# Program 1

# Report

Theoretical maximum speedup due to vectorization is 8 times

256-bit vector registers divide by 32-bit floats => operation done on them at same time is a 8 floats

| | user time (seconds) | system time (seconds) | sum of user time and system time | speedup | max resident set (kB) |
|---|---|---|---|---|---|
| mmnovec | 337.02 | 0.068 | 337.088 | | 788160 |
| mmvec | 163.364 | 0.072 | 163.436 | 2.062507648 | 788024 |
| | | | | | |
| mmnovec | 334.58 | 0.068 | 334.648 | | 788116 |
| mmvec | 163.644 | 0.076 | 163.72 | 2.044026387 | 788104 |
| | | | | | |
| mmnovec | 338.236 | 0.068 | 338.304 | | 788032 |
| mmvec | 189.92 | 0.068 | 189.988 | 1.780659831 | 788012 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Average | mmnovec | 336.612 | 0.068 | 336.68 | | 788102.667 |
| | mmvec | 172.309333 | 0.072 | 172.3813333 | 1.962397955 | 788046.667 |

I look over 3 ways to Vectorization

**Pointer disambiguation**: since I didn't use any pointer I use stack array so there is no use in this method

**Interprocedural optimization**: this one from what I understand will be done automatic beside there is not a lot of small function in my code.

**Data aligned**: This is the one I used. Since I created 3 arrays of size 8192 it is better to have memory and data aligned next to each other

Also, to reduce dependency I move `c[i][j] = tmp;` out side of the k loop

```
1   for(int i = 0; i<n; i++)
2   {
3       for(int j = 0; j<n; j++)
4       {
5           tmp = 0;
6           for(int k = 0; k < n; k++)
7           {
8               tmp += A[i][k] * B[k][j];
9           }
10          c[i][j] = tmp;
11      }
12  }
```

Conclusion:

Vectorization is a very big topic.

In my opinion, you can vectorization this program 3 times or even 4 times faster by using maybe malloc array then you can apply more way of vectorization for this program.

Also, that if your program is less dependency in loop you can make them run faster.