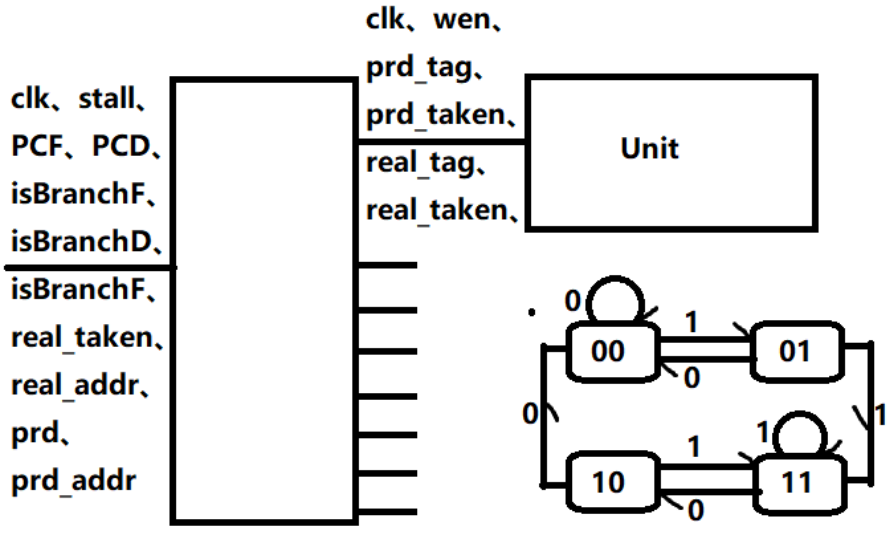
改动后的流水线处理器结构：



Decode器件需要向Fetch器件输出分支预测是否正确的信号MistakeD，Predictor分别从Fetch和Decode器件输入、输出相应的信号。同时，Fetch器件中新的PC值得选择逻辑部分也应该进行相应得改动（如图右上角所示）

各部件结构

Local



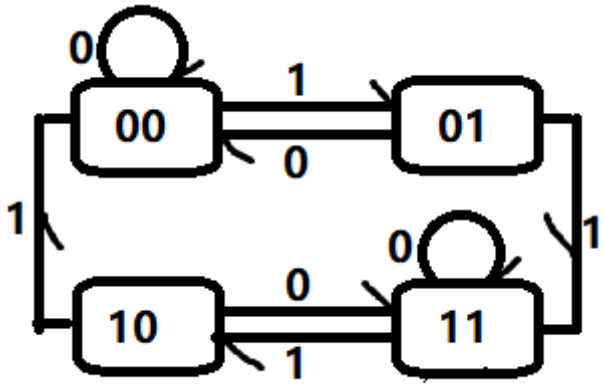
如图，在Local部件中包含8个Unit部件。在Unit中，实现了2位的有限状态机。当时时钟上升沿到来时，根据real_taken信号转移状态，转移策略如图所示。把2位状态值的高位作为是否跳转的预测值。

分别取输入的PCF、PCD后三位作为其对应的Unit入口。把PCF所在Unit的输出信号得到是否跳转的预测试，并通过比较其tag判断PCF在Unit是否命中。如不命中，则当作不跳转来处理。把PCD所在的Unit作为Decode部件输出信号的更新目标。比较该Unit的tag和PCD，若命中则根据上述状态转移策略进行状态转移，否则就重置状态为00，并更新该Unit的tag。

Gshare

Gshare部件和Local部件采取相同的结构和逻辑策略，另外增加ResF、ResD分别记录Fetch、Decode之前是否跳转的历史记录（取3次跳转）。在Local部件中PCF、PCD所对应的Unit编号为其后3位，在该部件中PCF、PCD所对应的Unit编号为其后3位的值和对应跳转历史（3位）执行按位或运算之后的数值。

Selector



采取和Local部件相似的结构。但Unit的转移信号变为分支预测是否错误的MistakeD信号，转移策略如图所示。

分别取输入的PCF、PCD后三位作为其对应的Unit入口。把PCF所在Unit的输出信号作为是否选择Gshare的信号。若是0则选择Local，否则选择Gshare。把PCD所在的Unit作为Decode部件输出信号的更新目标。比较该Unit的tag和PCD，若命中则根据上述状态转移策略进行状态转移，否则就重置状态为00，并更新该Unit的tag。

测试结果

无分支预测：2.207687

Local分支预测： 1.754276

Gshare分支预测： 1.887972

Predictor总和分支预测：1.756438

总结

Local仅仅根据分支本身的跳转历史记录来进行预测，Gshare则在此基础上又加入了最近几条分支的跳转记录。但是，由于Unit入口数量的限制，Gshare会产生较多的不命中，无法很好的进行预测。而且，在给定的测试集中，并没有出现相互之间有联系的分支跳转，导致Gshare的分支预测表现不佳。最后，为了能兼顾两者的优点，加入了另一个预测部件，来判断每句分支具体应该选哪个部件的输出当作最终的结果。

在*Computer Architecture A Quantitative Approach*一书中也提到了使用级联式的Gshare部件，分别求得兼顾过去1条、2条等分支的预测部件的结果，找到兼顾过去最多条分支且命中的部件的预测结果，作为最终的输出。由于时间较紧，且实现的思路也和本结构的思路相似，就没有实现。

参考资料

Computer Architecture A Quantitative Approach