1 About Me

I am a first year EECS graduate student, working on program optimization and correctness, specifically on scientific programs involving floating-point computations. I have a decent background in computer programming and I have done some parallel programming before, but the software tools used in this course such as OpenMP or MPI would be something totally new to me. I expect to improve my programming skills through this course and learn more about software tools available for parallel computing. I am also interested in knowing more about applications of high performance computing (HPC) in science research and the current research state of HPC and parallel computing.

2 Parallel Application: Shufps Instruction

Consider the problem of 4x4 matrix transpose. A sequential algorithm looks as follows, which costs $O(n^2)$ computations where $n$ is the size of the matrix.

```c
int [16] transpose (int [16] M) {
    int [16] T = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            T[4*i+j] = M[4*j+i];
    return T;
}
```

One can utilize SIMD instruction (which stands for single instruction, multiple data) to perform computations on multiple data in one instruction, resulting in significant performance improvement, 4 computations at once. In the problem of matrix transposition, one can use the `shufps` instruction to change the positions of 4 matrix entries at once time. The `shufps` instruction requires 2 arrays of 4 elements and 1 mask. `shufps` selects 2 elements from each arrays based on the mask. 2 elements from the first array are copied to the lower 2 elements in the destination array and 2 elements from the second operand are copied to the higher 2 elements in the destination array. For a more detail explanation of `shufps`, see [1]. The code utilizing `shufps` requires only $O(n)$ computation is as follows.

```c
int [16] trans (int [16] M) {
```
A challenge here is apparently the complexity of \texttt{shufps} instruction semantic which results in convoluted code which might be hard to debug and maintain. It is also unclear for me how to scale this approach for bigger \( n \).

\textbf{References}