sass指令文件分析

sass文件的核心是 16 个HMMA指令，其余的指令都是为了准备HMMA的参数或者保存HMMA的运算结果。

- */0000/* IMAD.MOV.U32 R1, RZ, RZ, c[0x0][0x28] ;
  - 每个线程对应的 R1 和 RZ 寄存器都置零
- */0020/* S2R R2, SR_LANEID ;
  - 每个线程的特殊寄存器（线程号，0~31）写入到 R2 中。
- */0030/* IMAD.MOV.U32 R12, RZ, RZ, 0x10 ;
  - 每个线程的 R12 和 RZ 都被设为 0x10
- */0040/* SHF.R.U32.HI R3, RZ, 0x2, R2.reuse ;
  - R3 中保存线程号 / 4 （4x0, 4x1, 4x2 ... 4x7）
- */0050/* LOP3.LUT R29, R2, 0x3, RZ, 0xc0, !PT ;
  - R29 等于 R2 & 0x3，即 R29 等于线程号 % 4 （0 1 2 3）*8
- */0060/* LOP3.LUT R3, R3, 0x3, RZ, 0xc0, !PT ;
  - R3 等于 R3 % 4（4x0, 4x1, 4x2, 4x3）x2
- */0070/* SHF.R.U32.HI R2, RZ, 0x4, R2 ;
  - R2 等于 R2 / 16（16x0,16x1）
- */0080/* SHF.R.U32.HI R0, RZ, 0x1, R3 ;
  - R0 等于 R3 / 2（8x0,8x1）x2
- */0090/* IMAD.SHL.U32 R6, R3, 0x8, RZ ;
  - RZ 是0，R6 等于 R3x8+RZ（4x0, 4x8, 4x16, 4x24）x2
- */00a0/* LOP3.LUT R4, R2, 0x1, RZ, 0xc0, !PT ;
  - R4 等于 R2 的最后一位（16x0,16x1）
- */00b0/* IMAD R7, R0, 0x8, R29.reuse ;
  - R7 等于 R0x8+R29（0 1 2 3 0 1 2 3 8 9 10 11 8 9 10 11）*2
- */00c0/* LOP3.LUT R5, R6, 0x8, R29, 0xe2, !PT ;
  - R5 等于（R6 & 0x8）|（~0x8 & R29），也就是（0 1 2 3 8 9 10 11 0 1 2 3 8 9 10 11）x2
- */00d0/* IMAD R7, R4.reuse, 0x4, R7 ;
  - R7 等于 R4x4+R7（0 1 2 3 0 1 2 3 8 9 10 11 8 9 10 11） （4 5 6 7 4 5 6 7 12 13 14 15 12 13 14 15）
- */00e0/* IMAD R5, R4, 0x4, R5 ;
  - R5 等于 R4x4+R5（0 1 2 3 8 9 10 11 0 1 2 3 8 9 10 11） （4 5 6 7 12 13 14 15 4 5 6 7 12 13 14 15）
- */00f0/* IMAD.SHL.U32 R7, R7, 0x2, RZ ;

    - R7 等于 R7X2+RZ（0 2 4 6 0 2 4 6 16 18 20 22 16 18 20 22）（8 10 12 14 8 10 12 14 24 26 28 30 24 26 28 30）
- */0100/* SHF.L.U32 R5, R5, 0x1, RZ ;
    - R5 等于 R5x2，即（0 2 4 6 16 18 20 22 0 2 4 6 16 18 20 22）（8 10 12 14 24 26 28 30 8 10 12 14 24 26 28 30）
- */0110/* IMAD.WIDE.U32 R20, R5, R12, c[0x0][0x160] ;
    - R20 是在 c[0x0][0x160] 的基础上加上一些偏置（R5xR12，R12是常值0x10），wide 表示要写入 R20 和 R21，其中 R21 存放高位，R20 存放低位。
    - R20 是用来保存第 3 个 SET 运算里 A 矩阵的值
- IMAD.WIDE.U32 R12, R7, R12, c[0x0][0x168] ;
    - R12 是在 c[0x0][0x168] 的基础上加上一些偏置（R7xR12，R12是常值0x10）
    - R12 是用来保存第 3 个 SET 运算里 B 矩阵的值
- */0130/* LDG.E.128.SYS R24, [R20] ;
- */0140/* LDG.E.128.SYS R16, [R12] ;
- */0150/* LDG.E.128.SYS R20, [R20+0x10] ;
- */0160/* LDG.E.128.SYS R12, [R12+0x10] ;
    - 这四个指令是到 global memory 中寻址，R20和R12中相邻两个线程相差16x2，这里的16是指WMMA运算的矩阵一行有16个元素，2是指一个half等于两个字节，因为这里是从global memory取数据，而global memory是一个按字节排列元素的数组
- */0170/* CS2R R8, SRZ ;
- */0180/* CS2R R10, SRZ ;
- */01a0/* CS2R R4, SRZ ;
- */01b0/* CS2R R6, SRZ ;
    - 这四个指令是给四个累加寄存器赋初值
- */0190/* IMAD.SHL.U32 R2, R2, 0x4, RZ ;
    - R2 等于 R2x4，也就是（16x0,16x4）
- */01c0/* LOP3.LUT R29, R2, 0x4, R29, 0xe2, !PT ;
    - R29 等于 （R2 & 0x4）|（~0x4 & R29），即（0 1 2 3）x4，（4 5 6 7）x4
- */01d0/* */01e0/* */01f0/* */0200/*
    - 这四条指令完成 SET1 的计算
- */0220/* LOP3.LUT R17, R29, 0x2, RZ, 0xc0, !PT ;
    - R17 等于（R29 & 0x2），即（0 0 2 2）x8
- */0230/* IMAD.SHL.U32 R16, R3, 0x8, RZ ;
    - R16 等于 R3x8，也就是（4x0，4x8，4x16，4x24）x2
- */0240/* LOP3.LUT R29, R29, 0x5, RZ, 0xc0, !PT ;
    - R29 等于（R29 & 0x5），也就是（0 1 0 1）x4，（4 5 4 5）x4
- */0250/* IMAD.MOV.U32 R3, RZ, RZ, RZ ;
    - R3 置零

- */0260/* LEA R2, R0, R17, 0x3 ;
  - R2 等于 （R0 << 3）| R17，也就是（0 0 2 2）x2， （8 10 8 10）x2， （0 0 2 2）x2， （8 10 8 10）x2
- */0290/* LOP3.LUT R29, R16, 0x8, R29, 0xe2, !PT ;
  - R29 等于 （R16 & 0x8）|（~0x8 & R29），也就是（0 1 0 1 8 9 8 9）x2， （4 5 4 5 12 13 12 13）x2
- */02c0/* IMAD.WIDE.U32 R2, R29, 0x10, R2 ;
  - R2 等于 R29 x 0x10 + R2，也就是
  - 0x00000000 0x00000010 0x00000002 0x00000012 0x00000080 0x00000090 0x00000082 0x00000092 0x00000008 0x00000018 0x0000000a 0x0000001a 0x00000088 0x00000098 0x0000008a 0x0000009a 0x00000040 0x00000050 0x00000042 0x00000052 0x000000c0 0x000000d0 0x000000c2 0x000000d2 0x00000048 0x00000058 0x0000004a 0x0000005a 0x000000c8 0x000000d8 0x000000ca 0x000000da
- */0310/* LEA R12, P0, R2.reuse, c[0x0][0x170], 0x2 ;
  - R12 等于 R2 x 4 + c[0x0][0x170]
- */0340/* LEA.HI.X R13, R2, c[0x0][0x174], R3, 0x2, P0 ;
  - R13 作为高位，和 R12 一起拼凑成索引global memory的地址，这里其实就是取的地址的高32 位，引入谓词寄存器 P0 的目的是防止加法溢出。
- */0360/* STG.E.64.SYS [R12], R8 ;
- */0370/* STG.E.64.SYS [R12+0x80], R10 ;
- */0380/* STG.E.64.SYS [R12+0x10], R4 ;
- */0390/* STG.E.64.SYS [R12+0x90], R6 ;
- 四个寄存器在 C D 矩阵中的位置对应关系如下图，
-

C、D

STEP1-R8

STEP2-R10

STEP3-R4

STEP4-R6

R8	R4	R8	R4
R10	R6	R10	R6
R8	R4	R8	R4
R10	R6	R10	R6
R8	R4	R8	R4
R10	R6	R10	R6
R8	R4	R8	R4
R10	R6	R10	R6

- global memory 数组中的元素是按照字节排列的，所以[R12+...]中的偏置表示的是相差的字节数。