UNICORE 体系结构中动态转移预测机制的研究与设计

朱德新, 程 旭, 慎 辉
(北京大学微处理器研究开发中心, 北京 100871)

摘 要: 本文采用投合预测器(Agree predictor)的设计思想扩展 UNICORE 体系结构, 旨在评测转移预测器中各项配置对单发射流水线结构的性能影响, 为 UNICORE 体系结构改进提供定量依据. 实验基于系统级模拟器, 综合转移预测策略和转移目标缓冲器行为进行完整模拟, 结论对于其它采用单发射流水线结构的微处理器设计具有较好的借鉴意义.

关键词: 指令系统体系结构; 流水线; 转移预测
中图分类号: TP30217 文献标识码: A 文章编号: 0372-2112 (2004) 08-1351-05

Dynamic Branch Prediction Research and Design for UNICORE Architecture

ZHU De-Xin, CHENG Xu, SHEN Hui
(Microprocessor Research and Development Center, Peking University, Beijing 100871, China)

Abstract: We introduced agree predictor into UNICORE architecture to evaluate the performance enhancement brought by the configuration of dynamic branch predictor, in order for giving quantitative reference for UNICORE reconstruction. Using system simulator, we synthetized the effect of the prediction scheme and the branch target buffer. The conclusion is useful to the single-issue-pipeline microprocessor design.

Key words: instruction set architecture; pipeline; branch prediction

1 引言

在高性能计算机系统中, 设计人员通常采用转移预测技术来开发指令级并行性, 并降低控制冒险带来的性能损失. 转移预测技术和转移目标缓冲器(BTB)联合, 可以有效地支持推测式执行, 提高处理器性能. 同延迟转移(Delayed branch)策略相比, 转移预测在采用深流水和多发射机制的处理器中优势更加明显.

转移预测技术分为静态预测和动态预测两类. 静态预测通常采用的方法有预测转移总是发生(或不发生)、或者根据程序制导信息、剖视文件(profile)、程序结构获取转移偏向性进行预测; 动态预测则基于程序执行过程中的转移历史信息预测当前指令的转移方向, 由处理器中的专用部件实现预测功能. 动态转移预测方法通常具有更高的正确率, 替换预测策略时不需要修改指令系统和已经存在的可执行程序.

目前, 动态转移预测相关研究多是针对多发射体系结构开展的, 面向单发射处理器(Scalar processor)结构的研究较少. 另一方面, 在嵌入式应用领域中, 由于成本、芯片面积和功耗因素的限制, 单发射处理器核仍然是嵌入式系统设计时的首选. 因此, 研究单发射流水线中的动态转移预测技术具有重要的现实意义. 事实上, 目前有部分嵌入式微处理器已经开始采用这些技术, 例如 ARM11 处理器中添加了动态转移预测模块, 但是有关的具体设计和定量评测还不充分. 而且, 目前的动态转移预测研究侧重于如何提高预测正确率, 而面向实际机器将预测策略和 BTB 联合进行性能评测的研究较少, 研究手段多采用踪迹驱动的方法, 不能真实地反映处理器的运行情况. 在本文中我们将面向实际机器, 考察动态转移预测策略和 BTB 对处理器性能的整体影响. 我们基于系统级模拟器进行实验, 采用程序驱动的评测方法, 可反映操作系统行为, 因此实验数据更加准确、可信, 其结论对于以 UNICORE* 为代表的、采用单发射流水线实现的微处理器设计具有较好的借鉴意义.

本文在第 2 节介绍动态转移预测的相关策略; 第 3 节介绍实验基础和模拟环境; 第 4 节针对 UNICORE 体系结构中添加的动态转移预测机制进行性能评测, 最后给出结论.

收稿日期: 2003-09-29; 修回日期: 2004-04-29
基金项目: 国家高技术研究发展计划(863 计划)课题(No. 2002AA1Z1010, No. 2002AA1Z2203)

* UNICORE 是由北京大学微处理器研究开发中心自主研制的一款 RISC 风格的 32 位微处理器, 目前已经过流片测试, 应用在北大众志[0] 网络计算机系统中.

2 动态转移预测策略

实现动态转移预测的一种方法是为最近发生跳转的指令建立缓存,预测时查询缓存. 另一种预测方法是基于指令的上次运行结果,在数据通路中建立转移历史表 BHT(Branch history table) 记录所有转移指令的执行情况. BHT 结构类似于 cache,需要存放指令标记位和 1 位计数器. 类似的,设计人员也可以直接在指令 cache 中附加 1 位计数器实现与 BHT 相同的功能[1].

对于构成循环的转移指令使用 1 位计数器进行预测,会导致程序在第一次进入和退出循环时预测失败. 而如图 1 采用 2 位计数器只失败一次,这种计数器被称为 2 位饱和计数器(Two bits saturating counter). 同理,可以设计 n 位饱和计数器,其值在[0, 2^{n-1} - 1] 范围时预测转移不发生,在[2^{n-1}, 2^n - 1] 范围内时预测转移发生. 相关的研究表明 2 位饱和计数器的预测效果通常最好[2],因此这种计数器目前被应用于各种复杂的预测策略中.

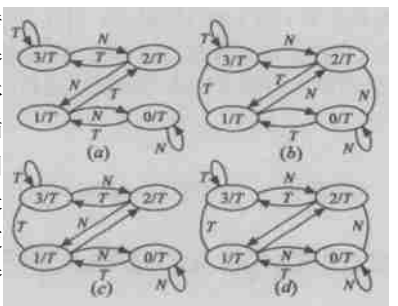

图 1 几种 2 位饱和计数器的状态转换图

根据转移历史信息不同,动态转移预测可分为局部预测和全局预测,前者根据特定指令的转移历史信息预测其将来转移行为,对于循环结构预测效果较好. 后者则以相邻转移指令行为作为预测根据,对于 if2then2else 结构以及部分循环嵌套预测正确率较高[3]. 全局预测时,不同的转移指令可能使用同一个 BHT 表项,即不同转移指令之间发生了/ 冲突0,如果转移指令分别往不同方向更新计数器,则会导致预测正确率降低,这种冲突被称为负面冲突(Destructive interference).

在全局动态转移预测中,负面冲突是制约预测正确率提高的主要因素. 相应的解决方案有增大转移历史信息表、使用更有效的索引方法、或根据转移指令特点分类以对应不同的 PHT 体,这些方案的基本思想都是从整体上降低 PHT 表项中存在的所有冲突(包括负面冲突) ,而投合预测器(Agree predictor)[4]通过设置偏向位将负面冲突转化为其它类型冲突,设计方案更加简练.

其它典型的预测策略还有双峰预测器(Bi2mode predictor)[5],强调根据指令偏向性动态划分转移指令为跳转和不跳转两类,并使用选择预测器进行仲裁; 扭斜预测器(Skewed predictor)[6]引入 cache 的 3C 模型对冲突进行分析,并借鉴扭斜相联高速缓存(Skewed associative cache) 的思想设置奇数个 PHT 体,通过索引函数保证转移指令不会在多于 1 个体内发生冲突,并通过投票方式决定最后的预测值; 过滤机制(Filtering mechanism)[7]将具有明显偏向性的转移指令过滤,从而减少 PHT 中存放的信息总量,降低发生负面冲突的几率. 其它的相关研究还包括使用静态、动态策略结合的方法,基于执行路径的预测等.

动态转移预测通常和 BTB 联合实现. BTB 结构与 cache 类似,其表项存放指令地址标签、目标地址、以及预测标记位. 进一步,BTB 中可以取代转移目标地址而存放转移目标地址处的指令,从而省略取指操作,并可以进行转移折叠(Branch folding) 优化. 在多发射体系结构中,BTB 表项中还可以存放转移目标处的多条指令.

3 实验基础和模拟环境

311 UNICORE 微处理器

UNICORE 是一款 32 位 RISC 风格的微处理器,支持 32 位和 16 位两套指令系统及多种操作模式运行,其数据通路采用单发射 5 级流水线设计并支持硬件互锁和前递功能. UNICORE 微处理器的流水线功能描述如表 1.

UNICORE 微处理器采用了 fall2through 和 squash 技术处理转移指令. 转移发生时流水线需要从目标地址重新取得指令,从转移指令进入译码阶段到取指后流水线重新执行,这一过程需要 3 个周期完成. 由于程序中存在大量的转移指令,因此这种方案对系统整体性能影响较大. 实验显示,在 UNICORE 体系结构下 SPEC95 和 SPEC2000 测试程序中转移指令的平均执行周期数为 2134,而整体 CPI 为 1.2 至 1.7 个周期数,转移指令的执行开销相对较大,因此通过动态转移预测方法来降低运行时转移开销,对于提高单发射流水线的性能非常重要.

表 1 UNICORE 流水线功能描述

符号	意义	行 为
IF	取指	从存储器取出下一条指令; PC= PC+ 4; IRd [IA]
ID	译码	32 位、16 位指令译码; 取寄存器操作数; 判断数据相关
EXE	执行	移位部件、算术部件执行相应操作; 为转移指令确定转移方向和目标地址
MEM	访存	等待存储器数据或写存储器数据; 回写状态寄存器 CPSR
WB	写回	结果写回寄存器

312 实验方案

基于投合预测器的思想,我们在 UNICORE 数据通路的取指阶段添加了转移预测部件及 BTB,图 2 为预测部件结构图.

图 2 中的 BHR 存储了全局转移历史信息,它与 PC 寄存器的异或运算结果作为 PHT 表索引,同时使用转移指令地址索引 BTB 查找转移目标地址. PHT 表项采用了 2 位饱和计数器设计,计数器的最高位与 BTB 中预置的偏向位的同或操作结果作为转移预测的最终结果.

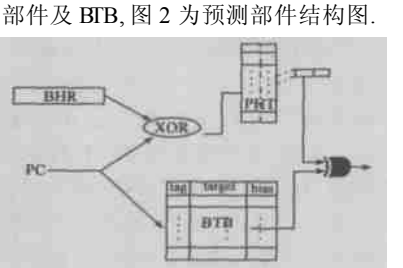

图 2 UNICORE 数据通路中实现的转移预测部件

BHR、BTB 和 PHT 缓存内容是动态更新的. UNICORE 数据通路可以在流水线执行阶段得到实际转移方向和目标地址,因此需要在访存阶段更新这些缓存. 其中 BHR 采用移位寄存

器实现,更新时在最低位记录转移是否发生. BTB 在转移指令第一次执行或 BTB 表项替换时进行更新,它的标签域(tag)写入指令地址高位,目标地址(target)域记录转移目标地址,偏向位(bias)域记录指令第一次是否发生跳转,该域不再动态更新. PHT 表项中的计数器在达到饱和并继续同向增长时不需要回写更新,否则如果实际转移方向与 BTB 偏向位相同,则计数器加 1,不同则减 1.

在实现动态预测器时,设计人员还需要考虑 PHT 表项中饱和计数器的初值设置. 由于计数器初值代表了对偏向位的/赞同0程度,所以基于偏向位设置正确的前提(因为不论使用剖视信息还是使用第一次跳转方向设置偏向位,都认为得到了转移的偏向性),初值应该选为 10_2 或 11_2,两者中选择 10_2 更合适,因为 10_2 在偏向位设置错误的情况下转化为/不赞同0的速度更快. 通过实验我们也验证了采用初值为 10_2 的效果最好.

设计动态预测策略中缓存模块的时候需要考虑到硬件实现的复杂度,因为采用 RAM 还是采用寄存器堆方案实现预测缓存对读写时序的影响是不同的. 在我们的模拟实验中,BTB 和 PHT 设计为可同时读、写,这在具体的硬件实现中是可行的[2].

以下是关于负面冲突的比较. 在传统的两级预测机制中[3],如果映射到同一个 PHT 表项的两条转移指令产生负面冲突,即分别反方向更新 2 位饱和计数器,则在图 2 中将转化为同方向更新,即都趋向于/赞同0偏向位,从而降低了负面冲突,提高了预测正确率. 可以使用下面的例子进行粗略对比. 假定被映射到同一个 PHT 表项的两条转移指令,其发生跳转的概率分别为 85% 和 15%,在传统的两级预测机制中,反方向更新 2 位饱和计数器的几率为(0.85 * 0.85) + (0.15 * 0.15) = 0.745. 而在图 2 中,如果偏向位选择得正确,则反方向更新计数器的几率为(0.85 * 0.15) + (0.15 * 0.85) = 0.255. 所以 UNICORE 中的动态预测实现方案将有效降低负面冲突.

3.3　模拟环境

Usim 是面向北大众志[0]网络计算机的系统级模拟器,包含对整点处理器 UNICORE、各种协处理器、MMU、cache、中断机制、外设部件的模拟. Usim 支持 Linux 操作系统和大型应用程序的运行及调试. 我们基于 Usim 模拟器并采用程序驱动的方法评测修改后 UNICORE 微处理器性能,分别针对 BHR 长度、BTB 和 PHT 大小、相联度等配置进行实验,并最终确定设计方案. 相对于其它动态转移预测研究,采用系统级模拟环境能够更加真实地反映机器实际执行情况,所得结论也更可信.

为了方便讨论,表 2 按照 BTB 是否命中、预测结果和实际运行结果是否相同将转移指令的各种执行情况分类,并结合硬件实现给出了相应的执行周期.

表 2　添加预测部件后的转移指令执行情况分类

分类	BTB 是否命中	预测结果	实际结果	执行周期
M1	Yes	Taken	Taken	1
M2	Yes	Taken	Not taken	3
M3	Yes	Not taken	Taken	3
M4	Yes	Not taken	Not taken	1
M5	No	Not taken	Taken	3
M6	No	Not taken	Not taken	1

4　实验结果

评测时,我们选取了 SPEC95 和 SPEC2000 测试程序中部分定点实例和浮点实例,输入规模为 Test 集,使用面向 UNICORE 体系结构的 GCC 编译器 O2 选项进行编译. 测试程序信息如表 3,其中第二、三、四列分别代表了程序在修改前的 UNICORE 体系结构下统计得到的静态指令数、动态执行的指令总数和转移指令动态执行数.

表 3　测试程序信息

SPEC95	静态指令数	动态指令总数	转移指令数
099.go	148,037	400,743,820	75,006,773
124.m88ksim	127,823	477,996,079	102,314,556
126.gcc	376,328	1,168,423,059	268,458,007
129.compress	99,541	13,169,928	1,658,181
130.li	174,804	902,301,738	195,842,513
132.ijpeg	128,791	475,102,875	63,132,247
134.perl	231,500	15,279,663	3,184,984
147.vortex	274,685	9,063,001,179	1,601,273,004
101.tomcatv	111,994	4,828,317,321	871,388,820
107.mgrid	116,094	5,693,110,142	66,187,792
145.fpppp	272,773	4,342,483,546	354,179,757
SPEC2000	静态指令数	动态指令总数	转移指令数
164.gzip	110,032	2,601,833,368	401,247,847
175.vpr	164,434	21,784,687,888	3,936,316,905
176.gcc	416,483	1,441,579,905	339,802,089
181.mcf	100,240	164,413,518	38,323,047
197.parser	120,448	3,610,491,690	822,986,833
255.vortex	275,163	9,047,485,398	1,585,575,907
256.bzip2	105,909	8,509,885,111	1,090,067,853
300.twolf	171,455	1,239,924,029	227,348,149

4.1.1　预测正确率分布

由于 UNICORE 处理器不存在多发射和推测式执行的情况,因此我们采用预测正确率而非两次错误预测之间的执行指令数作为评测标准. 传统的预测正确率定义为:

$$P = (C_{M_1} + C_{M_4}) / \sum_{i=1}^{4} C_{M_i}$$

其中 C_{M_i} 表示对应表 2 中 M_i 分类的转移指令动态执行数目. 我们重新定义转移预测正确率为:

$$P = (C_{M_1} + C_{M_4}) / \sum_{i=1}^{6} C_{M_i}$$

采用这种正确率定义方法可以反映 BTB 命中对预测正确率的影响. 图 3 是 SPEC95 和 SPEC2000 测试程序的预测正确率分布图,可以看到,预测正确率是和程序本身的行为密切相关的,不同的应用程

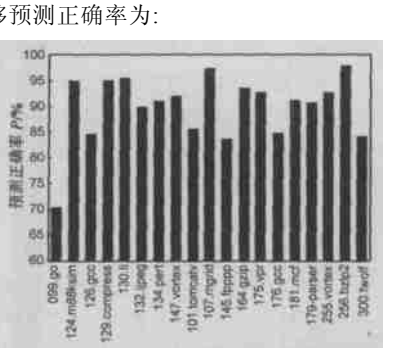

图 3　测试程序的预测正确率分布图

序预测正确率相差较大, 例如 SPEC95 中 099. go 程序的预测正确率较低, 这是因为程序中一半以上的转移指令没有明显的偏向性.

412 饱和计数器的状态转换对预测正确率的影响

PHT 表项中存储了 2 位饱和计数器, 用来强化或弱化 BTB 中记录的转移偏向性. 实现饱和计数器时可以采用图 1 中的某个有限状态机, 其中(b) 表示只有在连续两次实际都不发生跳转的情况下才预测下一次转移不发生, 其它情况下预测转移发生. (c) 、(d) 在状态机(a) 的基础上对偏向性信息的强化或弱化过程做了调整, 其中(c) 缩短了强化过程, (d) 将强化和弱化过程都缩短了. 对图 1 中给出的各种饱和计数器的状态转换图进行实验, 我们发现除(b) 的预测正确率较低以外, 采用其它三种状态机的 2 位饱和计数器其预测正确率相当, 其中(a) 的效果略好一些, 因此后续实验中的饱和计数器均采用了图 1(a) 所示有限状态机来实现.

413 BHR 长度对预测正确率的影响

增大 BHR 长度表示增加了更多的可用历史数据, 但预测效果同样是与程序本身特点相关的, 例如 SPEC95 中 go 程序的预测正确率会随 BHR 增大而降低, 因为 BHR 过大时, 部分干扰数据会被当做有效历史作为预测根据, 将会降低预测正确率, 因此我们在设计时最终选取 BHR 宽度为 8 位.

414 PHT 容量对预测正确率的影响

我们针对 PHT 表项个数为 64 至 2K 的各种情况进行了实验, 结果显示, 随着 PHT 增大, 转移预测的正确率升高, 这是由于 PHT 容量增大降低了负面冲突发生的几率. 鉴于 PHT 表项个数从 1024 变化至 2048 时, 部分程序的预测正确率变化不明显, 因此在最终的设计方案中我们选定 PHT 表项为 1024 个.

415 BTB 相联度对预测正确率的影响

BTB 失效率会影响预测正确率, 目前相关的转移预测研究很少考虑到 BTB 对预测正确率的影响, 我们的实验考虑到了这个因素. 实验显示, BTB 相联度由直接映射变化到两路组相联时对正确率有较大影响, 进而变化到四路组相联时对部分程序的预测正确率的提高影响并不大, 考虑到组相联实现的硬件开销, 我们在 UNICORE 系统中最终采用了 2 路组相联结构的 BTB, 置换策略采用最近最少使用策略(LRU) .

416 BTB 容量对预测正确率的影响

由预测正确率定义可知, BTB 容量对失效率的影响将导致预测正确率的变化. 理想的 BTB 通常需要和程序虚地址空间一样大, 或者使用非常复杂的映射函数. BTB 容量的选择还应考虑到实现代价如硬件成本、功耗等因素, 以及对转移预测后取指周期的影响. 参考 PowerPC 和 Pentium 中的 BTB 设计, 我们对几种 BTB 容量进行了评测, 并确定最终设计中 BTB 容量为 1024 项.

417 PHT 计数器饱和率的变化图

PHT 表项中存放了 2 位饱和计数器, 在达到饱和时如果同向更新则不必写回, 从而减少了硬件回写开销. 图 4 给出了各种配置对于计数器饱和率的影响. 饱和率定义为计数器饱和并同向增长的次数与所有预测次数的比值. 好的动态预测策略应尽可能的提高 PHT 计数器的饱和率. 饱和率越高, 代表动态预测器达到稳定状态的时间越长、预测正确率越高、硬件回写次数越少. 图 4 中的基本配置为 8 位 BHR, 1024 项、2 路组相联结构的 BTB, 1024 项 PHT. 图中自上而下选用的测试程序分别为 SPEC2000 中的 gzip、gcc、parser、vortex、bizp2 基准程序.

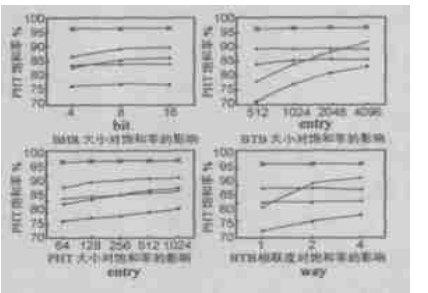

图 4 PHT 计数器饱和率变化图

418 进程切换对预测正确率的影响

操作系统进程切换时需要刷新 BTB, 同时设计人员还需要考虑对 BHR、PHT 是否进行刷新. 进程切换时可以考虑以下几种刷新方案:

A: BHR 不刷新, PHT 不刷新

B: BHR 不刷新, PHT 刷新

C: BHR 刷新, PHT 不刷新

D: BHR 刷新, PHT 刷新

表 4 转移指令平均执行周期比较

SPEC95 测试程序	M1%	M2%	M3%	M4%	M5%	M6%	CPB	BP
099.go	39192	11182	12100	30148	31132	21446	21210	11154
124.m88ksim	57165	3140	1187	36179	01118	01111	21219	11110
126.gcc	40198	4112	3182	43111	41121	31176	11198	11124
129.compress	66126	2178	2182	28107	01104	01103	21238	11111
130.li	44101	1174	2151	51138	01125	01111	11194	11109
132.ijpeg	58143	5139	5142	30146	01118	01112	21228	11122
134.perl	49144	1162	2128	42153	21130	11183	21208	11112
147.vortex	42168	1128	1134	49160	21173	21137	11194	11111
101.tomcatv	58172	6117	5142	26177	11168	11124	21232	11127
107.mgrid	91122	2103	0147	6122	01104	01102	21283	11105
145.fpppp	68180	7169	8166	14177	01106	01102	21255	11133
SPEC2000 测试程序	M1%	M2%	M3%	M4%	M5%	M6%	CPB	CPB-BP
164.gzip	61136	3127	3125	32112	01100	01100	21229	11113
175.vpr	64127	2183	2153	28173	11108	01156	21236	11113
176.gcc	40176	3194	3148	43175	41121	31186	11197	11123
181.mcf	56174	4143	4134	34129	01105	01115	21222	11118
197.parser	57164	4126	4163	33121	01116	01110	21225	11118
255.vortex	43162	1148	1122	48195	21163	21110	11195	11111
256.bzip2	79199	1136	0167	17198	01100	01100	21161	11104
300.twolf	52144	7144	7128	31158	01193	01133	21221	11131

刷新 BHR 寄存器是指把 BHR 寄存器清零, 刷新 PHT 是指将 PHT 所有表项内容设置为 10_2. 不刷新是指保留被换出进程的缓存不变.

实验显示, 进程切换时是否刷新 BHR 和 PHT 对预测正确率的影响很小, 正确率变化范围在 0101% 至 0103% 之间, 因此我们得到与文[8] 相同的结论: 即进程切换时对 BHR 和 PHT 不需要刷新. 本章上述实验数据都是基于操作系统不刷新

BHR 和 PHT 的前提而得到的.

4.1.9　总体性能比较

以上实验表明, 采用 8 位 BHR, 1024 项 PHT, 以及采用 2 路组相联结构的 1024 项 BTB 具有较高的预测正确率, 并且硬件实现代价较低. 采用上述配置的投合预测器扩展 UNICORE 体系结构, 针对 SPEC95 程序和 SPEC2000 程序可分别获得 1.109 和 1.125 的平均加速比.

导致 UNICORE 微处理器的性能提升的主要原因是转移指令的执行开销被降低. 表 4 给出了表 2 中各种执行情况所占的百分比, 并计算得到了转移指令的平均执行周期, 其中 CPB(cycle per branch) 和 CPB2BP 分别表示 UNICORE 流水线添加动态转移预测部件前后转移指令的平均执行周期, 从表中可以看出添加预测模块之前的 UNICORE 处理器执行一条转移指令平均需要 2.134 个周期, 添加预测模块之后平均只需要 1.12 个周期, 转移指令的执行速度提高了一倍.

5　结论

在单发射流水线结构中实现动态转移预测功能, 对提升系统性能非常重要. 实验显示, 改造后的 UNICORE 体系结构可得到 1.11 的平均加速比, 转移指令的执行开销降低了 50%.

使用系统级别的模拟器, 可以反映操作系统行为对预测器的影响, 所得到的实验结论更加可信, 对于类似的处理器设计非常关键. 我们借助 USim 模拟器, 通过对比 BHR、PHT、以及 BTB 大小、BTB 相联度对预测正确率的影响并考虑了硬件实现代价, 最后确定了设计方案.

参考文献:

[1] Smith J E. A study of branch prediction strategies[A]. In Proceedings of the 8th ISCA[C]. IEEE Computer Society Press, May 1981. 135-48.

[2] John L Hennessy, David A. Patterson: Computer Architecture: A Quantitative Approach[M]. Third Edition, Morgan Kaufmann Publishers, 2002.

[3] T Y Yeh, Y N Patt. Alternative implementations of two-level adaptive branch prediction[A]. Proceedings of the 19th ISCA[C]. ACM Press, May. 1992. 124- 134.

[4] E Sprangle, et al. The agree predictor: A mechanism for reducing negative branch history interference[A]. Proceedings of the 24th ISCA[C]. ACM Press, May 1997. 284- 291.

[5] Lee C C, Chen I C, Mudge T. The Bi-mode branch predictor[A]. In Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture[C]. IEEE Computer Society, 1997. 4- 13.

[6] Michaud P, et al. Trading conflict and capacity aliasing in conditional branch predictors[A]. Proceedings of the 24th ISCA[C]. ACM Press, June 1997. 292- 303.

[7] Chang P Y, Evers M, Patt Y. Improving branch prediction accuracy by reducing pattern history table interference[J]. International Journal of Parallel Programming, 1997, 25(5): 339- 362.

[8] Eden A N, Mudge T. The YAGS branch prediction scheme[A]. International Symposium on Microarchitecture[C]. IEEE Computer Society Press, Dec. 1998. 69- 77.

作者简介:

朱德新　男, 1976 年出生, 1998 年本科毕业于北京大学, 现在该校攻读博士学位, 主要研究领域为计算机体系结构、优化编译技术.

程　旭　男, 1967 年出生, 教授, 博士生导师, 主要研究领域为计算机体系结构、处理器芯片设计和系统软件.