**SIMD Intrinsics Lab using Intel AVX**

**An Intel processor is required for this lab.**

1. Run the attached matrix multiplication code provided in the lab using gcc with -O3.
2. Do you notice any performance difference with intrinsics (function dgemmIntrin) and without intrinsics (function dgemm)? Document your findings about the time taken and FLOPS [1].

Vary the size of the matrix for experiments using the macro:

#define SIZE 1024

1. “Check Assembly Code” folder contains matmul.c and matmulIntrinsic.c. First, inspect the **assembly** code with dgemm separately and then for dgemmIntrin. Check for vector instructions (vaddsd, vaddpd, vmulpd, vmovapd, etc) in the assembly code.

Compiling code with intrinsics: gcc -O3 -o prog -mavx matmul\_intrinsic.c

Running code: ./prog

Generating **assembly** code: gcc -O3 -mavx **-S** matmul\_intrinsic.c

1. Convert the following **vecAdd** function by using Intel Intrinsics (AVX) as shown in matrix multiplication.

Measure the GIGA FLOPS (GFLOPS) [1]. Formula shown after code as well as in the attached matrix multiplication code.

This function is already present in file “arrayLab.c”.

Note: For intrinsics, include header file #include<immintrin.h>

void vecAdd(int n, double \*A, double \*B, double \*C)

{

int i;

for(i = 0; i<n; i++)

{

C[i] = C[i] + A[i] \* B[i];

}

}

1. Polynomial evaluation using SIMD instructions was discussed in class (see slides). An example is provided here below:

3x2 + 2x – 3

If x = 1,

the polynomial will evaluate to 3.1.1 + 2.1 – 3 = 3 + 2 – 3 = 2.

Coefficients are {3, 2, -3}.

“evalPoly.c” file contains function “evaluate” to solve a polynomial given an input x and coefficients of the polynomial. Lab work is to implement “evaluateSIMD” function that uses intrinsics to evaluate the polynomial.

**Formula for calculating GFLOPs**

1. Note down the time taken and input size (n).
2. numOps ← Calculate the number of arithmetic operations; e.g., floating point additions and multiplications. For matrix multiplication, numOps is 2\*n\*n\*n.
3. FLOPS = numOps/time\_spent
4. Divide by 10^9 to get Giga FLOPS.
5. **SIMD Intrinsic Function Implementation, 10 points.**

The function computeArrays shown below does not use intrinsic functions for load, store, add, mul. Rewrite the same function computeArrays using intrinsics.

Use a large array as follows to measure execution time of your new function and compare it with the execution time of the function below. The challenge here is to get faster execution time by using intrinsics.

int NUM\_ELEMENTS = 1024\*1024\*128;

Measure the GIGA FLOPS (GFLOPS) and show your calculation in a word or text file. (2 points)

Hint: Use Broadcast function.

void computeArrays(int n, double \*A, double \*B, double \*C)

{

int i,j;

double a0, c0;

for(i = 0; i<n; i++)

{

c0 = C[i];

a0 = A[i];

for(j = 0; j<n; j++)

{

c0 = c0 + a0 \* B[j];

}

C[i] = c0;

}

}

**References**

[1] FLOPS wikipedia entry <https://en.wikipedia.org/wiki/FLOPS#:~:text=In%20computing%2C%20floating%20point%20operations,than%20measuring%20instructions%20per%20second>.

[2] Intel Intrinsics AVX, https://software.intel.com/sites/landingpage/IntrinsicsGuide/#techs=AVX