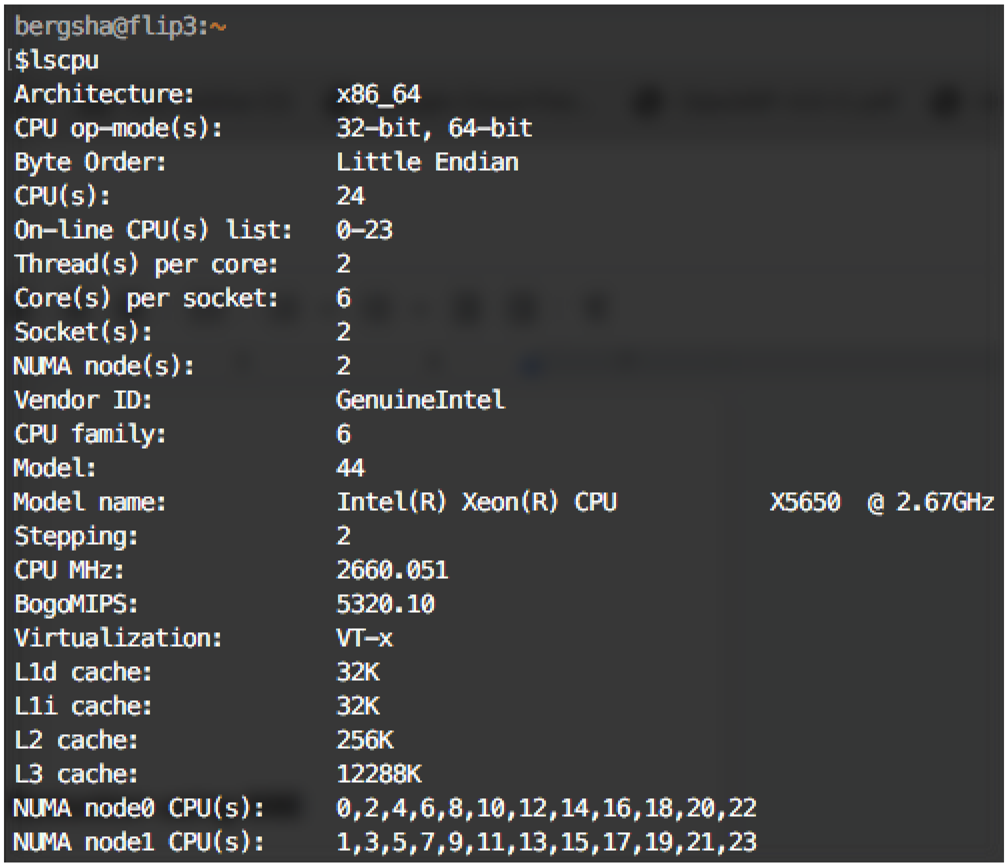
Shawn Berg
bergsha@oregonstate.edu
May 14th, 2019
CS 475

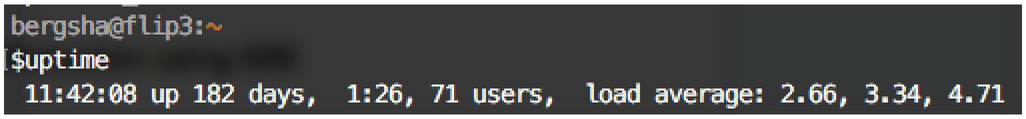
Project 4
Vectorized Array Multiplication and Reduction using SSE
Writeup

Machine:
This project was run on OSU flip3. The details of this machine are below:



```
bergsha@flip3:~
[$lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Thread(s) per core:    2
Core(s) per socket:    6
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 44
Model name:            Intel(R) Xeon(R) CPU           X5650  @ 2.67GHz
Stepping:              2
CPU MHz:               2660.051
BogoMIPS:              5320.10
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              12288K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15,17,19,21,23
```

And the uptime for the machine during the trials was roughly:



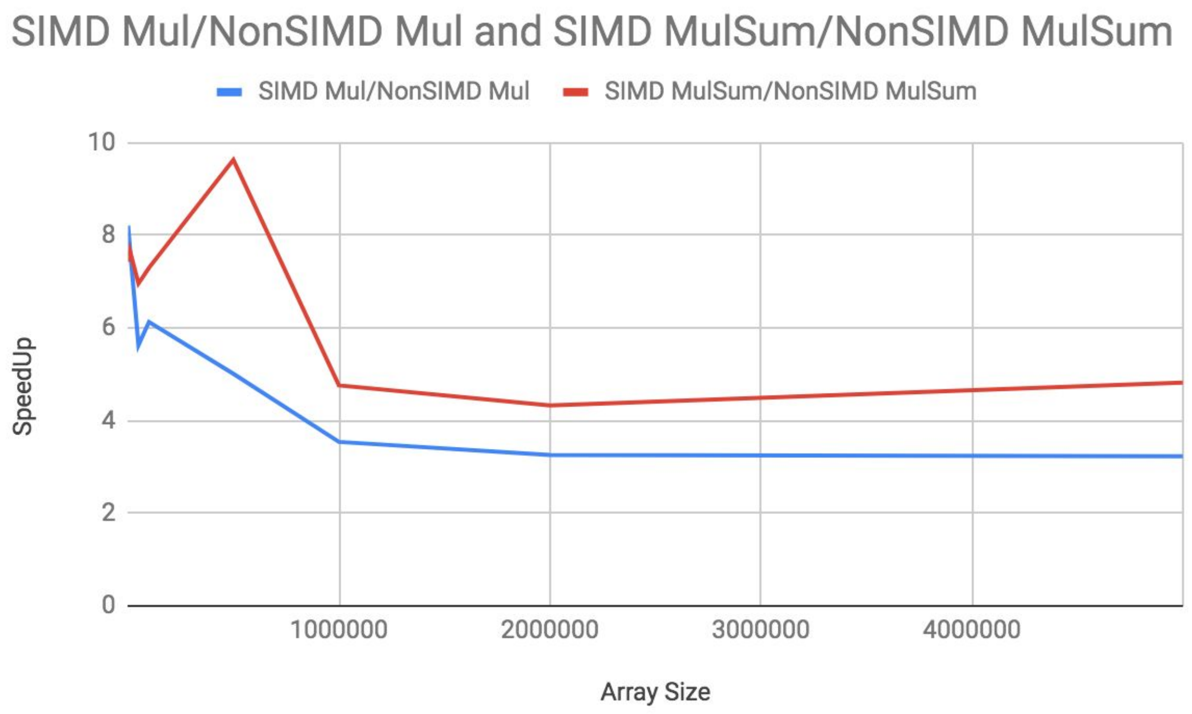
```
bergsha@flip3:~
[$uptime
 11:42:08 up 182 days,  1:26, 71 users,  load average: 2.66, 3.34, 4.71
```

Table (Speedup values as a function of array size)

	SIMD Mul/NonSIMD Mul	SIMD MulSum/NonSIMD MulSum
1000	8.21	7.43
10000	7.71	7.66

50000	5.63	6.96
100000	6.13	7.3
500000	5.01	9.63
1000000	3.54	4.76
2000000	3.26	4.33
5000000	3.23	4.82

Graph 1 (Speedup as a function of array size)



Commentary

1. What patterns are you seeing in the speedups? **Both speed ups start out relatively high, and then drop at an array size ~1,000,000 and then level off.**
2. Are they consistent across a variety of array sizes? **The speedups are consistent after an array size of ~1,000,000. Prior to this size, there is some variation in the speedup value.**
3. Why or why not, do you think? **Prior to an array size of ~1,000,000 the variation can be attributed to noise. The variation tends to occur more frequently with smaller array sizes and could be attributed to a variety of factors. After an array size of**

~1,000,000, the speedups level out and could perhaps be expected to trend downwards due to cache constraints.

4. Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array-mutiplication? **When the speed-up is < 4.0, the likely cause is the cache being unable to keep up. When the speed-up is > 4.0, the likely cause not a large enough array size to reach the point where the cache gets bogged down.**
5. Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array-mutiplication-reduction? **When the speed-up is < 4.0, the likely cause is the cache being unable to keep up. When the speed-up is > 4.0, the likely cause not a large enough array size to reach the point where the cache gets bogged down. Another factor is the reduction step. Adding into the xmm register is much more efficient than standard C code.**