Implementation of Branch Decision

Computer Architecture: Lab #4

Due on May 31, 2020 at 23:59pm

魏剑宇
PB17111586

May 31, 2020

Contents

1 实验环境和工具

- **Host OS**: macOS Mojave 10.14.6
- **Guest OS**: Ubuntu Bionic Beaver
- **VSCode**: 1.44.2
- **Vivado**: HL WebPack Edition
- **Git**: 2.20.1 (Apple Git-117)

本次实验 Simulation 部分在 Ubuntu 虚拟机中完成 (Vivado 没有提供 macOS 版本)，代码编写在 mac 上使用 VSCode 完成.

我的做法是在虚拟机 vivado 创建项目时，将源码的文件夹添加进去的时候，不拷贝，而是映射到项目中。然后在 mac 上通过共享文件夹将源码映射到 ubuntu 上，这样，源码编辑和 Git 版本控制都可以在 mac 上完成，只需在仿真的时候切换到 Ubuntu。整个一套 workflow 还是非常流畅舒适的。

2 Branch History Table

下表中 target 表示既不在 BTB 中，也不是 PC_IF+4，而是指令中决定的跳转地址。

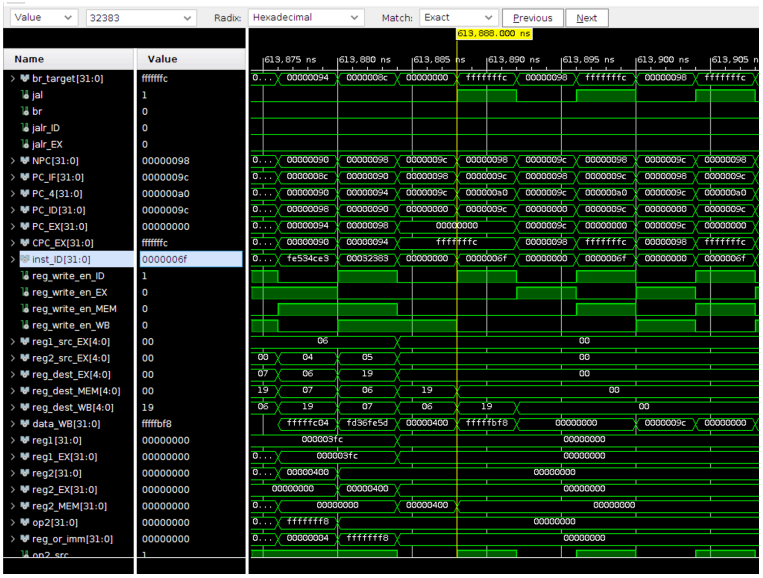
BTB	BHT	REAL	NPC_PRED	flush	NPC_REAL	BTB update
Y	Y	Y	BUF	N	BUF	N
Y	Y	N	BUF	Y	PC_IF+4	N
Y	N	Y	PC_IF+4	Y	BUF	N
Y	N	N	PC_IF+4	N	PC_IF+4	N
N	Y	Y	PC_IF+4	Y	target	Y
N	Y	N	PC_IF+4	N	PC_IF+4	N
N	N	Y	PC_IF+4	Y	target	Y
N	N	N	PC_IF+4	N	PC_IF+4	N

Table 1: BHT 策略矩阵

3 性能测试 & 分析

下面在进行性能测试的时候，我都是截止到最后死循环的时候（而不是死循环前的那个遍历数组的循环）.

例如，如下是矩阵乘法时结束的时间点:



另外，对于 bht.S 和 btb.S 两个程序，由于最后没有死循环，直接截止到最后一个循环执行的时间即可.（之后指令是 XXX）

在测试的时候需要注意，其它几个程序都没问题，但对于 quicksort 需要保证内存是同一次生成的。因为内存是随机生成的，对其它的程序的运行流程没有影响，但对于排序会造成影响。

同时在测试的时候，我的 cache 使用的同一个参数 (WAY_CNT=4, SET_ADDR_LEN=3, LINE_ADDR_LEN=3).

得到结果如下（矩阵乘法和排序皆使用默认参数）:

程序名	分支策略	时间 (ns)	加速比	正确次数	错误次数	总次数	正确率
matmul.s	BHT	613880	1.053	4342	282	4624	93.9%
	BTB	616016	1.049	4076	548	4624	88.1%
	baseline	646432	1	274	4350	4624	5.9%
quicksort.s	BHT	159484	1.020	10475	1908	12383	84.6%
	BTB	161692	1.006	8749	3634	12383	70.7%
	baseline	162710	1	7871	4512	12383	63.6%
btb.s	BHT	1260	1.616	98	3	101	97.0%
	BTB	1252	1.626	99	2	101	98.0%
	baseline	2036	1	1	100	101	1.0%
bht.s	BHT	1472	1.454	95	15	120	79.2%
	BTB	1524	1.404	88	22	120	73.3%
	baseline	2140	1	11	99	120	9.2%

Table 2: 性能测试

在上表 2 中，baseline 代表“没有分支预测的情况”。但实际上，由于在我前面的实现中，我实现了静态的分支预测（永远预测不跳转），所以 baseline 代表的是一直预测不跳转的性能（也因此有预测正确的个数和不正确的个数）.

观察上表我们可以发现:

1. BHT 大部分情况下都稍优于单纯的 BTB 实现

3.1 分支收益和分支代价

分析实验数据，btb.s 这一行 BHT 和 BTB 的对比能够很清晰地展现出预测所带来的收益。对于 BHT 和 BTB 两种实现策略，预测正确的次数差了 1，而时间差了 8ns。每一个时钟周期是 4ns，这刚好是两个时钟周期的时间。也就是说，预测错误的分支代价是 2.

理论上进行分析，每次错误的预测，都会使得实际到 EX 段才会得出正确的分支结果，这样我们需要 flush 调已经取入的错误指令（也就是下一个周期的 ID 和 EX），这样，我们会浪费掉两个流水线周期，和实验的结果相符。

总体的分支收益上，我们可以观察实验数据，两种预测策略在不同程序上都得到了一定加速比，且正确率也不错。因此而带来的分支收益是 □□□ × □□□ × $8ns$.

3.2 BHT vs BTB

我们可以发现，在快速排序和矩阵乘法中，BHT 都稍微优于 BTB. 只有在 btb.s 中 BTB 稍微更胜一筹。分析具体的原因，我们对比 btb.s 和 bht.s 这两个相对比较简单的程序，更容易分析。

观察这两个程序我们发现，bht.s 是一个双层循环，btb.s 是一个单层循环。在双层循环中，每一个内层循环的末尾都将是一个实际不跳转，内层循环的开头都将是一个实际跳转。如果使用 BTB 作为分支策略，那么每一次内层循环的末尾都将预测失败，且由于上一次内层循环的末尾预测跳转而实际不跳转，下一次开头时就会预测不跳转，但实际跳转。对于 bht 则没有这种情况