SASS Instruction variance analysis

# Generate SASS code

vd:

$(NVCC) --generate-line-info --cubin [$@.cu](mailto:$@.cu)

$(NVCC) -o $@ [$@.cu](mailto:$@.cu)

nvdisasm --print-line-info $@.cubin > [instruction/$@\_instruction.txt](mailto:instruction/$@_instruction.txt)

(--generate-line-info and –print-line-info to output the line number of source code)

# Source Code

|  |  |
| --- | --- |
| Kernel | Source Code (The first column is Line number) |
| vd |  |
| Vd\_shared |  |
| Vd\_constant |  |
| Vd\_texture |  |

# Analysis

We can divide the Sass code into 5 stage:

1: Start;

2: Compute\_ID;

3: Load from memory;

4: Do Addition;

5: End;

**Instruction counts**

**#compute\_id**: instructions used to compute thread id, memory placement dependent.

**#data\_objects**: data objects that involved with memory placement.

in this case, #data\_objects = 2 [only array a and b]

**#Static**: static instructions that independent with memory placement

in this case, #Static = stage 1+ stage4 + stage 5 = 1+4+1=6

**#memory\_load** : instructions that related to memory load.

global memory = 3

constant memory = 1

texture memory = 1

share memory = 1

#Total Instructions = #Static + #compute\_id + #data\_objects \* #meomry\_load

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stage | vd | vd\_constant | Vd\_tex | Vd\_shared |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 5 | 4 | 5 |
| 3 | 6 | 2 | 2 | 10 |
| 4 | 4 | 4 | 4 | 4 |
| 5 | 1 | 1 | 1 | 1 |
| Total instructions | 16 | 13 | 12 | 21 |

# SASS Code

|  |  |  |  |
| --- | --- | --- | --- |
| **//Load from global memory**  **//Data object 1**  **IMAD R6.CC, R3, R5, c[0x0][0x20]; IMAD.HI.X R7, R3, R5, c[0x0][0x24];**  **LD.E R2, [R6];**  **//Data object 2**  **IMAD R8.CC, R3, R5, c[0x0][0x28];**  **IMAD.HI.X R9, R3, R5, c[0x0][0x2c];**  **LD.E R0, [R8];** | **//Load from texture memory**  **//Data object 1**  **TLD.LZ.T R2, R0, 0x0, 1D, 0x1;**  **//Data object 2**  **TLD.LZ.T R3, R0, 0x1, 1D, 0x1;** | **//Load from constant memory**  **//Data object 1**  **LDC R4, c[0x2][R2];**  **//Data object 2**  **LDC R5, c[0x2][R2+0x1000];** | **//Load from shared memory**  **//Data object 1**  **LDS R2, [R3];**  **LDS R0, [R3+0x1000];** |