实验报告

实验名称： **神经网络卷积层的计算与优化**

院系：**计算机学院 19 级计算机科学与技术**

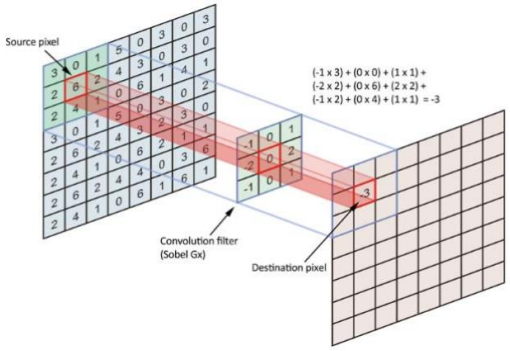
班级： **计科 2 班**

姓名： **施天予**

指导老师：**陈刚**

一、 实验内容

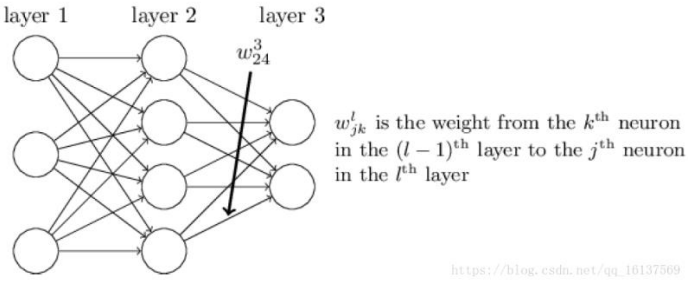
1. 使用 MIPS 汇编和 MIPS 仿真器，设计并实现一个普通整数卷积计算算子，要求有完整的输入输出，输入为 7*7*1 格式的张量，对应的卷积核一个，尺寸为 3*3*1，步长为 1，输出为经过卷积计算后的对应的 5*5*1 张量，要求计算结果正确。参考卷积操作图：

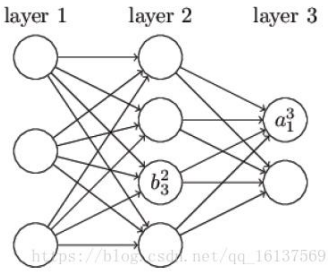


2. 设计并实现一个二值卷积计算算子，要求有完整的输入输出，输入为 7*7*16bit 格式的张量，对应的卷积核一个，尺寸为 3*3*16bit，输出为经过卷积计算后的对应的 5*5*1 的张量，要求计算结果正确。参考文献：论文 1、论文 2
3. 在 2 的基础上，将卷积层的偏置项、BN 层整合到同一层内。参考文献：论文 2 （3,4 二选一）
4. 在 3 的基础上，对二值卷积计算算子进行寄存器复用优化，参考文献：论文 1，论文 1 已经开源基于 ARM 的 BNN 代码，可以参照学习 （3,4 二选一）

二、设计及分析

神经网络前向推理引擎





记w_{jk}^l为第$l-1$层第k个神经元到第l层第j个神经元的权重，b_j^l为第l层第j个神经元的偏置，a_j^l为第l层第j个神经元的激活值（激活函数的输出）。不难看出，a_j^l的值取决于上一层神经元的激活：

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \tag{1}$$

将上式重写为矩阵形式：

$$a^l = \sigma(w^l a^{l-1} + b^l) \tag{2}$$

为了方便表示，记 $z^l = w^l a^{l-1} + b^l$ 为每一层的权重输入，(2)式则变为 $a^l = \sigma(z^l)$。

利用(2)式一层层计算网络的激活值，最终能够根据输入 X得到相应的输出$\hat{Y}$。

优化原理

二进制神经网络（BNN）是一种有前途的移动计算解决方案，因为它们的模型尺寸大大减小，算术运算复杂。通过使用按位运算来代替浮点运算，BNN 可以实现显着的模型压缩，并以较小的精度损失为代价来加快推理速度。PhoneBit，这是一种 GPU 加速的 BNN 推理引擎，用于基于 Android 的移动设备，它探索了移动 GPU 上 BNN 的软件和硬件级优化机会。PhoneBit 为 BNN 的执行提供了高度优化的框架，该框架探索了高效的 BNN 运算符以在移动设备上应用高级优化。使用这些运算符优化，可以在运行时以最小的内存占用量有效地执行 BNN 的推断。在 PhoneBit 框架中，几乎所有常见的 BNN 层类型都得到了很好的支持，例如卷积，池化，批处理规范化和密集（即完全连接）层。

PhoneBit 框架利用有效的二进制神经网络运营商在手机上进行高级优化。它实现了许多操作员级别的优化，包括：本地友好的数据布局，可实现有效的内存访问；带矢量化的位打包，可在通道方向上打包位；以及层集成，它将不同层中的多个操作集成在一起，重点是二进制卷积。

二值神经网络与普通卷积神经网络在卷积层计算上的差别与好处

神经网络二值化能够最大程度地降低模型的存储占用和模型的计算量，将神经网络中原本 32 位浮点数参数量化至 1 位定点数，降低了模型部署的存储资源消耗，同时极大加速了神经网络的推断过程。但二值化会不可避免地导致严重的信息损失，其量化函数不连续性也给深度网络的优化带来了困难。

在 BNN 中，权值被二进制化，这大大减少了内存大小和访问量。为了实现高效的卷积算子，二进制卷积运算中的点产生可以用 xor（乘法）和 popcount（累加）代替。

$$\vec{A} \cdot \vec{B} = Len - 2 \times (popcount\,(xor\,(a_i, b_i)))$$

卷积层的输入通常是图像，这与二进制卷积层对二进制输入的要求相冲突。在位平面 Ii 和二进制权值 W 上进行二进制卷积运算，将所有位平面的卷积相加得到输出 s。

$$s = \sum_{n=1}^{8} 2^{n-1} < I_i \cdot W >$$

三、实现及调试分析

1、 普通整数卷积 CNN

程序代码

```
        .data
arr1:   .word   1:101
arr2:   .word   1:101
arr3:   .word   1:101
space:  .asciiz " "
line:   .asciiz "\n"
        .text

        addi    $t0,$t0,7               # 输入的尺寸为7*7
        addi    $t1,$t1,3               # 卷积核的尺寸为3*3
        move    $s0,$0                  # 初始化
        move    $s1,$0
        move    $s2,$0
read1:  mult    $s0,$t0                 # 行号$s0 * 7
        mflo    $s2                     # $s2 = $s0 * 7
        add     $s2,$s2,$s1             # $s2 = $s2 + $s1，代表数组偏移量
        li      $v0,5                   # 输入一个元素
        syscall
        move    $t4,$v0                 # 将输入的元素放入数组$t4
        sll     $s2,$s2,2               # 因为一个输入4个字节，$s2*4是正确的偏移量
        sw      $t4,arr1($s2)           # 将$t4的值存到arr1相应位置中
        addi    $s1,$s1,1
        bne     $s1,$t0,read1           # 如果列号$s1小于7，继续循环
        move    $s1,$0                  # 如果列号$s1等于7，清零
        addi    $s0,$s0,1               # 行号$s0加1
        bne     $s0,$t0,read1           # 如果行号$s0小于7，继续循环

        move    $s0,$0                  # 初始化
        move    $s1,$0
        move    $s2,$0
read2:  mult    $s0,$t1                 # 行号$s0 * 3
        mflo    $s2                     # $s2 = $s0 * 3
        add     $s2,$s2,$s1             # $s2 = $s2 + $s1，代表数组偏移量
        li      $v0,5                   # 输入一个元素
        syscall
        move    $t4,$v0                 # 将输入的元素放入数组$t4
        sll     $s2,$s2,2               # 因为一个输入4个字节，$s2*4是正确的偏移量
        sw      $t4,arr2($s2)           # 将$t4的值存到arr2相应位置中
        addi    $s1,$s1,1
        bne     $s1,$t1,read2           # 如果列号$s1小于3，继续循环
        move    $s1,$0                  # 如果列号$s1等于3，清零
        addi    $s0,$s0,1               # 行号$s0加1
        bne     $s0,$t1,read2           # 如果行号$s0小于3，继续循环

```

```
45          sub     $t2, $t0, $t1
46          addi    $t2, $t2, 1             # 卷积计算结果$t2 = 7 - 3 + 1 = 5
47
48          move    $s0, $0                 # $s0 = i (循环变量)
49  for1:   move    $s1, $0                 # $s1 = j (循环变量)
50  for2:   move    $s2, $0                 # $s2 = k (循环变量)
51          move    $t6, $0                 # $t6清零，用于每次arr3的赋值
52  for3:   move    $s3, $0                 # $s3 = 1 (循环变量)
53  for4:   mul     $t3, $s0, $t2
54          add     $t3, $t3, $s1
55          sll     $t3, $t3, 2             # $t3 = i*5+j arr3的偏移量
56          add     $t4, $s0, $s2
57          mul     $t4, $t4, $t0
58          add     $t4, $t4, $s1
59          add     $t4, $t4, $s3
60          sll     $t4, $t4, 2             # $t4 = (i+k)*7+j+1 arr1的偏移量
61          mul     $t5, $s2, $t1
62          add     $t5, $t5, $s3
63          sll     $t5, $t5, 2           # $t5 = k*3+1 arr2的偏移量
64          lw      $t7, arr1($t4)        # $t7为arr1对应值
65          lw      $t8, arr2($t5)        # $t8为arr2对应值
66          mul     $t9, $t7, $t8         # $t9 = $t7 * $t8 临时变量
67          add     $t6, $t6, $t9         # $t6 = $t6 + $t9 arr3对应值
68          sw      $t6, arr3($t3)        # 将$t6的值存入arr3对应位置
69          addi    $s3, $s3, 1           # 1 = 1 + 1
70          bne     $s3, $t1, for4        # 如果1 == 3，跳出循环
71          addi    $s2, $s2, 1           # k = k + 1
72          bne     $s2, $t1, for3        # 如果k == 3，跳出循环
73          addi    $s1, $s1, 1           # j = j + 1
74          bne     $s1, $t2, for2        # 如果j == 5，跳出循环
75          addi    $s0, $s0, 1           # i = i + 1
76          bne     $s0, $t2, for1        # 如果i == 5，跳出循环
77
78          move    $s0, $0                 # $s0 = i (循环变量)
79  print1: move    $s1, $0                 # $s1 = j (循环变量)
80  print2: mul     $t3, $s0, $t2
81          add     $t3, $t3, $s1
82          sll     $t3, $t3, 2             # $t3 = i*5+j arr3的偏移量
83          lw      $a0, arr3($t3)
84          li      $v0, 1
85          syscall                         # 输出arr3对应值
86          la      $a0, space
87          li      $v0, 4
88          syscall                         # 输出空格
89          addi    $s1, $s1, 1             # j = j + 1
90          bne     $s1, $t2, print2        # 如果j == 5，跳出循环
91          la      $a0, line
92          li      $v0, 4
93          syscall                         # 输出回车
94          addi    $s0, $s0, 1             # i = i + 1
95          bne     $s0, $t2, print1        # 如果i == 5，跳出循环
96  end:
97          li      $v0, 10
98          syscall                         # 结束程序
```

调试及结果

```
3 0 1 5 0 3 0
2 6 2 4 3 0 3
2 4 1 0 6 1 4
3 0 1 5 0 3 0
2 6 2 4 3 2 3
2 4 1 0 6 2 1
2 6 2 4 4 0 3

-1 0 1
-2 0 2
-1 0 1
```

在程序中一个一个地输入以上 7*7 和 3*3 的张量，得到如下结果

```
-3 -3 6 -9 -2

-4 -5 10 -4 -4

-5 4 4 -5 -2

-3 -3 6 -4 -5

-2 -12 13 -2 -11
```

与写好的 C++程序比较发现结果正确

```
3 0 1 5 0 3 0
2 6 2 4 3 0 3
2 4 1 0 6 1 4
3 0 1 5 0 3 0
2 6 2 4 3 2 3
2 4 1 0 6 2 1
2 6 2 4 4 0 3
-1 0 1
-2 0 2
-1 0 1
-3 -3 6 -9 -2
-4 -5 10 -4 -4
-5 4 4 -5 -2
-3 -3 6 -4 -5
-2 -12 13 -2 -11
```

2、　二值卷积 BNN

方法分析

将 3*3*16bit 中的 3*3 位打包，通过异或操作和公式得到结果

$$\vec{A} \cdot \vec{B} = Len - 2 \times (popcount\,(xor\,(a_i, b_i)))$$

程序代码

```
        .data
arr1:   .word   1:3136
arr2:   .word   1:576
arr3:   .word   1:100
space:  .asciiz " "
line:   .asciiz "\n"
        .text

        addi    $t0,$0,7            # 输入的尺寸为7*7
        addi    $t1,$0,3            # 卷积核的尺寸为3*3
        addi    $s7,$0,16           # 通道数位为16

        move    $s0,$0              # 初始化（循环变量）
        mul     $s2,$t0,$t0
        mul     $s2,$s2,$s7         # 输入大小为7*7*16
read1:  sll     $t3,$s0,2
        li      $v0,5               # 输入一个元素
        syscall
        addi    $v0,$v0,1
        srl     $v0,$v0,1           # A=(a+1)/2
        sw      $v0,arr1($t3)       # 存入数组对应位置中
        addi    $s0,$s0,1
        bne     $s0,$s2,read1       # 循环次数未到，继续循环

        move    $s0,$0              # 初始化（循环变量）
        mul     $s2,$t1,$t1
        mul     $s2,$s2,$s7         # 卷积核大小为3*3*16
read2:  sll     $t3,$s0,2
        li      $v0,5               # 输入一个元素
        syscall
        addi    $v0,$v0,1
        srl     $v0,$v0,1           # B=(b+1)/2
        sw      $v0,arr2($t3)       # 存入数组对应位置中
        addi    $s0,$s0,1
```

```
35          bne     $s0, $s2, read2          # 循环次数未到，继续循环
36
37          sub     $t2, $t0, $t1
38          addi    $t2, $t2, 1              # 卷积结果的尺寸$t2 = 7 - 3 + 1 = 5
39
40          move    $s0, $0                  # $s0 = i （循环变量）
41  for1:   move    $s1, $0                  # $s1 = j （循环变量）
42  for2:   move    $s5, $0                  # $s5用于保存count函数后得到的1的数量
43          move    $s2, $0                  # $s2 = k （循环变量）
44  for3:   move    $t6, $0                  # $t6用于得出输入张量1与0连起来后的结果
45          move    $t7, $0                  # $t7用于得出卷积核1与0连起来后的结果
46          move    $s3, $0                  # $s3 = l （循环变量）
47  for4:   move    $s4, $0                  # $s4 = r
48  for5:   mul     $s6, $t0, $t0
49          mul     $s6, $s6, $s2
50          add     $t4, $s0, $s3
51          mul     $t4, $t4, $t0

52          add     $t4, $t4, $s1
53          add     $t4, $t4, $s4
54          add     $t4, $t4, $s6
55          sll     $t4, $t4, 2              # $t4 = (i+l)*7+j+r+k*7*7 arr1的偏移量
56
57          mul     $s6, $t1, $t1
58          mul     $s6, $s6, $s2
59          mul     $t5, $s3, $t1
60          add     $t5, $t5, $s4
61          add     $t5, $t5, $s6
62          sll     $t5, $t5, 2              # $t5 = l*3+r+k*3*3 arr2的偏移量
63
64          lw      $t8, arr1($t4)           # $t7为arr1对应值
65          lw      $t9, arr2($t5)           # $t8为arr2对应值
66
67          sll     $t6, $t6, 1
68          add     $t6, $t6, $t8            # 合并输入张量3*3*1中的1和0

69          sll     $t7, $t7, 1
70          add     $t7, $t7, $t9            # 合并卷积核3*3*1中的1和0
71
72          addi    $s4, $s4, 1              # r = r + 1
73          bne     $s4, $t1, for5           # 如果r == 3，跳出循环
74          addi    $s3, $s3, 1              # l = l + 1
75          bne     $s3, $t1, for4           # 如果l == 3，跳出循环
76          xor     $a0, $t6, $t7            # $t6与$t7异或
77          jal     count                    # 调用count函数，计算1的个数
78          add     $s5, $s5, $v0            # 将结果加到$s5中
79          addi    $s2, $s2, 1              # k = k + 1
80          bne     $s2, $s7, for3           # 如果k == 16，跳出循环
81
82          mul     $t3, $s0, $t2
83          add     $t3, $t3, $s1
84          sll     $t3, $t3, 2              # $t3 = i*5+j arr3的偏移量
85
```

```
86          mul     $s6, $t1, $t1
87          mul     $s6, $s6, $s7
88          sll     $s5, $s5, 1
89          sub     $s5, $s5, $s6
90          sub     $s5, $0, $s5            # 公式result=N-2*BCNT(XNOR(A*B))
91          sw      $s5, arr3($t3)          # 将$s5的值存入arr3对应位置
92
93          addi    $s1, $s1, 1             # j = j + 1
94          bne     $s1, $t2, for2          # 如果j == 5，跳出循环
95          addi    $s0, $s0, 1             # i = i + 1
96          bne     $s0, $t2, for1          # 如果i == 5，跳出循环
97
98          move    $s0, $0                 # $s0 = i （循环变量）
99  print1: move    $s1, $0                 # $s1 = j （循环变量）
100 print2: mul     $t3, $s0, $t2
101         add     $t3, $t3, $s1
102         sll     $t3, $t3, 2             # $t3 = i*5+j arr3的偏移量
```

```
103         lw      $a0, arr3($t3)
104         li      $v0, 1
105         syscall                         # 输出arr3对应值
106         la      $a0, space
107         li      $v0, 4
108         syscall                         # 输出空格
109         addi    $s1, $s1, 1             # j = j + 1
110         bne     $s1, $t2, print2        # 如果j == 5，跳出循环
111         la      $a0, line
112         li      $v0, 4
113         syscall                         # 输出回车
114         addi    $s0, $s0, 1             # i = i + 1
115         bne     $s0, $t2, print1        # 如果i == 5，跳出循环
116 end:
117         li      $v0, 10
118         syscall                         # 结束程序
119
```

```
120 count:                                  # count函数，用于计算一个二进制数中1的个数
121         move    $t4, $0
122         addi    $t5, $0, 0x00000001     # $t5用于判断二进制数末尾是否是1
123         move    $v0, $0
124         addi    $t8, $0, 9              # 循环次数
125         move    $t9, $0                 # 循环变量
126 loop:   and     $t4, $a0, $t5           # $t4用于保存按位与结果，得到1说明末尾位是1
127         add     $v0, $v0, $t4           # 计数
128         srl     $a0, $a0, 1             # 将二进制数右移一位
129         addi    $t9, $t9, 1
130         bne     $t9, $t8, loop          # 未到循环次数，继续循环
131         jr      $ra                     # 函数返回
132
```

调试及结果

为了输入方便，我设置每一通道 7*7 的输入和 3*3 的卷积核都相同

（真实测试将通道数设为 1）即假设 16 个通道都是以下输入

```
-1 1 1 1 1 1 -1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 -1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 -1 1 1 1

1 1 1
-1 -1 -1
1 1 1
```

调用 mips 程序得到如下结果

```
16 48 48 48 16
48 16 16 16 48
48 80 80 80 48
48 16 16 16 48
48 16 16 16 48

— program is finished running —
```

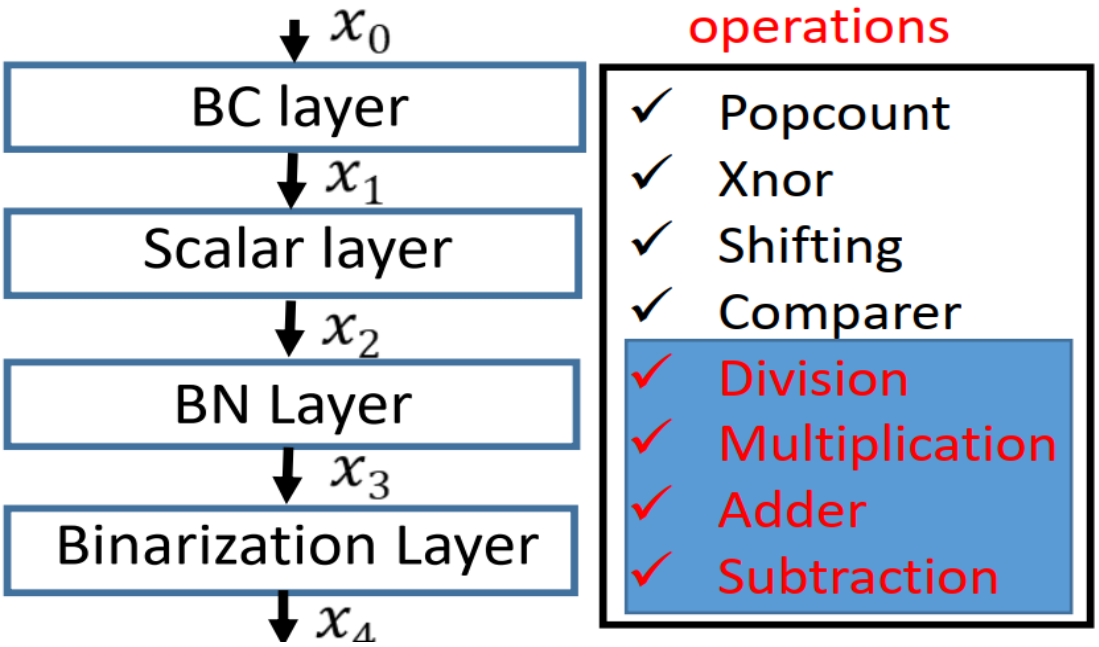
发现与 C++测试结果相同，程序正确！

```
sty@sty-virtual-machine:~$ g++ bnn.cpp
sty@sty-virtual-machine:~$ ./a.out
-1 1 1 1 1 1 -1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 -1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 -1 1 1 1
1 1 1
-1 -1 -1
1 1 1
16 48 48 48 16
48 16 16 16 48
48 80 80 80 48
48 16 16 16 48
48 16 16 16 48
```

3、 层整合

方法分析

根据实验 2，我们得到了 BC 层，通过一个方法可以不用这么多层的计算，直接得出最后我们要的二值化层。



$$x_2 = x_1 + b$$

$$x_3 = \gamma \cdot \frac{x_2 - \mu}{\sigma} + \beta$$

$$x_4 = \begin{cases} 1 & \text{if } x_3 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

层整合公式

$$\xi = \mu - \frac{\beta \cdot \sigma}{\gamma} - b$$

$$x_4 = \begin{cases} 1 & \text{if } x_1 \geq \xi \text{ and } \gamma > 0 \\ 0 & \text{if } x_1 < \xi \text{ and } \gamma > 0 \\ 1 & \text{if } x_1 \leq \xi \text{ and } \gamma < 0 \\ 0 & \text{if } x_1 > \xi \text{ and } \gamma < 0 \end{cases}$$

程序代码

```
        .data
arr1:   .word   1:3136
arr2:   .word   1:576
arr3:   .word   1:100
space:  .asciiz " "
line:   .asciiz "\n"
        .text

        addi    $t0,$0,7            # 输入的尺寸为7*7
        addi    $t1,$0,3            # 卷积核的尺寸为3*3
        addi    $s7,$0,16           # 通道数位为16

        move    $s0,$0              # 初始化（循环变量）
        mul     $s2,$t0,$t0
        mul     $s2,$s2,$s7         # 输入大小为7*7*16
read1:  sll     $t3,$s0,2
        li      $v0,5               # 输入一个元素
        syscall
        addi    $v0,$v0,1
        srl     $v0,$v0,1            # A=(a+1)/2
        sw      $v0,arr1($t3)        # 存入数组对应位置中
        addi    $s0,$s0,1
        bne     $s0,$s2,read1        # 循环次数未到，继续循环

        move    $s0,$0               # 初始化（循环变量）
        mul     $s2,$t1,$t1
        mul     $s2,$s2,$s7          # 卷积核大小为3*3*16
read2:  sll     $t3,$s0,2
        li      $v0,5                # 输入一个元素
        syscall
        addi    $v0,$v0,1
        srl     $v0,$v0,1            # B=(b+1)/2
        sw      $v0,arr2($t3)        # 存入数组对应位置中
        addi    $s0,$s0,1
        bne     $s0,$s2,read2        # 循环次数未到，继续循环

        sub     $t2,$t0,$t1
        addi    $t2,$t2,1            # 卷积结果的尺寸$t2 = 7 - 3 + 1 = 5

        move    $s0,$0               # $s0 = i（循环变量）
for1:   move    $s1,$0               # $s1 = j（循环变量）
for2:   move    $s5,$0               # $s5用于保存count函数后得到的1的数量
        move    $s2,$0               # $s2 = k（循环变量）
for3:   move    $t6,$0               # $t6用于得出输入张量1与0连起来后的结果
        move    $t7,$0               # $t7用于得出卷积核1与0连起来后的结果
        move    $s3,$0               # $s3 = l（循环变量）
for4:   move    $s4,$0               # $s4 = r
for5:   mul     $s6,$t0,$t0
        mul     $s6,$s6,$s2
        add     $t4,$s0,$s3
        mul     $t4,$t4,$t0
```

```
52          add     $t4, $t4, $s1
53          add     $t4, $t4, $s4
54          add     $t4, $t4, $s6
55          sll     $t4, $t4, 2             # $t4 = (i+1)*7+j+r+k*7*7 arr1的偏移量
56
57          mul     $s6, $t1, $t1
58          mul     $s6, $s6, $s2
59          mul     $t5, $s3, $t1
60          add     $t5, $t5, $s4
61          add     $t5, $t5, $s6
62          sll     $t5, $t5, 2             # $t5 = l*3+r+k*3*3 arr2的偏移量
63
64          lw      $t8, arr1($t4)          # $t7为arr1对应值
65          lw      $t9, arr2($t5)          # $t8为arr2对应值
66
67          sll     $t6, $t6, 1
68          add     $t6, $t6, $t8           # 合并输入张量3*3*1中的1和0
```

```
69          sll     $t7, $t7, 1
70          add     $t7, $t7, $t9           # 合并卷积核3*3*1中的1和0
71
72          addi    $s4, $s4, 1             # r = r + 1
73          bne     $s4, $t1, for5          # 如果r == 3，跳出循环
74          addi    $s3, $s3, 1             # l = l + 1
75          bne     $s3, $t1, for4          # 如果l == 3，跳出循环
76          xor     $a0, $t6, $t7           # $t6与$t7异或
77          jal     popcount                # 调用popcount函数，计算1的个数
78          add     $s5, $s5, $v0           # 将结果加到$s5中
79          addi    $s2, $s2, 1             # k = k + 1
80          bne     $s2, $s7, for3          # 如果k == 16，跳出循环
81
82          mul     $t3, $s0, $t2
83          add     $t3, $t3, $s1
84          sll     $t3, $t3, 2             # $t3 = i*5+j arr3的偏移量
85
```

```
86          mul     $s6, $t1, $t1
87          mul     $s6, $s6, $s7
88          sll     $s5, $s5, 1
89          sub     $s5, $s5, $s6
90          sub     $s5, $0, $s5            # 公式result=N-2*BCNT(XNOR(A*B))
91          sw      $s5, arr3($t3)          # 将$s5的值存入arr3对应位置
92
93          addi    $s1, $s1, 1             # j = j + 1
94          bne     $s1, $t2, for2          # 如果j == 5，跳出循环
95          addi    $s0, $s0, 1             # i = i + 1
96          bne     $s0, $t2, for1          # 如果i == 5，跳出循环
97
98
99          addi    $s2, $0, -48            # 参数b = -48
100         addi    $s3, $0, 1              # 参数γ = 1
101         addi    $s4, $0, 1              # 参数μ = 1
102         addi    $s5, $0, 1              # 参数σ = 1
```

```
        addi    $s6,$0,1                # 参数β = 1

        mul     $s6,$s6,$s5
        div     $s6,$s6,$s3
        sub     $s7,$s4,$s6
        sub     $s7,$s7,$s2             #ξ=μ-(β*σ)/γ-b

        move    $s0,$0                  # $s0 = i (循环变量)
print1: move    $s1,$0                  # $s1 = j (循环变量)
print2: mul     $t3,$s0,$t2
        add     $t3,$t3,$s1
        sll     $t3,$t3,2               # $t3 = i*5+j arr3的偏移量
        lw      $a0,arr3($t3)

        addi    $s2,$0,1                # 以下注释是层整合的公式
        beq     $a0,$s7,assign          # x4=1 if x1≥ξ and γ>0
        sgt     $s4,$a0,$s7             # x4=0 if x1<ξ  and γ>0

        sgt     $s5,$s3,$0              # x4=1 if x1≤ξ and γ<0
        seq     $s2,$s4,$s5             # x4=0 if x1>ξ  and γ<0
assign: move    $a0,$s2
        li      $v0,1
        syscall                         # 输出arr3对应值
        la      $a0,space
        li      $v0,4
        syscall                         # 输出空格
        addi    $s1,$s1,1               # j = j + 1
        bne     $s1,$t2,print2          # 如果j == 5，跳出循环
        la      $a0,line
        li      $v0,4
        syscall                         # 输出回车
        addi    $s0,$s0,1               # i = i + 1
        bne     $s0,$t2,print1          # 如果i == 5，跳出循环
end:
        li      $v0,10

        syscall                         # 结束程序

popcount:                               # popcount函数，用于计算一个二进制数中1的个数
        move    $t4,$0
        addi    $t5,$0,0x00000001       # $t5用于判断二进制数末尾是否是1
        move    $v0,$0
        addi    $t8,$0,9                # 循环次数
        move    $t9,$0                  # 循环变量
loop:   and     $t4,$a0,$t5             # $t4用于保存按位与结果，得到1说明末尾位是1
        add     $v0,$v0,$t4             # 计数
        srl     $a0,$a0,1               # 将二进制数右移一位
        addi    $t9,$t9,1
        bne     $t9,$t8,loop            # 未到循环次数，继续循环
        jr      $ra                     # 函数返回

```

调试及结果

设置 b = -48， γ = 1， μ = 1， σ = 1， β = 1

为了输入方便，我设置每一通道 7*7 的输入和 3*3 的卷积核都相同（真实测试将通道数设为 1）即假设 16 个通道都是以下输入

```
-1 1 1 1 1 1 -1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 -1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 -1 1 1 1

1 1 1
-1 -1 -1
1 1 1
```

程序结果如下

```
0 1 1 1 0
1 0 0 0 1
1 1 1 1 1
1 0 0 0 1
1 0 0 0 1

— program is finished running —
```

经检验发现结果正确！

四、实验总结

刚开始神经网络、卷积这些陌生的知识让我有些恐惧。但通过反复仔细阅读论文，还有广泛搜集相关资料，我逐步找到了解决的办法。第一个实验 CNN 比较简单，其实就是一个 4 重循环的计算。第二个实验 BNN 稍微复杂了一些，但通过位打包、异或、popcount 等操作最终我也解决了问题。第三个实验层整合看起来很复杂，但读懂了之后发现其实也很简单，只要用参数代入公式计算一下，就能得到正确的结果。通过这次大作业我学到了很多知识，但最重要的是让我提升了自己解决问题的能力。过程虽然痛苦，但成功的喜悦也让人振奋！