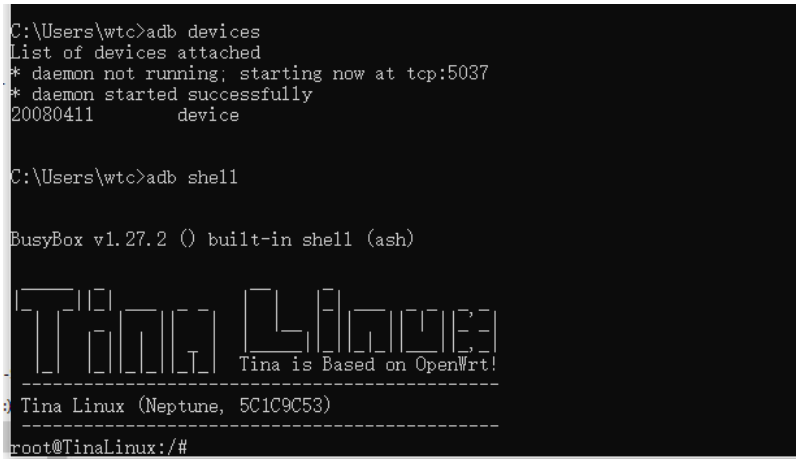
D1 开发板 YOLO 算法优化

1、实验目的

熟悉个 RISC-V 架构的硬件平台，并在之上启动一个 Linux 操作系统。然后在此之上运行 YOLO 算法，之后需要对卷积层进行优化，在此过程会使用嵌入的汇编代码进行性能监测，并对优化的性能进行分析，在这个过程中我们可以把书中学习到的缓存原理知识应用到实际的工程开发中，并真切地感受到其发挥的作用。最后通过 HHB 流程对 RISC-V 的向量扩展的应用有一个初步的认识。

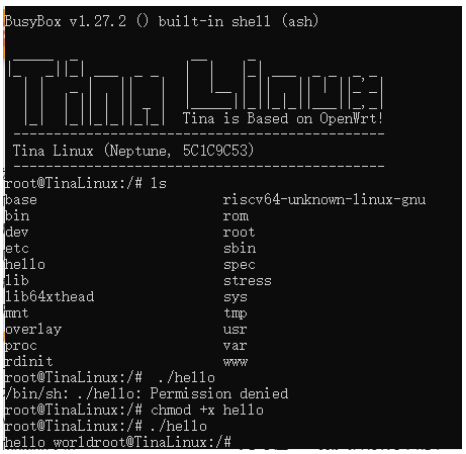
2、实验步骤（包括代码、实验结果、数据记录、截图等）

1. 将修改过的 Tina Linux 固件烧写到 D1 开发板中



2. 在 D1 开发板上运行 hello_world 的全流程：

将 hello 文件 adb push 到开发板上。然后运行。



3. 攥写带有缓存优化版本的卷积函数，并通过 qemu 的测试。（需要写出优化思路，以及优化的原理）：理想中通过减少循环层数，将 6 层循环减少至三层。考虑分别将内三层嵌套的循环合并成一层，外三层循环中关于行列的循环合并为 1 层从而减少循环层数。具体而言，可以先进行卷积核核数的循环，在这第一层循环中将 conv_size_h * conv_size_w 相乘，然后遍历所有的结果矩阵中元素，for (idx = 0; idx < conv_size_h * conv_size_w; idx++)，最后用 for (c = 0; c < kernel_size *

kernel_size * padding_size_c; c++)遍历每个卷积核核数对应的所有卷积核元素。然后在循环体内，根据 c 的值分别计算出对应的输入位置和权重值，从而进行卷积运算。

事实上，虽然循环次数减少，但实际上，因为运算和复制表达式增多，反而会导致运行速度减慢。效率无法提升。

要使得缓存优化，思路应当是使循环在访存过程中尽可能横向访问，即访问内存上连续相邻的数据。这样速度才能优化。

```
insts retire: 277052512
        Done.
Net 19 (none) forwarding...     Done.
Net 20 (route) forwarding...    Done.
Net 21 (convolutional) forwarding...
Function: convolutional_compute
cycles: 30704733857
insts retire: 30704737953
        Done.
Net 22 (convolutional) forwarding...
Function: convolutional_compute
cycles: 2225147584
insts retire: 2225151232
        Done.
Net 23 (yolo) forwarding...     Done.
data/dog.jpg: Predicted in 45.240879 seconds.
dog: 57%
car: 52%
truck: 56%
car: 62%
bicycle: 59%
```

4. 在 D1 开发板上进行三个版本的 YOLO 性能测试，并攥写分析报告(关于 GEMM 库版本的分析，同学们需要额外查找资料，重点放在 img2col 算法上。下面表格中统计的数据除了“执行时间”，其他的数据可以使用计算量最大的一个层 conv12 来代表。)

img2col 是一种实现卷积操作的加速计算策略。它能将卷积操作转化为 GEMM，从而最大化地缩短卷积计算的时间。

GEMM 是通用矩阵乘 (General Matrix Multiply) 的英文缩写，其实就是一般意义上的矩阵乘法

因为线性代数领域已经有非常成熟的计算接口（BLAS，Fortran 语言实现）来高效地实现大型的矩阵乘法，几乎可以做到极限优化。

将卷积过程中用到的所有特征子矩阵整合成一个大型矩阵存放在连续的内存中，虽然增加了存储成本，但是减少了内存访问的次数，从而缩短了计算时间。

因为矩阵的存储和计算其实都是非常规则的，很容易通过分布式和并行的方式来解决导致内存不够用或者计算速度非常慢的问题。

	Base	gemm	cache-opt
Cycles	48572945925	8095785459	8099484435
instructions	8122891602	5652096362	5652540667
L1D cache read access	1632823064	1607081422	1607190223
L1D cache read miss	270421043	2735025	2782890
L1D cache read hit ratio	0.857915733	0.998301033	0.998271468
L1D cache write access	25233491	803347082	803350001
L1D cache write miss	1028129	265004	265970
L1D cache write hit ratio	0.960850511	0.999670234	0.999669033
执行时间	140.340530	30.828560	30.833986

5. 使用平头哥 HBB 流程编译 mobilenet 生成两个版本的可执行文件（一个带向量扩展，一个不带扩展），并分别在 qemu 和 D1 开发板运行。另行选取三张图片重复上述过程，并将实验结果统计在如下表格中。

图片 2：  图片 3： 

表 1 在不同平台下不同输入的执行时间

	qemu_ref	qemu_c906	D1_ref	D1_c906
图片 1dog	31277.63672ms	8237.31055ms	41120.45312ms	358.04813ms
图片 2	31268.05078ms	8446.84766ms	41196.62891ms	357.66541ms
图片 3	31370.37500ms	8469.34863ms	41229.74609ms	357.38419ms

表 2 在不同平台下不同输入的分类结果

	qemu_ref	qemu_c906	D1_ref	D1_c906
图片 1	263(Pembroke, Pembroke Welsh corgi): 0.843262 264(Cardigan, Cardigan Welsh corgi): 0.117798	263(Pembroke, Pembroke Welsh corgi): 0.852051 264(Cardigan, Cardigan Welsh corgi): 0.110840	263(Pembroke, Pembroke Welsh corgi): 0.843262 264(Cardigan, Cardigan Welsh corgi): 0.117798	263(Pembroke, Pembroke Welsh corgi): 0.852539 264(Cardigan, Cardigan Welsh corgi): 0.110107

	248(Eskimo dog, husky): 0.011482 249(malamute, malemute, Alaskan malamute): 0.011482 250(Siberian husky): 0.006699	248(Eskimo dog, husky):0.012444 249(malamute, malemute, Alaskan malamute):0.010643 250(Siberian husky): 0.007019	248(Eskimo dog, husky): 0.011482 249(malamute, malemute, Alaskan malamute): 0.011482 250(Siberian husky): 0.006699	248(Eskimo dog, husky):0.012070 249(malamute, malemute, Alaskan malamute): 0.010483 250(Siberian husky): 0.006916
图片2	339(sorrel): 0.523438 268(Mexican hairless): 0.052216 345(ox): 0.050629 180(American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier): 0.046814 246(Great Dane): 0.043976	339(sorrel): 0.540039 268(Mexican hairless): 0.051025 345(ox): 0.050629 180(American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier): 0.046082 246(Great Dane): 0.043640	339(sorrel): 0.523438 268(Mexican hairless): 0.052216 345(ox): 0.050629 180(American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier): 0.046814 246(Great Dane): 0.043976	339(sorrel): 0.542480 268(Mexican hairless): 0.052032 345(ox): 0.050446 180(American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier): 0.045959 246(Great Dane): 0.043182
图片3	331(hare): 0.910645 330(wood rabbit, cottontail, cottontail rabbit): 0.061951 332(Angora, Angora rabbit): 0.022430 335(fox squirrel, eastern fox	331(hare): 0.917480 330(wood rabbit, cottontail, cottontail rabbit): 0.060547 332(Angora, Angora rabbit): 0.019958 335(fox squirrel, eastern fox squirrel, Sciurus niger): 0.000544 358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000386	331(hare): 0.910645 330(wood rabbit, cottontail, cottontail rabbit): 0.061951 332(Angora, Angora rabbit): 0.022430 335(fox squirrel, eastern fox	331(hare): 0.916016 330(wood rabbit, cottontail, cottontail rabbit): 0.061371 332(Angora, Angora rabbit): 0.020401 335(fox squirrel, eastern fox squirrel, Sciurus niger): 0.000538

	squirrel, Sciurus niger): 0.000605 358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000406		squirrel, Sciurus niger): 0.000605 358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000406	358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000394

Rabbit d1 c906:

```
339(sorrel): 0.523438
268(Mexican hairless): 0.052216
345(ox): 0.050629
180(American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier): 0.0468
246(Great Dane): 0.043976
root@TinaLinux:/hhb_outhorse# cd ..
root@TinaLinux:/# cd hhb_outrabbit/
root@TinaLinux:/hhb_outrabbit# chmod +x c_runtime_ref
root@TinaLinux:/hhb_outrabbit# chmod +x c_runtime_c906
root@TinaLinux:/hhb_outrabbit# ./c_runtime_c906 model.params data.0.bin
Run graph execution time: 357.38419ms, FPS=2.80

=== tensor info ===
shape: 1 3 224 224
data pointer: 0x3d852760

=== tensor info ===
shape: 1 1000 1 1
data pointer: 0x3d7be840
The max_value of output: 0.916016
The min_value of output: 0.000000
The mean_value of output: 0.001001
The std_value of output: 0.000842
 ============ top5: ===========
331(hare): 0.916016
330(wood rabbit, cottontail, cottontail rabbit): 0.061371
332(Angora, Angora rabbit): 0.020401
335(fox squirrel, eastern fox squirrel, Sciurus niger): 0.000538
358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000394
root@TinaLinux:/hhb_outrabbit#
```

Rabbit d1 ref:

```
Run graph execution time: 41229.74609ms, FPS=0.02

=== tensor info ===
shape: 1 3 224 224
data pointer: 0x3fd8318010

=== tensor info ===
shape: 1 1000 1 1
data pointer: 0xf41fc00
The max_value of output: 0.910645
The min_value of output: 0.000000
The mean_value of output: 0.000998
The std_value of output: 0.000833
 ============ top5: ===========
331(hare): 0.910645
330(wood rabbit, cottontail, cottontail rabbit): 0.061951
332(Angora, Angora rabbit): 0.022430
335(fox squirrel, eastern fox squirrel, Sciurus niger): 0.000605
358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000406
root@TinaLinux:/hhb_outrabbit#
```

Rabbit qemu c906

```
root@dd4950e18611:/home/example/c906/caffe_mobilenetv1/hhb_out# qemu-riscv64 -cp
u c906fdv c_runtime_c906 model.params data.0.bin
Run graph execution time: 8469.34863ms, FPS=0.12

=== tensor info ===
shape: 1 3 224 224
data pointer: 0x2212c0

=== tensor info ===
shape: 1 1000 1 1
data pointer: 0x18d3a0
The max_value of output: 0.917480
The min_value of output: 0.000000
The mean_value of output: 0.001001
The std_value of output: 0.000845
 ============ top5: ===========
331(hare): 0.917480
330(wood rabbit, cottontail, cottontail rabbit): 0.060547
332(Angora, Angora rabbit): 0.019958
335(fox squirrel, eastern fox squirrel, Sciurus niger): 0.000544
358(polecat, fitch, foulmart, foumart, Mustela putorius): 0.000386
```

3、实验分析和总结

实验中让学生熟悉个 RISC-V 架构的硬件平台，并在之上启动一个 Linux 操作系统。然后在此之上进一步优化 YOLO 算法的卷积层，同时使用嵌入的汇编代码进行性能监测，并对优化的性能进行分析。深刻认识到缓存优化的原理，在这个过程中把书中学习到的缓存原理知识应用到实际的工程开发中，并真切地感受到其发挥的作用。最后通过 HHB 流程对 RISC-V 的向量扩展的应用有一个初步的认识。同时熟悉了开发板的使用，提升综合能力。

4、实验收获、存在问题、改进措施或建议等

提升综合能力，了解了 GEMM 库版本在卷积层优化中的巨大作用和 HHB 流程对 RISC-V 的向量扩展的应用。进一步学会 gdb 使用，深化对卷积层函数理解。