YOLO 卷积函数的 SMART 仿真

1、实验目的

通过将 YOLO 源码的卷积函数放在 SMART 上进行仿真和性能监测，进一步介绍嵌入式 C 裸机程序在 C910 上的开发过程和仿真原理。通过观测分支预测开关对程序性能的影响，理解分支预测在现代处理器中的重要性。通过指令分布统计，了解内存墙的概念，并进一步理解缓存在现代处理器中的重要性。

2、实验步骤（包括实验结果，数据记录、截图等）

1.文件核对和执行仿真

核对 lab9 文件夹下文件，core num 默认为 1，无需改动。在反汇编后查看 conv_test.s 文件内容，发现 guard_start()与 guard_end()函数的入口地址分别为 40’h1b50 和 40’h1c10，在 tb/tb.v 中修改开头定义为 `define GUARD_START_PC 40’h1b10 以及`define GUARD_END_PC 40’h1c10。然后删除 conv_test.s 文件，bkill 当前 job 再次仿真。

2.CPI、Cache 缺失率和分支预测统计

All on 的 run.log:

```
num_cycle is 5155777
num_instret is 9449136
CPI is 0.545635
num_conditional_branch_mis is 5009
num_L1_Dcache_read_access is 3795267
num_L1_Dcache_read_miss is 15851
num_L1_Dcache_write_access is 1898563
num_L1_Dcache_write_miss is 1828
num_L2_Dcache_read_access is 474246
num_L2_Dcache_read_miss is 4163
num_L2_Dcache_write_access is 123983
num_L2_Dcache_write_miss is 399
guard_cycle_count is    5156429
guard_inst_count is    8724884
**********************************************
*    simulation finished successfully       *
**********************************************
$finish called from file "../tb/tb.v", line 549.
$finish at simulation time           52192450
           V C S  S i m u l a t i o n  R e p o r t
Time: 52192845000 ps
```

Btb,l0btb off 的 run.log:

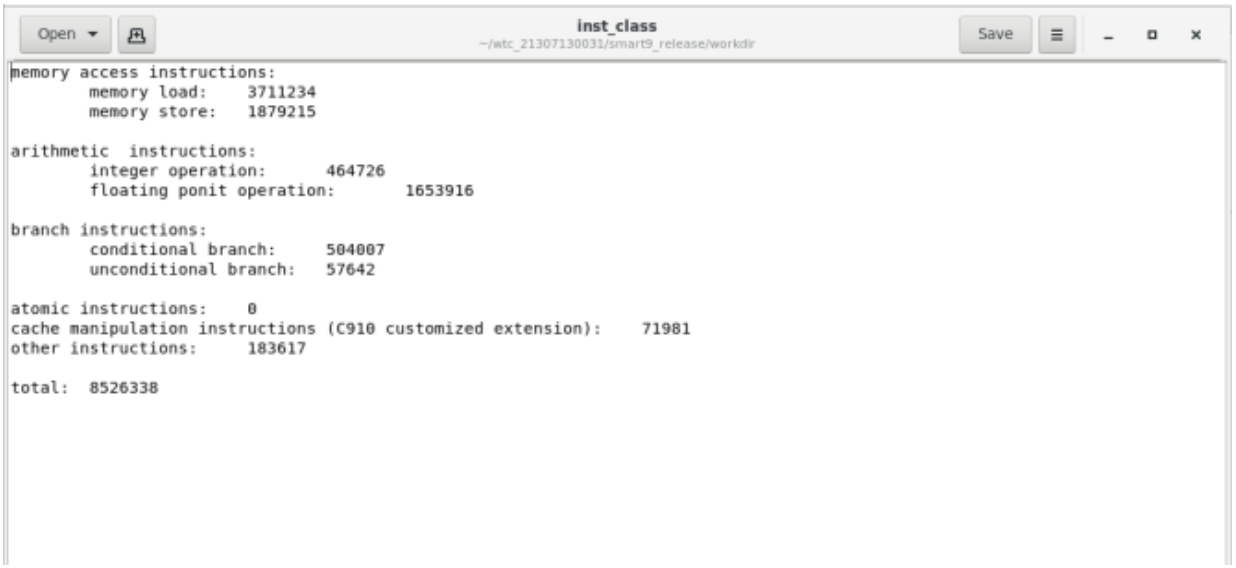
```
num_cycle is 5225076
num_instret is 9449136
CPI is 0.552969
num_conditional_branch_mis is 4439
num_L1_Dcache_read_access is 3780669
num_L1_Dcache_read_miss is 15687
num_L1_Dcache_write_access is 1897808
num_L1_Dcache_write_miss is 1858
num_L2_Dcache_read_access is 475590
num_L2_Dcache_read_miss is 4164
num_L2_Dcache_write_access is 123959
num_L2_Dcache_write_miss is 400
guard_cycle_count is    5225728
guard_inst_count is    8681067
**********************************************
*    simulation finished successfully       *
**********************************************
$finish called from file "../tb/tb.v", line 549.
$finish at simulation time           530514050
```

三次仿真后得到表格：

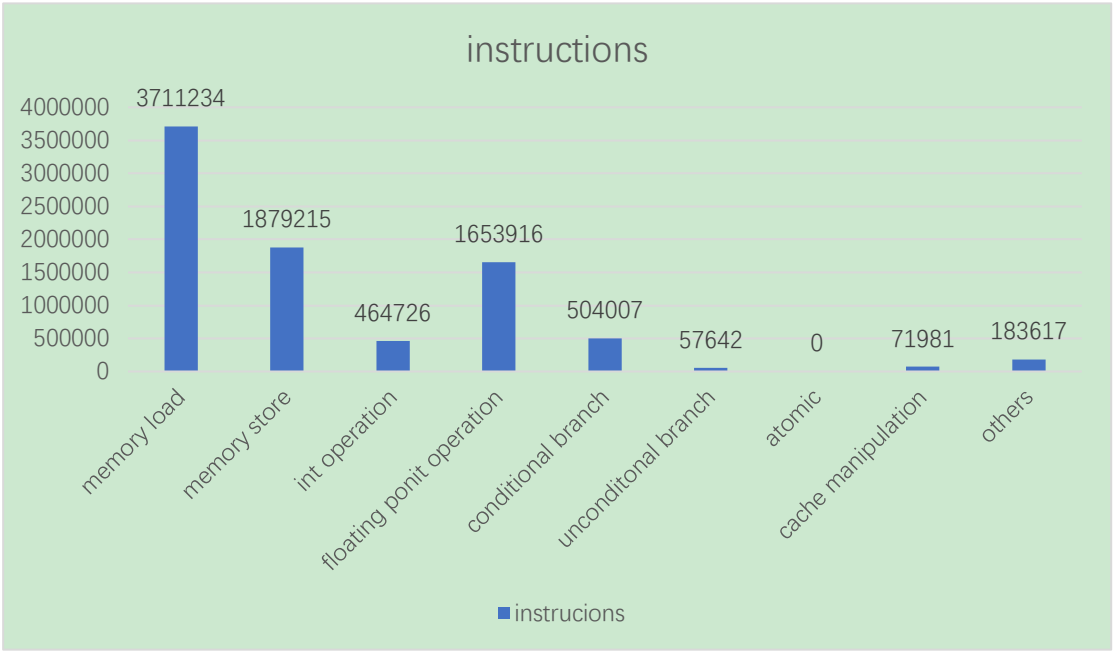
configure	all prediction off	BPE on, BTB off	all prediction on

cycle	8936645	5225076	5155777
insts	9449136	9949136	9949136
CPI	0.945763	0.552969	0.545635
conditional branch miss	253598	4439	5009
L1_Dread access	4579889	3780669	3795267
L1_Dread miss	12291	15687	15851
L1_Dread miss_rate	0.002683689	0.004149266	0.004159147
L1_Dwrite access	1920568	1897808	1898563
L1_Dwrite miss	1805	1858	1828
L1_Dwrite miss_rate	0.000939826	0.000979024	0.000961907
L2_Dread access	481926	475590	474246
L2_Dread miss	4183	4164	4163
L2_Dread miss_rate	0.008679756	0.008755441	0.008701759
L2_Dwrite access	122750	123959	123983
L2_Dwrite miss	400	400	399
L2_Dwrite miss_rate	0.003258656	0.003226873	0.00320786

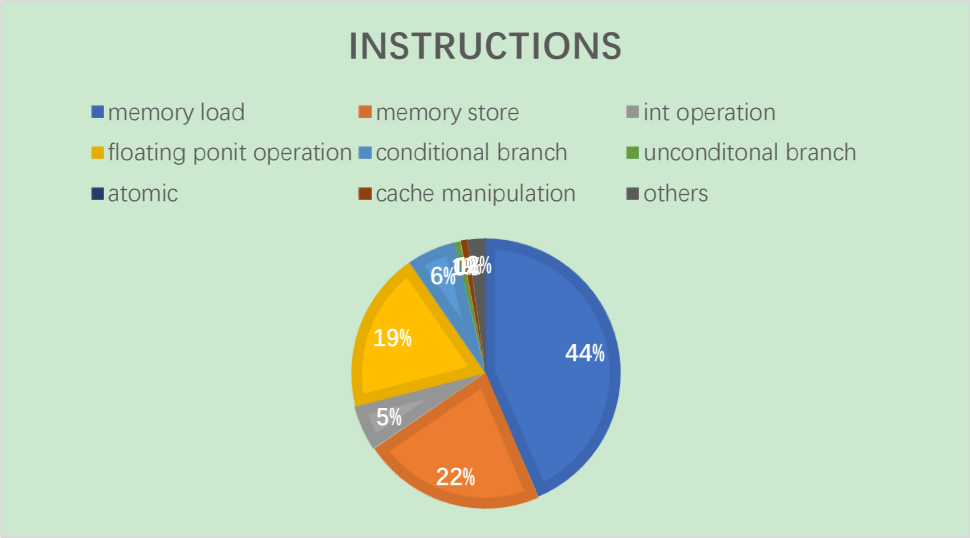
3. 指令分布统计绘图



绘出图形：



占比：



3、实验分析和总结

本实验详细讲解了 smart 平台运行 c 语言程序的步骤，从源码编译，内存划分与链接开始引入，讲解了以下内容：

(1)编译阶段 makefile 文件据链接脚本对段存储位置的划分（划分为两段），生成完全链接的二进制文件（.elf 文件），并通过仿真脚本生成可以直接被 testbench 读入的便于 RTL 仿真模型加载的文件

(2) 了解 SMART 是一个使用 irun 或 vcs 进行 RTL 仿真的平台，所提供的 CPU 核、总线、Memory 及其他外围模块均是 RTL 模型而非真实硬件。了解 Testbench 如何利用内存映射规则把指令编码和程序数据初始化到正确的内存位置。

(3) 使用 Verilog 的系统字符代替 printf 进行输出打印；了解 smart 平台如何对输入数据初始化

(4) 针对性能分析，在熟悉的 run.log 给出的指标以外，在源码中添加两个哨兵函数 guard_start 和 guard_end,准确测定 yolo 函数指令分布情况，以及总指令数，并根据 opcode 分析出不同指令类别

4、实验收获、存在问题、改进措施或建议等

一方面阐述了 FPGA 嵌入式系统运行程序与 RTL 仿真平台运行程序的不同点。另一方面了解了具体 smart 平台运行程序的过程，并通过对 yolo 函数监听，通过观测分支预测开关对程序性能的影响，理解了分支预测在现代处理器中的重要性。并通过指令分布统计，了解了内存墙的概念。