Pledge: *I pledge my honor that I have abided by the Stevens Honor System.* - **Eric Altenburg**

6.4: Page 579 - 580

(6.4.1)

Each iteration of the loop segment of the code requires 17 cycles repeated 999 times plus the additional 3 instructions before the loop gives a total cycle count of $17 * 999 + 3 =$ **16,986**. The break down of the stall insertions can be seen below:

```
        MOV     X10 #8000
        ADD     X2, X0, X10
        ADDI    X1, X0, #16
LOOP:   LDUR    D0, [X1, #-16]
        LDUR    D2, [X1, #-8]
        STALL
        STALL
        STALL
        STALL
        STALL
        STALL
        FADDD   D4, D0, D2
        STALL
        STALL
        STALL
        STALL
        STUR    D4, [X1, #0]
        ADDI    X1, X1, #8
        CMP     X1, X2
        B.LE    LOOP
```

6.7: Page 582

(6.7.1)

```
x=2  y=2  w=1  z=0
x=2  y=2  w=3  z=0
x=2  y=2  w=5  z=0

x=2  y=2  w=1  z=2
x=2  y=2  w=3  z=2
x=2  y=2  w=5  z=2

x=2  y=2  w=1  z=4
x=2  y=2  w=3  z=4
x=2  y=2  w=5  z=4
```

X and Y are held constant at 2, however, W and Z have some variance due to the possibility that some combination of X being 0 or 2 and Y being 0 or 2 could happen for a given moment.

6.16: Page 586

(6.16.1)

For an n-cube of order N, being 2^N, the network can handle $N-1$ broken links while still guaranteeing there be a path to all nodes in the network.

1: Non-textbook

Consider the following code, which multiplies two vectors that contain single precision floating point values:

```
1       for(i = 0; i < 100; i++) {
            c[i] = a[i] * b[i];
3       }
```

We define the Arithmetic Intensity of a code as the number of operations to run the program (for example in C) divided by the number of bytes accessed in the main memory. Arithmetic Intensity is a good measure of arithmetic operations versus the number of bytes transferred between memory and CPU so we can see whether the system is computation bound or memory bandwidth bound as we discussed in class. What is the arithmetic intensity of the above code?

$Arithmetic\,Intensity = \frac{\#operations}{\#bytes\,accessed}$

SP number is 4-bytes.

In each iteration of the above code, the total bytes read from the main memory is $2 * 4 = 8$-bytes (a[i] and b[i]).

Also, in each iteration, the total bytes written to the main memory is $1 * 4 = 4$-bytes (c[i]).

Total is 12-bytes. The number of operations done is 1 (one single multiplication).

$Arithmetic\,Intensity = \frac{1}{12} = 0.08\overline{3}$

$0.08\overline{3}$ *or* **$8.\overline{3}\%$**

2: Non-textbook

Assume a GPU that runs at 1.5 GHz with 16 SIMD processors, each having 16 single –precision FP units. This GPU is supported by a 100GB/s off-chip memory. Assume all memory latencies can be hidden. Assume each operation is an addition.

(a) Ignoring memory bandwidth, what is the peak SP FP operation in GFLOP/sec.?

(b) Is this throughput sustainable given the bandwidth? Justify your answer.

(a)

Peak SP FP:

$1.5 * 16 * 16 = 384$ GFLOP/sec.

(b)

Assuming each SP operation requires 4-byte two operands and outputs only one 4-byte result, the memory bandwidth would be:

$\frac{12\,bytes}{1\,FLOP} * \frac{384\,GFLOP}{1\,sec.} = 4.608$ TB/sec.

However, since 4.608 TB/sec. is far greater than the 100 GB/sec. meaning that the throughput is not sustainable, but it is still achievable using on-chip cache in short-bursts.

3: Non-textbook

Assume a GPU architecture that contains 10 SIMD processors. Each SIMD instruction is 32 bits, and each SIMD processor contains 8 lanes for single precision arithmetic, and load/store instructions, meaning that each non-diverged SIMD instruction can produce 32 results every 4 cycles. Assume a kernel that has divergent branches that causes, on average 80% of threads to be active. Also assume that 70% of all instructions are SP arithmetic and 20% load/store. Because not all memory latencies are covered, assume an average SIMD instruction issue rate of 0.85. Assume the GPU has a clock speed of 1.5 GHz. Compute the throughput, in GFLOP/sec, for this code on this GPU.

$$\begin{aligned} Throughput &= (\text{clock speed}) * (\text{average \# of threads active}) * (\text{instruction issue rate}) \\ &\quad * (\text{SP instruction percentage}) * (\text{\# of SIMD processors}) * (\text{\# of results per cycle}) \\ &= 1.5 * 0.8 * 0.85 * 0.7 * 10 * 8 \\ &= 57.12 \text{ GFLOP/sec.} \end{aligned}$$