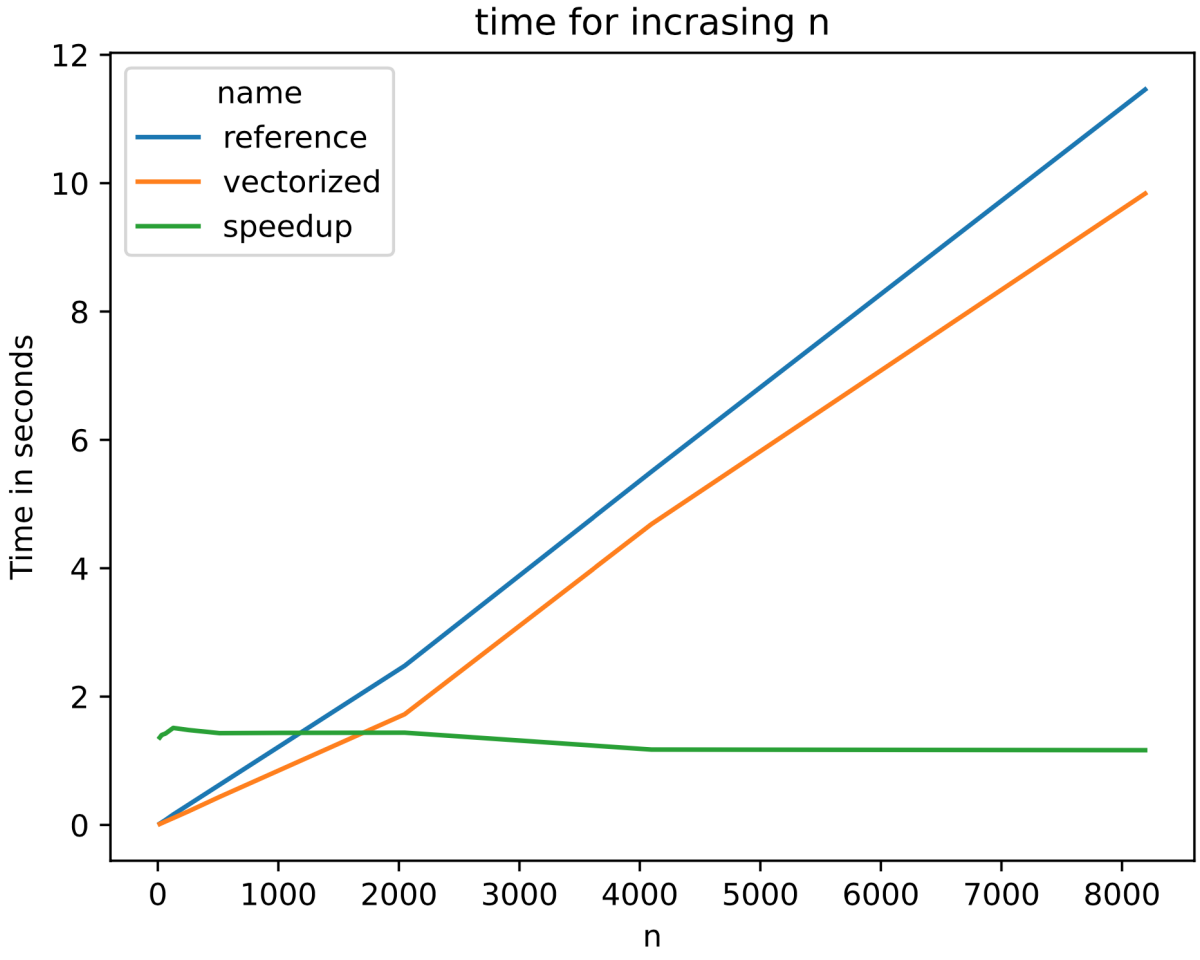
The program takes n (vector length) as input paramenter, and executes the given code.
The vector are initialized as follows:
`a = 1, b = 2, c = 3`
The expected value of `a[i]` is 600001 (23100000 + 1). This gets checked at the end of the program by assertions over each element of the vector.

Vectorization is enabled by using `-ftree_vecortize` (default with `-O3`).
To get information about what was being vectorized `-fopt-info-vec` can be used. The output is:

```
auto_vec.c:38:5: optimized: loop vectorized using 16 byte vectors (init)
auto_vec.c:38:5: optimized:  loop versioned for vectorization because of possible
aliasing (init)
auto_vec.c:48:9: optimized: loop vectorized using 16 byte vectors
auto_vec.c:48:9: optimized:  loop versioned for vectorization because of possible
aliasing
```

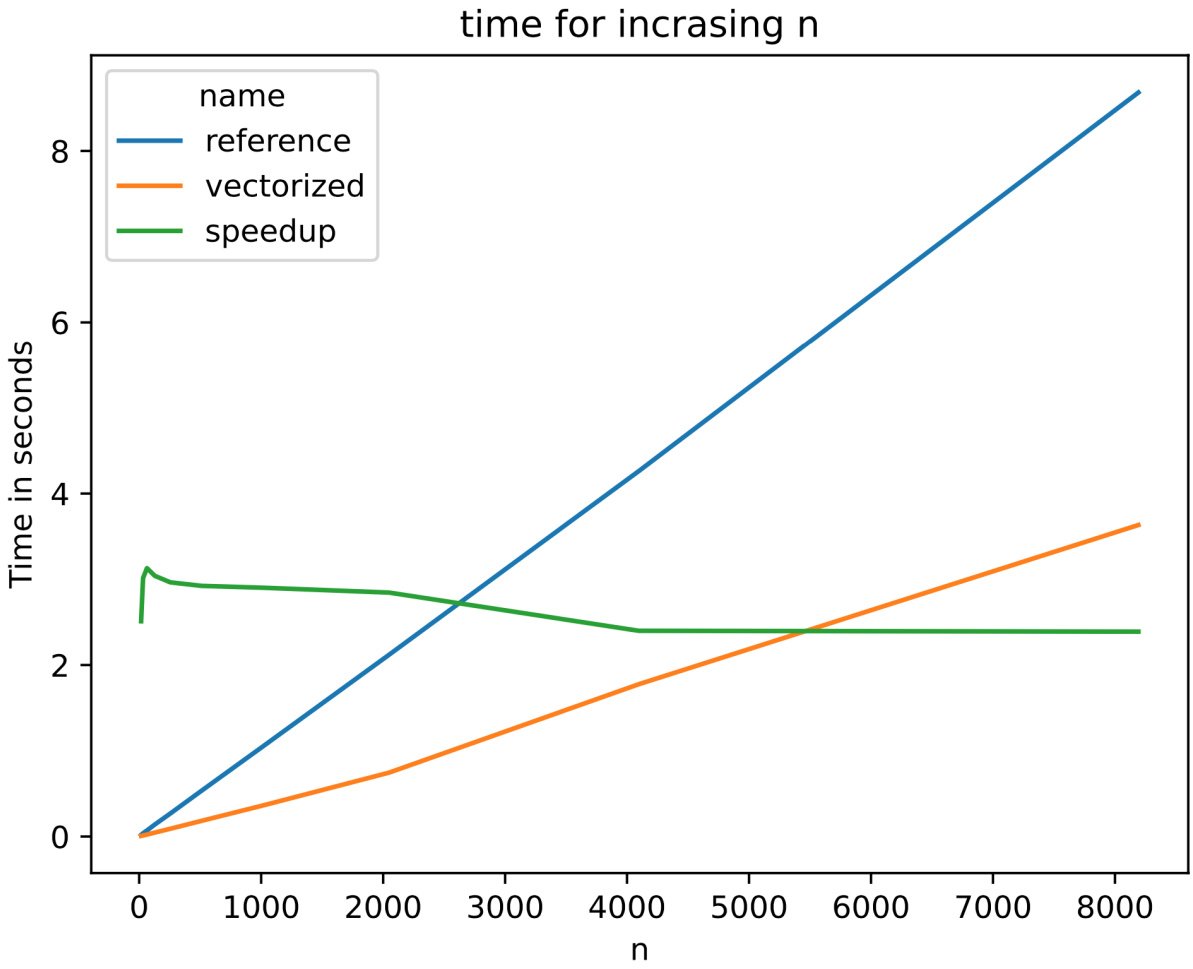
Adding the `restrict` keyword to the parameter of the funnction gets rid if the loop versionisation, but doesn't increase performance.

The vectorized version still gives correct results.
On the lcc2, the execution time increases pretty much linear, with increasing vector size.
The speedup increases first for small n, and then slowly decreases again, the bigger n gets.



On my local system, there is similar behavior, but the performance increase is much higher.
On the lcc2, there is a speedup of roughly 140%. It increases as the vector length gets bigger, and

decreases again for big n.
On my local system (OSX / intel) this behaivior is similar, but the speedup is between 240% and 300%, depending on n.



The output of perf stat for the different masks of event C7 is:

```
Performance counter stats for './reference 2048':

        8182260841      r02C7:u
(33.37%) -- number of SSE scalar-single instructions retired
                0      r04C7:u
(33.35%) -- number of SSE2 packed-double instructions retired
            12323      r08C7:u
(33.35%) -- number of SSE2 scalar-double instructions retired
             6392      r10C7:u
(33.32%) -- number of SSE2 vector instructions retired
        8197440673      r1fC7:u
(33.28%) -- overall number of SIMD instructions retired

       2.497522500 seconds time elapsed

       2.494361000 seconds user
       0.002000000 seconds sys


 Performance counter stats for './vectorized 2048':
```

```
        1025086552      r01C7:u
(33.30%) -- number of SSE packed-single instructions retired
                0       r02C7:u
(33.34%) -- number of SSE scalar-single instructions retired
                0       r04C7:u
(33.41%) -- number of SSE2 packed-double instructions retired
        2046289980      r08C7:u
(33.36%) -- number of SSE2 scalar-double instructions retired
        2050316230      r10C7:u
(33.30%) -- number of SSE2 vector instructions retired
        3075415709      r1fC7:u
(33.30%) -- overall number of SIMD instructions retired

       1.738455197 seconds time elapsed

       1.734788000 seconds user
       0.002000000 seconds sys
```

For the reference version, almost all instructions are scalar-single (instructions for one single-persision float), as expected.
The vectorized version has packed-single instructions (multiple single-persision float instructions), as expected,
but also scalar-double instructions, which is interesting, as all instructions used in the algorithm are single-persision float instructions.