深圳大学实验报告

课程名称： 计算机系统(3)

实验项目名称： 处理器结构实验二

学院： 计算机与软件学院

专业： 计算机与软件学院所有专业

指导教师： 刘　刚

报告人：　**学号：**　**班级：**

实验时间： 2025 年 11 月 26 日

实验报告提交时间： 2025 年 12 月 1 日

教务处制

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

一、实验目的

——控制冒险与分支预测

了解控制冒险分支预测的概念

了解多种分支预测的方法，动态分支预测更要深入了解

理解什么是 BTB（Branch Target Buffer），并且学会用 BTB 来优化所给程序

利用 BTB 的特点，设计并了解在哪种状态下 BTB 无效

了解循环展开，并于 BTB 功能进行对比

对 WinMIPS64 的各个窗口和操作更加熟悉

二、实验内容

按照下面的实验步骤及说明，完成相关操作记录实验过程的截图：

首先，给出一段矩阵乘法的代码，通过开启 BTB 功能对其进行优化，并且观察流水线的细节，解释 BTB 在其中所起的作用；

其次，自行设计一段使得即使开启了 BTB 也无效的代码。

第三，使用循环展开的方法，观察流水因分支停顿的次数减少的现象，并对比采用 BTB 结构时流水因分支而停顿的次数。

（选做：在 x86 系统上编写 C 语言的矩阵乘法代码，用 perf 观察分支预测失败次数，分析其次数是否与你所学知识吻合。再编写前面第二部使用的令分支预测失败的代码，验证 x86 是否能正确预测，并尝试做解释）

三、实验环境

硬件：桌面 PC

软件：Windows

四、实验步骤及说明

背景知识

在遇到跳转语句的时候，我们往往需要等到 MEM 阶段才能确定这条指令是否跳转（通过硬件的优化，可以极大的缩短分支的延迟，将分支执行提前到 ID 阶段，这样就能够将分支预测错误代价减小到只有一条指令），这种为了确保预取正确指令而导致的延迟叫控制冒险（分支冒险）。

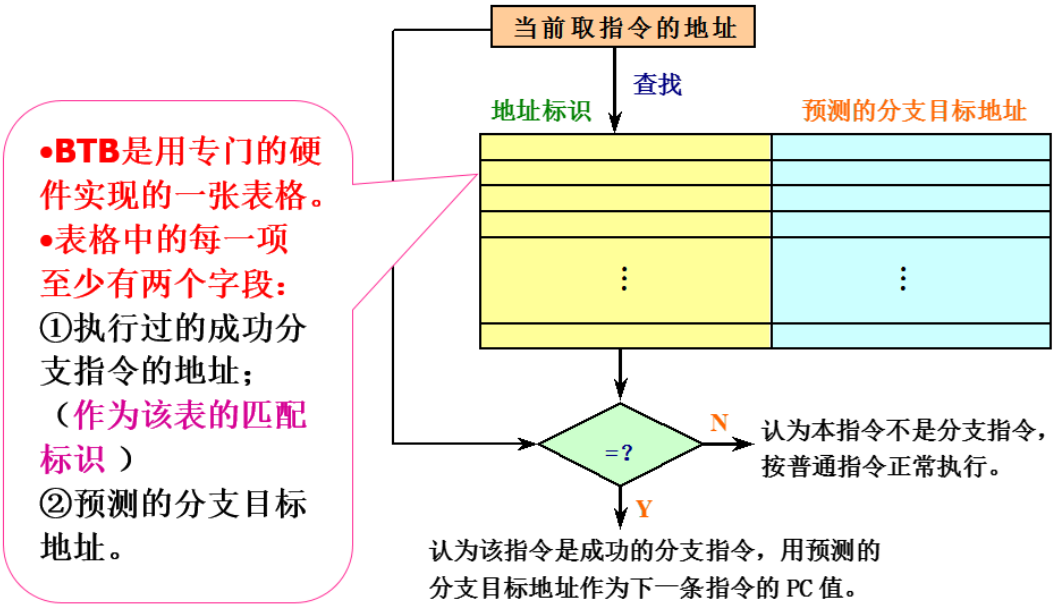
为了降低控制冒险所带来的性能损失，一般采用分支预测技术。分支预测技术包含编译时进行的静态分支预测，和执行时进行的动态分支预测。这里，我们着重介绍动态分支预测中的 BTB（Branch Target Buffer）技术。

BTB 即为分支目标缓冲器，它将分支指令（对应的指令地址）放到一个缓冲区中保存起来，当下次再遇到相同的指令（跳转判定）时，它将执行和上次一样的跳转（分支或不分支）预测。

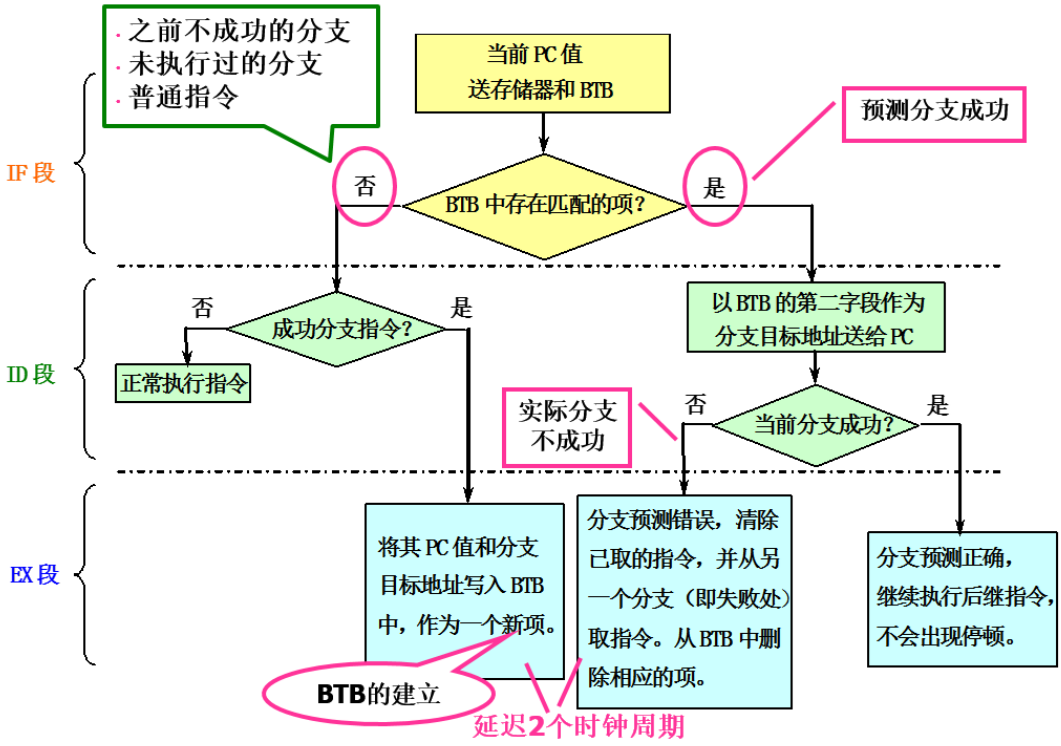
一种可行的 BTB 结构示意图如下：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。



在采用了 BTB 之后，在流水线各个阶段所进行的相关操作如下：



注意，为了填写 BTB，需要额外一个周期。

一、矩阵乘法及优化

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

在这一阶段，我们首先给出矩阵乘法的例子，接着将流水线设置为不带 BTB 功能（configure->enable branch target buffer）直接运行，观察结果进行记录；然后，再开启 BTB 功能再次运行，观察实验结果。将两次的实验结果进行对比，观察 BTB 是否起作用，如果有效果则进一步观察流水线执行细节并且解释 BTB 起作用原因。

矩阵乘法的代码如下：

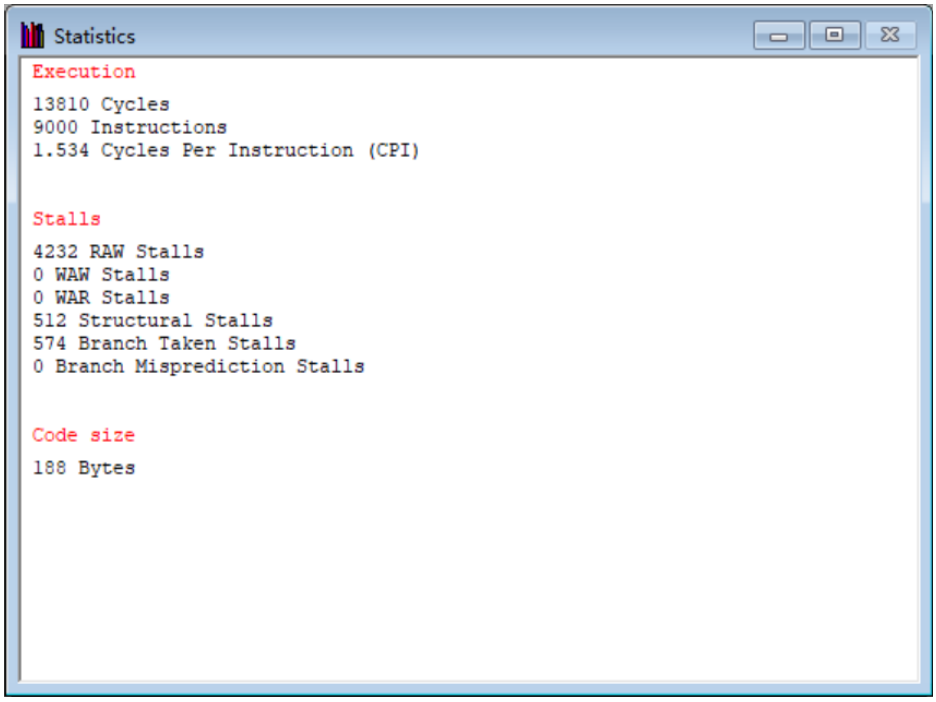
```
        .data
str:  .asciiz "the data of matrix 3:\n"
mx1:   .space 512
mx2:   .space 512
mx3:   .space 512

            .text
initial:    daddi r22,r0,mx1  #这个initial模块是给三个矩阵赋初值
            daddi r23,r0,mx2
            daddi r21,r0,mx3
input:      daddi r9,r0,64
            daddi r8,r0,0
loop1:      dsll r11,r8,3
            dadd r10,r11,r22
            dadd r11,r11,r23
            daddi r12,r0,2
            daddi r13,r0,3
            sd r12,0(r10)
            sd r13,0(r11)

            daddi r8,r8,1
            slt r10,r8,r9
            bne r10,r0,loop1

mul:        daddi r16,r0,8
            daddi r17,r0,0
loop2:      daddi r18,r0,0    #这个循环是执行for(int i = 0, i < 8; i++)的内容
loop3:      daddi r19,r0,0    #这个循环是执行for(int j = 0, j < 8; j++)的内容
            daddi r20,r0,0    #r20存储在计算result[i][j]过程中每个乘法结果的叠加值
loop4:      dsll r8,r17,6     #这个循环的执行计算每个result[i][j]
            dsll r9,r19,3
            dadd r8,r8,r9
            dadd r8,r8,r22
            ld r10,0(r8)      #取mx1[i][k]的值
            dsll r8,r19,6
            dsll r9,r18,3
            dadd r8,r8,r9
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

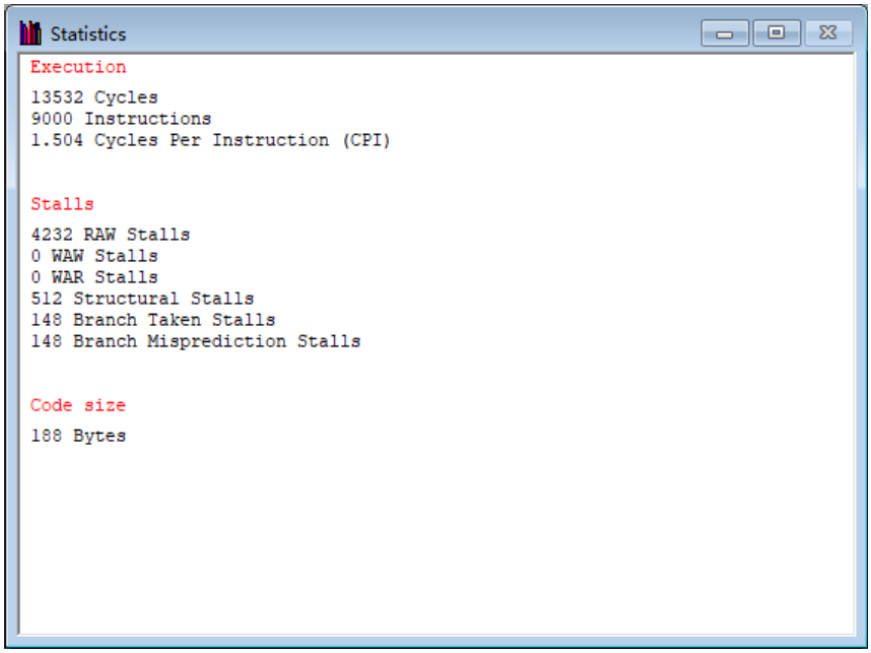
```
dadd r8,r8,r23
ld r11,0(r8)       #取mx2[k][j]的值
dmul r13,r10,r11   #mx1[i][k]与mx2[k][j]相乘
dadd r20,r20,r13   #中间结果累加

daddi r19,r19,1
slt r8,r19,r16
bne r8,r0,loop4

dsll r8,r17,6
dsll r9,r18,3
dadd r8,r8,r9
dadd r8,r8,r21     #计算result[i][j]的位置
sd r20,0(r8)       #将结果存入result[i][j]中

daddi r18,r18,1
slt r8,r18,r16
bne r8,r0,loop3

daddi r17,r17,1
slt r8,r17,r16
bne r8,r0,loop2

halt
```

不设置 BTB 功能，运行该程序，观察 Statistics 窗口的结果截屏并记录下来。



注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

接着，设置 BTB 功能（在菜单栏处选择 Configure 项，然后在下拉菜单中为 Enable Branch Target Buffer 选项划上钩）。并在此运行程序，观察 Statistics 窗口的结果并截屏记录下来。

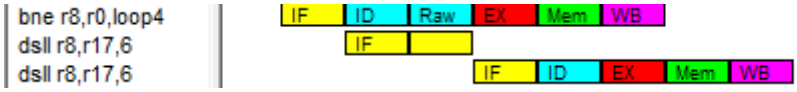


在这里，我们仅仅观察比较 Stalls 中的最后两项------Branch Taken Stalls 和 Branch Misprediction Stalls。

接下来，对比其结果。我们就结合流水线执行细节分析造成这种情况发生的原因。

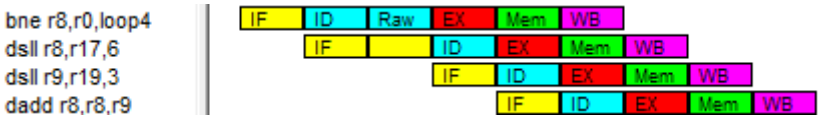
对比于开启了 BTB 功能后的情况，没有开启 BTB 功能时，Branch Taken Stalls 的数目远大于开启 BTB 功能后的 Branch Taken Stalls 和 Branch Misprediction Stalls 的值。

不设置 BTB 运行时，看 Cycles 窗口的执行过程，随意找一处 Branch Taken Stalls，其中的执行过程如下：



由于分支语句基本都跳转了的缘故，造成其每次都要等一下。

开启 BTB 后，执行到此处：



对比发现，这次分支语句由于 BTB 的存在，直接跳转到 loop4 的位置执行，而不是像 zhiq 那样卡两个周期。

同理，BTB 对每一处分支语句都有相同的优化作用，这在前边的对比结果右明显体现。

（30 分，结果的获取 10 分，细节分析 20 分）

二、设计使 BTB 无效的代码

在这个部分，我们要设计一段代码，这段代码包含了一个循环。根据 BTB 的特性，我们设计的这个代码将使得 BTB 的开启起不到相应的优化作用，反而会是的性能大大降低。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

提示：一定要利用 BTB 的特性，即它的跳转判定是根据之前跳转成功与否来决定的。

给出所用代码以及设计思路，给出运行结果的截屏证明代码实现了目标。
（30 分，代码及思路 20，获取结果并证明目标实现 10 分）

代码：

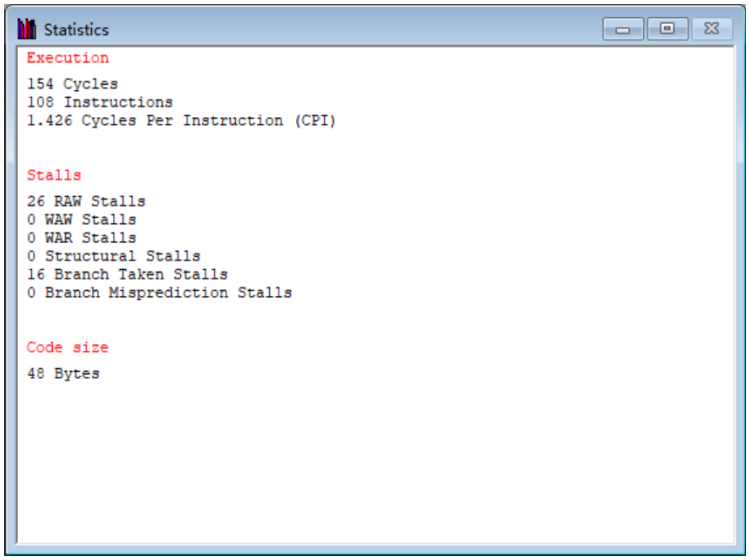
```
.data
array:    .word 0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1

          .text
          daddi r23,r0,array
          daddi r9,r0,16
          daddi r8,r0,0
loop:     dsll r11,r8,3
          dadd r10,r11,r23
          ld r12,0(r10)          #上面三行取数组的值
          daddi r8,r8,1

          bne r12,r0,loop        #如果数组值不为 0 则跳转到 loop

          slt r10,r8,r9          #次循环判定是为了保证每次都循环 16 次
          bne r10,r0,loop

          daddi r17,r0,1      #此行代码是为了方便观察 Cycles 窗口中跳转的情况
     halt
```

思路：

代码中 0，1 交替的 16 个 word 的数组决定跳转，这样连续两次的跳转都不一样。这样就很好地利用 BTB 的特性，使其每次分支预测都错误，导致性能比不启用 BTB 还降低。

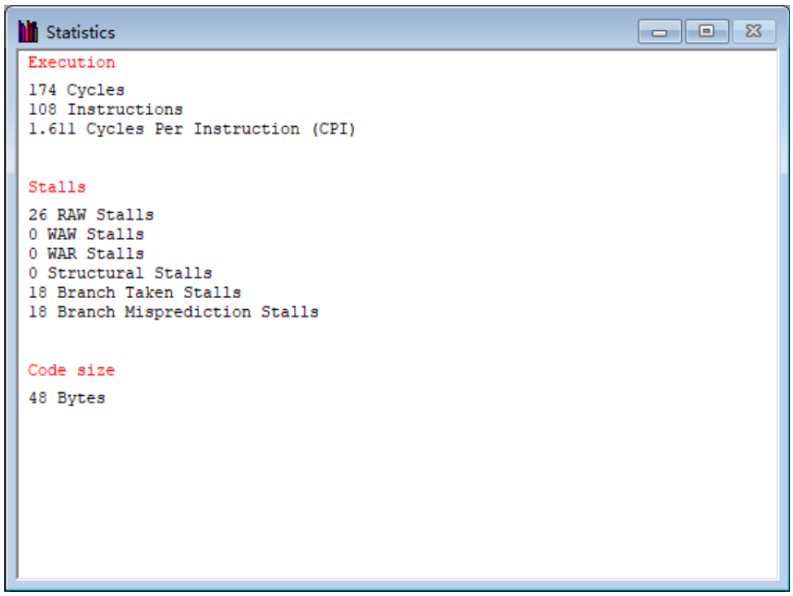
结果：

不启用 BTB：



注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

启用 BTB：



从上边两图数据对比可以明显看出，在开启了 BTB 功能之后，效率反而降低了，当然这是人为故意导致的。

三、循环展开与 BTB 的效果比对

首先，我们需要对循环展开这个概念有一定的了解。

什么是循环展开呢？所谓循环展开就是通过在每次迭代中执行更多的数据操作来减小循环开销的影响。其基本思想是设法把操作对象线性化，并且在一次迭代中访问线性数据中的一个小组而非单独的某个。这样得到的程序将执行更少的迭代次数，于是循环开销就被有效地降低了。

接下来，我们就按照这种思想对上述的矩阵乘法程序进行循环展开。要求将上述的代码通过循环展开将最里面的一个执行迭代 8 次的循环整个展开了，也就是说，我们将矩阵相乘的三个循环通过代码的增加，减少到了两个循环。

比较，通过对比循环展开（未启用BTB）、使用BTB（未进行循环展开）以及未使用BTB且未作循环展开的运行结果。比较他们的Branch Tanken Stalls和Branch Misprediction Stalls的数量，并尝试给出评判。

（30 分，循环展开代码及思路 20 分，评判 10 分）

将原来的代码做成线性的：

```
.data
str:  .asciiz "the data of matrix 3:\n"
mx1:   .space 512
mx2:   .space 512
mx3:   .space 512
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
            .text
initial:    daddi r22,r0,mx1
            daddi r23,r0,mx2
            daddi r21,r0,mx3
input:      daddi r9,r0,64
            daddi r8,r0,0
loop1:      dsll r11,r8,3
            dadd r10,r11,r22
            dadd r11,r11,r23
            daddi r12,r0,2
            daddi r13,r0,3
            sd r12,0(r10)
            sd r13,0(r11)

            daddi r8,r8,1
            slt r10,r8,r9
            bne r10,r0,loop1

mul:        daddi r16,r0,8
            daddi r17,r0,0    #r17代表i
loop2:      daddi r18,r0,0    #r18代表j
loop3:      daddi r19,r0,0    #r19代表k
            daddi r20,r0,0    #r20存储叠加乘积的中间值

            dsll r8,r17,6     #r8存储mx1[i][0]的位置，在整个loop3循环中保持不变
            dsll r9,r19,3
            dadd r9,r8,r9     #r9是保存mx1[i][k]相对于mx1[i][0]的偏移量，r9在loop3循环
中会改变
            dadd r9,r9,r22
            ld r9,0(r9)       #读取mx1[i][k]的值

            dsll r10,r19,6    #r10存储mx2[k][0]值，在整个loop3循环中会发生改变
            dsll r11,r18,3    #r11存储mx2[k][j]相对于mx2[k][0]的值，在整个loop3种保持不
变
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)     #读取mx2[k][j]的值
            dmul r12,r9,r10   #mx1[i][k]与mx2[k][j]相乘
            dadd r20,r20,r12  #结果与r20累加
            daddi r19,r19,1   #r9的值加一

            dsll r9,r19,3
            dadd r9,r8,r9
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
            dadd r9,r9,r22
            ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
            dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
            dmul r12,r9,r10
            dadd r20,r20,r12
            daddi r19,r19,1    #r19+1

            dsll r9,r19,3
            dadd r9,r8,r9
            dadd r9,r9,r22
            ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
            dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
            dmul r12,r9,r10
            dadd r20,r20,r12
            daddi r19,r19,1    #r19+1


            dsll r9,r19,3
            dadd r9,r8,r9
            dadd r9,r9,r22
            ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
            dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
            dmul r12,r9,r10
            dadd r20,r20,r12
            daddi r19,r19,1    #r19+1

            dsll r9,r19,3
            dadd r9,r8,r9
```

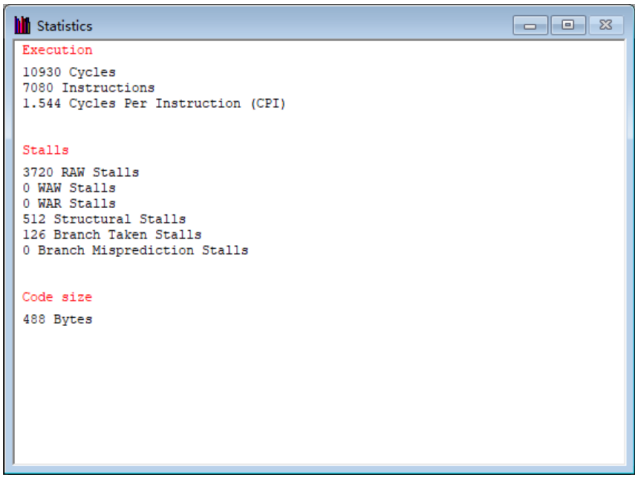
注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
            dadd r9,r9,r22
            ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
            dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
            dmul r12,r9,r10
            dadd r20,r20,r12
            daddi r19,r19,1    #r19+1


            dsll r9,r19,3
            dadd r9,r8,r9
            dadd r9,r9,r22
            ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
            dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
            dmul r12,r9,r10
            dadd r20,r20,r12
            daddi r19,r19,1    #r19+1


            dsll r9,r19,3
            dadd r9,r8,r9
            dadd r9,r9,r22
            ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
            dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
            dadd r10,r10,r11
            dadd r10,r10,r23
            ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
            dmul r12,r9,r10
            dadd r20,r20,r12
            daddi r19,r19,1    #r19+1
```

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```
dsll r9,r19,3
dadd r9,r8,r9
dadd r9,r9,r22
ld r9,0(r8)        #重复上面取mx1[i][k]值的过程，变化的是r19的值加了一，意味着k+1
dsll r10,r19,6     #r10的值由于r19的改变而发生了改变，即k变成k+1
dadd r10,r10,r11
dadd r10,r10,r23
ld r10,0(r10)      #复上面取mx1[k][j]值的过程，变化的是r19的值加了一，意味着k+1
dmul r12,r9,r10
dadd r20,r20,r12
daddi r19,r19,1    #r19+1



dsll r8,r17,6
dsll r9,r18,3
dadd r8,r8,r9
dadd r8,r8,r21
sd r20,0(r8)

daddi r18,r18,1
slt r8,r18,r16
bne r8,r0,loop3

daddi r17,r17,1
slt r8,r17,r16
bne r8,r0,loop2

halt
```

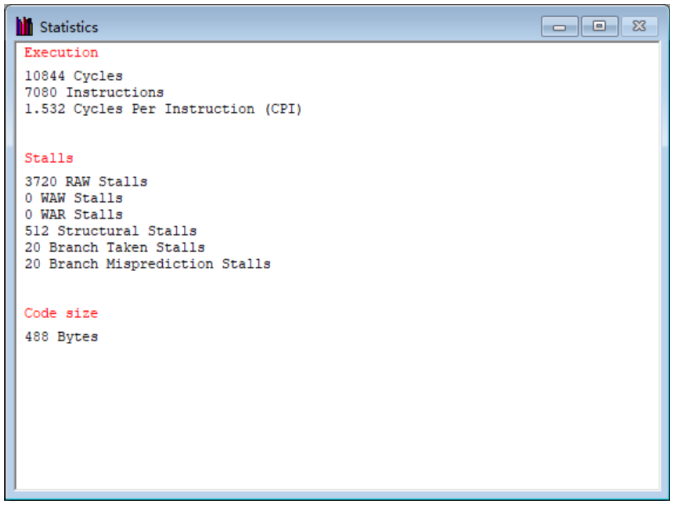
运行结果如下图所示（未开启 BTB）：



注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

开启 BTB：



根据以上的所有结果，可以总给出：

	Branch Tanken Stalls	Branch Misprediction Stalls
循环展开（未启用 BTB）	126	0
使用 BTB（未进行循环展开）	148	148
未使用 BTB 且未作循环展开	574	0

可以看出，通过循环展开所得到的结果比原始代码，无论有无BTB，Branch Tanken Stalls和Branch Misprediction Stalls的数量都更少。

总结来说，就是循环展开通过更冗余的代码为代价，拥有三者中最好的优化效果；启用 BTB 则在大多数情况下能起到优化作用，但在特殊情况下也有可能频繁预测失败“适得其反”。

四、结束语

写下对于这次试验的所得与感想。

（报告撰写质量 10 分）

BTB 优化效果： 启用 BTB 功能后，程序的流水线停顿次数显著减少，尤其是 Branch Taken Stalls 和 Branch Misprediction Stalls 的数值明显降低，证明 BTB 在分支预测中的有效

BTB 失效代码： 在故意设计的代码中，利用 BTB 的特性，每次分支预测都是错误的，导致性能反而下降，展示了 BTB 的局限性。

循环展开的效果： 循环展开减少了分支判断的次数，尤其是在配合 BTB 使用时，进一步优化了程序的执行。

五、实验结果

细节见第三部分，以下为结论的汇总。

1、对比于开启了 BTB 功能后的情况，没有开启 BTB 功能时，Branch Taken Stalls 的数目远大于开启 BTB 功能后的 Branch Taken Stalls 和 Branch Misprediction Stalls 的值。

2、BTB 功能不绝对能优化。在特殊设计的代码中，在开启了 BTB 功能之后，效率反而降低了，当然这是人为故意导致的。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

3、可以看出，通过循环展开所得到的结果比原始代码，无论有无BTB，Branch Tanken Stalls和Branch Misprediction Stalls的数量都更少。

总结来说，就是循环展开通过更冗余的代码为代价，拥有三者中最好的优化效果；启用 BTB 则在大多数情况下能起到优化作用，但在特殊情况下也有可能频繁预测失败“适得其反”。

五、实验总结与体会

通过这次实验，我加深了对动态分支预测（BTB）的理解。在未启用 BTB 时，由于分支预测失败导致了明显的性能下降，这表明分支预测对流水线性能的重要性。启用 BTB 后，程序的分支预测准确性有所提高，分支停顿时间减少，性能得到明显优化。

然而，实验中也发现，设计一些特定的代码模式（如交替跳转的循环）可以使 BTB 失效，从而导致性能下降。通过设计特定的代码，使得 BTB 的预测效果大打折扣。代码中交替出现的 0 和 1 数组使得每次分支预测都是错误的，导致启用 BTB 后的性能反而下降。这展示了 BTB 的局限性，特别是在跳转模式高度不规律的情况下，BTB 的预测反而无法起到优化作用。

此外，通过对矩阵乘法代码进行循环展开，我观察到减少分支指令是优化程序性能的有效手段。展开后的代码减少了分支停顿，使得计算更加高效。

总的来说，这次实验帮助我更好地理解了分支预测的机制，并通过实际操作了解了不同优化策略对程序性能的影响。在未来的开发中，我可以根据程序的控制流特性来选择合适的优化策略，避免过度依赖 BTB，提高程序效率。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

指导教师批阅意见： **成绩评定：** 指导教师签字： 2025 年 12 月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
　　2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。