香山处理器 循环预测器和循环缓冲部件 设计与实现

邹江瑞[1]　勾凌睿[2]　金越[2]　张林隽[2]

[1]深圳大学　[2]中科院计算所

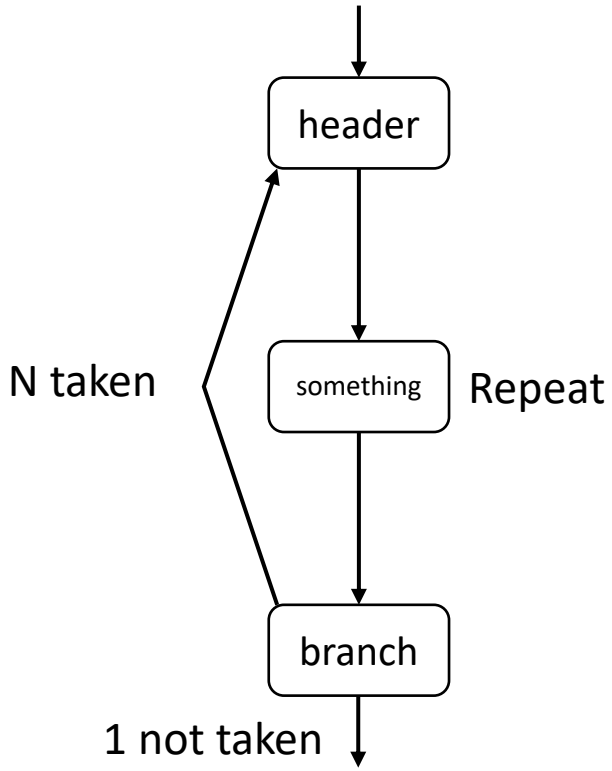
2021年6月25日

背景

- 处理器架构中针对循环的优化
  - 循环由循环体和循环体最后的一条跳转指令组成
  - 这条跳转指令一定是N次跳转+1次不跳转，这种循环跳转指令依靠近期分支历史的分支预测可能无法预测到
  - 循环会迭代执行很多次，而且每次执行的都是相同的指令，即使是相同的指令，仍然会从ICache里重复地取相同的指令

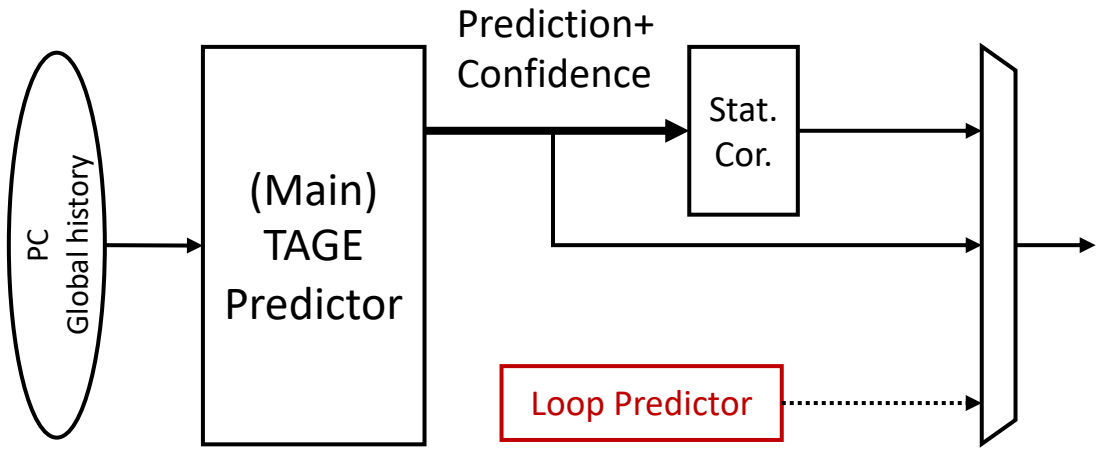
- （**循环预测器** Loop Predictor）
- 在循环体较长、循环次数较多的循环中，专门用于预测循环跳转指令

- （**循环缓冲** Loop Buffer）
- 将循环中的指令缓存起来，从循环缓冲向后传递指令，关闭ICache节省功耗



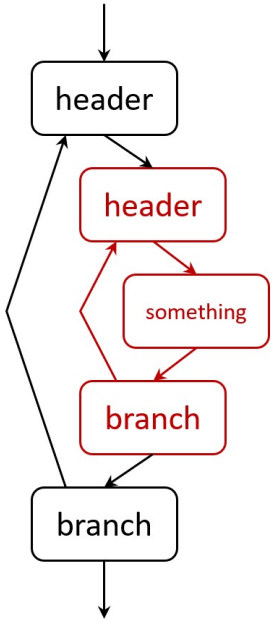
循环预测器-原理

- 属于TAGE-SC-L预测器的一部分
- 针对多重循环中的内层循环:
- 循环的基本形式是
  N次跳转+1次不跳转
- 在循环执行的过程中，通过训练记录下循环的总迭代次数(trip count)，以及循环当前的迭代次数(spec count)，可以预测出什么时候循环退出
  (spec count == trip count)



循环预测器在TAGE-SC-L预测器中的位置

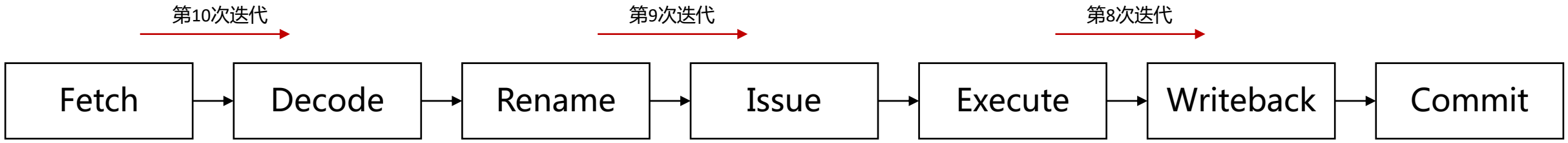
```
for (i=0; i<N; i++)
{
   for (j=0; j<M; j++)
   {
      do something
   }
}
```



简单的双重循环

循环预测器-硬件实现

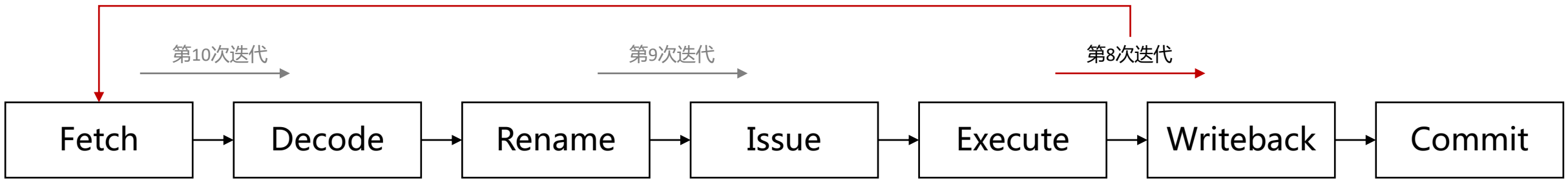
- **如何记录迭代次数和总迭代次数？**
- 当一条指令发给后端时，推测更新对应的迭代次数，并将当前的迭代次数n随着指令一起送入流水线
- 当某条循环跳转指令误预测时，需要恢复到之前的迭代次数，即误预测指令携带的迭代次数n
- 误预测时的迭代次数n可能是总迭代次数，误预测一定次数后则认为这个循环被训练完成



在指令经过取指时推测更新迭代次数

循环预测器-硬件实现

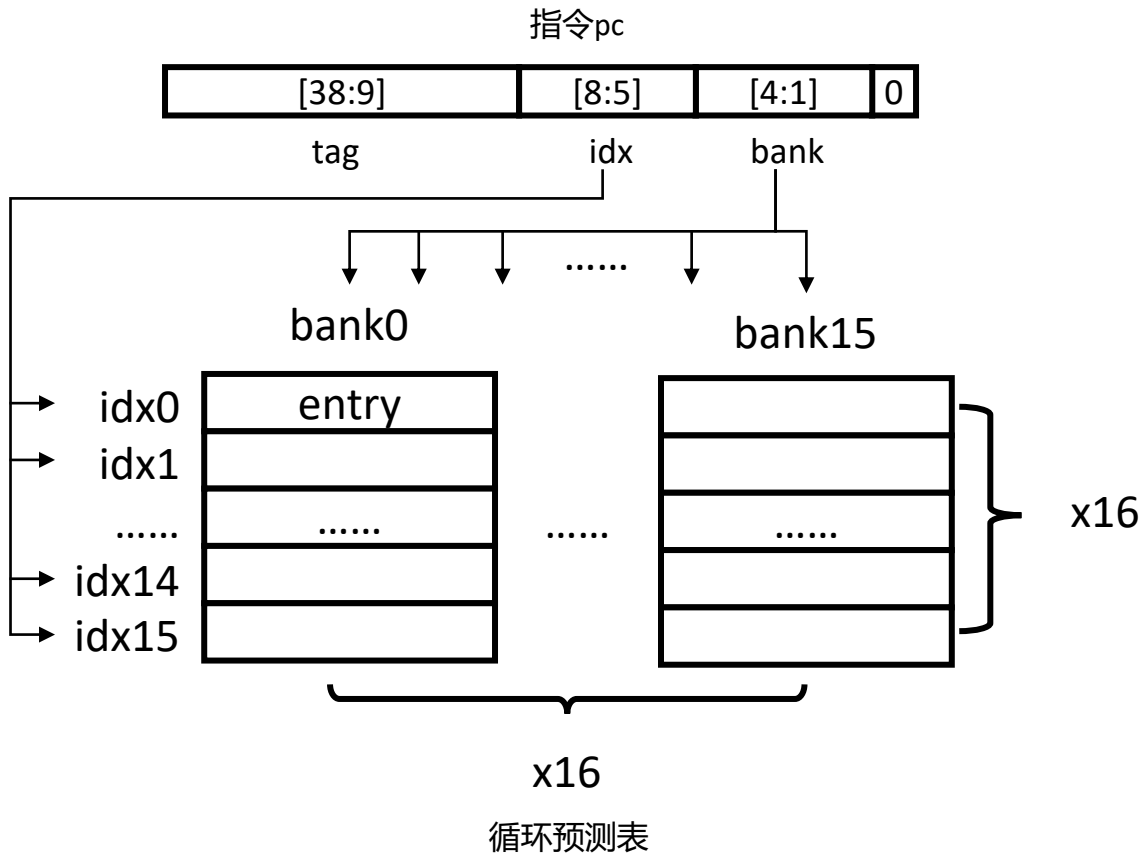
- **如何记录迭代次数和总迭代次数？**
- 当一条指令发给后端时，推测更新对应的迭代次数，并将当前的迭代次数n随着指令一起送入流水线
- 当某条循环跳转指令误预测时，需要恢复到之前的迭代次数，即误预测指令携带的迭代次数n
- 误预测时的迭代次数n可能是总迭代次数，误预测一定次数后则认为这个循环被训练完成



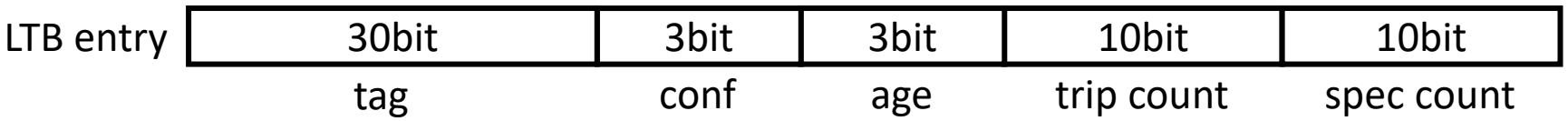
在指令误预测时发回信号，恢复出错的表项

循环预测器-硬件实现

- **如何存储循环的信息？**
- 循环预测表 Loop Termination Buffer (LTB)
  - 用来存储循环相关信息
  - 将每个循环绑定到该循环的最后一条跳转指令的PC上
  - 由于取指宽度最大为16，因此分16个bank
    - LTB要求多读多写，分成16个bank，每个bank只要一读一写，降低面积，布局更灵活
  - 每个bank里存16个表项，通过指令pc来索引



循环预测表

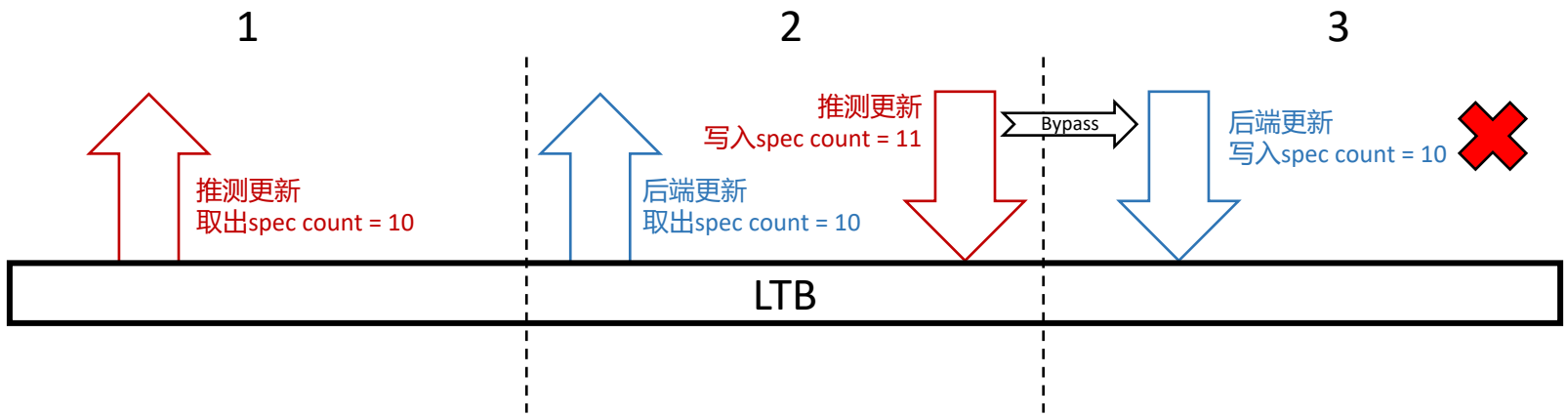


循环预测表表项

循环预测器-硬件实现

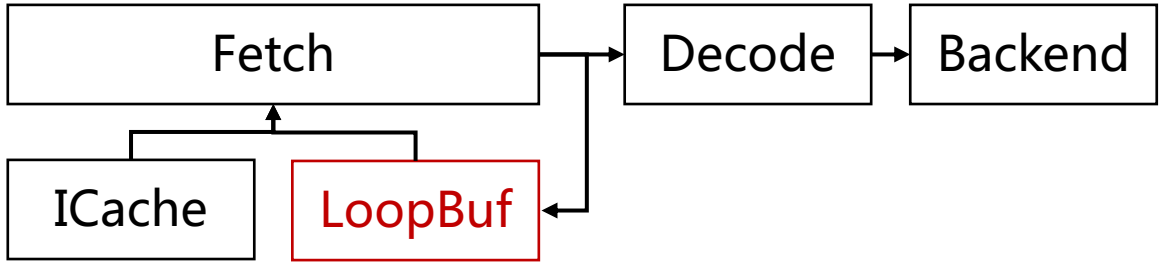
- **循环预测表表项的更新**
- 提前读出与延迟写入
  - 在一个周期内将表项从循环预测表中读取出来，更新相关属性之后再写回，会极大的影响频率
  - 推测更新逻辑可以通过提前将表项取出，指令真正经过循环预测器时再写入
  - 后端更新逻辑可以在误预测时先读出更新，下一拍再延迟写入

  - 推测更新和延迟更新需要添加旁路
    - 比如说上一拍取出的推测更新表项，这一拍后端发回更新，这一拍取出的数据就是旧的，下一拍又会把旧的数据写回



循环缓冲-原理

- 循环缓冲和ICache同级，但是会检测前端传给后端流水线的指令信息
- 针对循环体中的指令
- 通过检测指令流中多次重复出现的指令，训练识别循环，存储循环体内的指令，减少从ICache取指
- 降低功耗
- 缓冲队列大小：64*2Byte，能够存储32-64条指令长度的循环



循环缓冲在前端的位置

```
8000007c:   0014041b            addiw   s0,s0,1
80000080:   00040793            mv  a5,s0
80000084:   07340063            beq s0,s3,800000e4 <main+0xa4>
80000088:   0327c73b            divw    a4,a5,s2
8000008c:   0347c53b            divw    a0,a5,s4
80000090:   032766bb            remw    a3,a4,s2
80000094:   0327e7bb            remw    a5,a5,s2
80000098:   02d6873b            mulw    a4,a3,a3
8000009c:   02a505bb            mulw    a1,a0,a0
800000a0:   02f7863b            mulw    a2,a5,a5
800000a4:   02d7073b            mulw    a4,a4,a3
800000a8:   02a585bb            mulw    a1,a1,a0
800000ac:   02f606bb            mulw    a3,a2,a5
800000b0:   00b707bb            addw    a5,a4,a1
800000b4:   00d787bb            addw    a5,a5,a3
800000b8:   fc8792e3            bne a5,s0,8000007c <main+0x3c>
```

某汇编程序中的循环

循环缓冲-硬件实现

- **如何检测循环？**
- 循环的特征
  - 以任意一条指令开始
  - 以一条跳转指令结束
  - 跳转指令一定是从地址高位向地址低位跳转
  - 循环跳转指令到目标地址之间就是循环体

```
8000007c:   0014041b            addiw   s0,s0,1
80000080:   00040793            mv  a5,s0
80000084:   07340063            beq s0,s3,800000e4 <main+0xa4>
80000088:   0327c73b            divw    a4,a5,s2
8000008c:   0347c53b            divw    a0,a5,s4
80000090:   032766bb            remw    a3,a4,s2
80000094:   0327e7bb            remw    a5,a5,s2
80000098:   02d6873b            mulw    a4,a3,a3
8000009c:   02a505bb            mulw    a1,a0,a0
800000a0:   02f7863b            mulw    a2,a5,a5
800000a4:   02d7073b            mulw    a4,a4,a3
800000a8:   02a585bb            mulw    a1,a1,a0
800000ac:   02f606bb            mulw    a3,a2,a5
800000b0:   00b707bb            addw    a5,a4,a1
800000b4:   00d787bb            addw    a5,a5,a3
800000b8:   fc8792e3            bne a5,s0,8000007c <main+0x3c>
```

SBB

- 检测预测跳转，且向后跳转的指令
- Short Backward Branch (SBB)

循环缓冲-硬件实现

- 第一次迭代
  - 检测到SBB，记录pc和offset
- 第二次迭代
  - 再次看到相同的SBB，认为这是一个循环
- 第三次迭代
  - 把循环中每条指令都缓存进循环缓冲
  - 再次看到SBB则缓存完成
- 第四次迭代
  - 开始从循环缓冲供指

```
8000007c:   0014041b            addiw   s0,s0,1
80000080:   00040793            mv  a5,s0
80000084:   07340063            beq s0,s3,800000e4 <main+0xa4>
80000088:   0327c73b            divw    a4,a5,s2
8000008c:   0347c53b            divw    a0,a5,s4
80000090:   032766bb            remw    a3,a4,s2
80000094:   0327e7bb            remw    a5,a5,s2
80000098:   02d6873b            mulw    a4,a3,a3
8000009c:   02a505bb            mulw    a1,a0,a0
800000a0:   02f7863b            mulw    a2,a5,a5
800000a4:   02d7073b            mulw    a4,a4,a3
800000a8:   02a585bb            mulw    a1,a1,a0
800000ac:   02f606bb            mulw    a3,a2,a5
800000b0:   00b707bb            addw    a5,a4,a1
800000b4:   00d787bb            addw    a5,a5,a3
800000b8:   fc8792e3            bne a5,s0,8000007c <main+0x3c>
```

SBB

循环缓冲-硬件实现

- 并不是所有循环都是理想的
- 当在训练过程中，和循环缓冲供指的过程中出现了SBB以外的跳转指令时，需要退出训练或退出供指
- 当循环体内的if-else语句跳转时，也需要退出训练或退出供指，因为不能保证循环体所有指令都在缓存中

```
8000007c:   0014041b            addiw   s0,s0,1
80000080:   00040793            mv  a5,s0
80000084:   07340063            beq s0,s3,800000e4 <main+0xa4>
80000088:   0327c73b            divw    a4,a5,s2
8000008c:   0347c53b            divw    a0,a5,s4
80000090:   032766bb            remw    a3,a4,s2
80000094:   0327e7bb            remw    a5,a5,s2
80000098:   02d6873b            mulw    a4,a3,a3
8000009c:   02a505bb            mulw    a1,a0,a0
800000a0:   02f7863b            mulw    a2,a5,a5
800000a4:   02d7073b            mulw    a4,a4,a3
800000a8:   02a585bb            mulw    a1,a1,a0
800000ac:   02f606bb            mulw    a3,a2,a5
800000b0:   00b707bb            addw    a5,a4,a1
800000b4:   00d787bb            addw    a5,a5,a3
800000b8:   fc8792e3            bne a5,s0,8000007c <main+0x3c>  SBB
```

从循环内跳转到循环外

性能

- **循环预测器**
- 测试程序：Coremark 100轮
- IPC：提升1.5%
- 针对循环退出的分支指令能够达到75.3%正确率
- 降低了6.6%的总分支指令误预测数

- **循环缓冲**
- Coremark 100轮
- 增加了5.27%的总请求次数
- 降低32.7%的ICache请求次数
- 会导致IPC下降2.23%，因此暂时没有合进香山主线

敬请批评指正！

附加页

- 循环预测器误预测更新逻辑