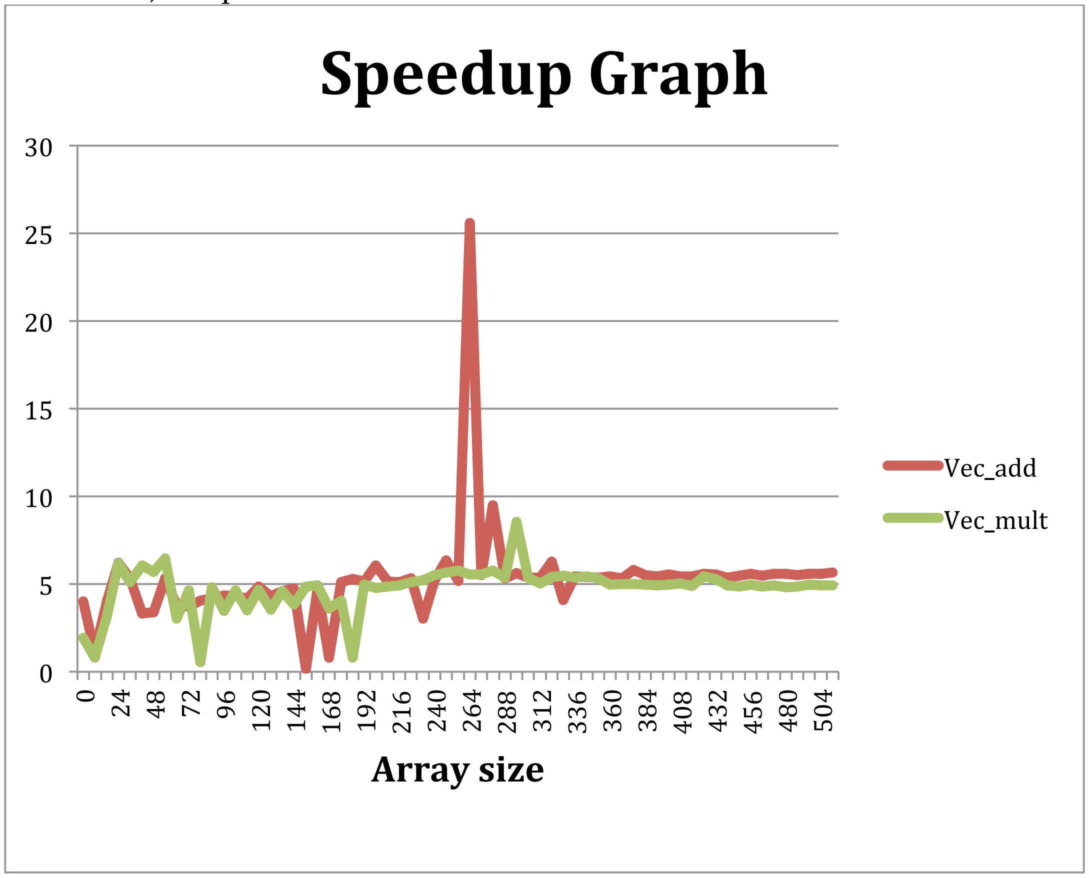
Problem 2 Table:

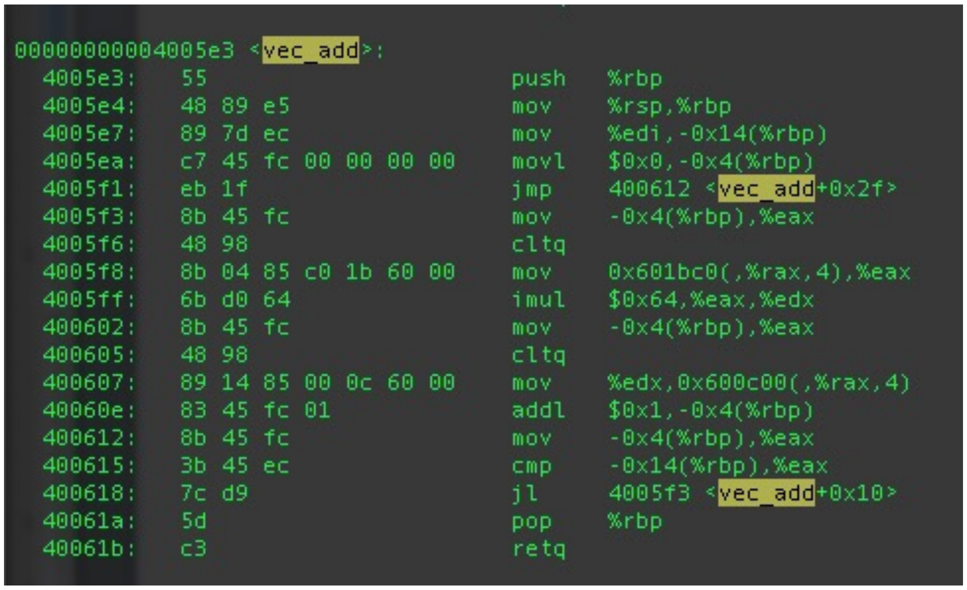
	Vec_add		Vec_mult		Speedup	
Array size	**SISD**	**SIMD**	**SISD**	**SIMD**	**Vec_add**	**Vec_mult**
0	84	21	81	42	4	1.928571429
8	1383	1263	198	243	1.095011876	0.814814815
16	228	57	219	69	4	3.173913043
24	390	63	426	69	6.19047619	6.173913043
32	369	69	444	87	5.347826087	5.103448276
40	270	81	582	96	3.333333333	6.0625
48	285	84	633	111	3.392857143	5.702702703
56	495	93	738	114	5.322580645	6.473684211
64	372	99	372	123	3.757575758	3.024390244
72	402	108	627	135	3.722222222	4.644444444
80	450	111	465	855	4.054054054	0.543859649
88	480	114	711	147	4.210526316	4.836734694
96	585	135	549	159	4.333333333	3.452830189
104	576	132	810	174	4.363636364	4.655172414
112	597	144	639	183	4.145833333	3.491803279
120	801	165	882	189	4.854545455	4.666666667
128	672	156	726	204	4.307692308	3.558823529
136	732	159	969	210	4.603773585	4.614285714
144	756	159	816	213	4.754716981	3.830985915
152	810	5160	1062	219	0.156976744	4.849315068
160	846	180	1095	222	4.7	4.932432432
168	918	1155	957	264	0.794805195	3.625
176	963	189	996	246	5.095238095	4.048780488
184	1014	192	1239	1572	5.28125	0.788167939
192	1032	201	1278	258	5.134328358	4.953488372
200	1236	204	1320	276	6.058823529	4.782608696
208	1083	210	1371	282	5.157142857	4.861702128
216	1122	219	1452	294	5.123287671	4.93877551
224	1203	225	1560	306	5.346666667	5.098039216
232	1251	411	1617	309	3.04379562	5.233009709
240	1224	234	1749	318	5.230769231	5.5
248	1560	246	1845	327	6.341463415	5.642201835
256	1320	255	1938	336	5.176470588	5.767857143
264	6528	255	1893	342	25.6	5.535087719
272	1437	261	1983	357	5.505747126	5.554621849
280	2616	276	2055	357	9.47826087	5.756302521
288	1491	279	2034	378	5.344086022	5.380952381

296	1800	321	3279	384	5.607476636	8.5390625
304	1566	291	2103	393	5.381443299	5.351145038
312	1608	300	2010	399	5.36	5.037593985
320	1920	306	2235	414	6.274509804	5.398550725
328	1974	483	2253	411	4.086956522	5.481751825
336	1728	318	2322	432	5.433962264	5.375
344	1773	327	2370	435	5.422018349	5.448275862
352	1800	336	2379	447	5.357142857	5.322147651
360	1842	339	2217	447	5.433628319	4.959731544
368	1857	348	2304	462	5.336206897	4.987012987
376	2103	363	2343	468	5.79338843	5.006410256
384	1962	357	2391	480	5.495798319	4.98125
392	2004	369	2424	492	5.430894309	4.926829268
400	2019	363	2472	498	5.561983471	4.963855422
408	2106	387	2523	501	5.441860465	5.035928144
416	2088	384	2547	522	5.4375	4.879310345
424	2211	396	2805	516	5.583333333	5.436046512
432	2202	396	2811	531	5.560606061	5.293785311
440	2241	417	2685	546	5.374100719	4.917582418
448	2223	405	2736	564	5.488888889	4.85106383
456	2343	420	2778	558	5.578571429	4.978494624
464	2331	426	2778	570	5.471830986	4.873684211
472	2466	441	2826	573	5.591836735	4.931937173
480	2403	429	2898	600	5.601398601	4.83
488	2571	465	2910	600	5.529032258	4.85
496	2520	450	3000	603	5.6	4.975124378
504	2592	465	3045	618	5.574193548	4.927184466
512	2577	456	3069	624	5.651315789	4.918269231

Problem 2) Graph:



Problem 3 a)
SISD:

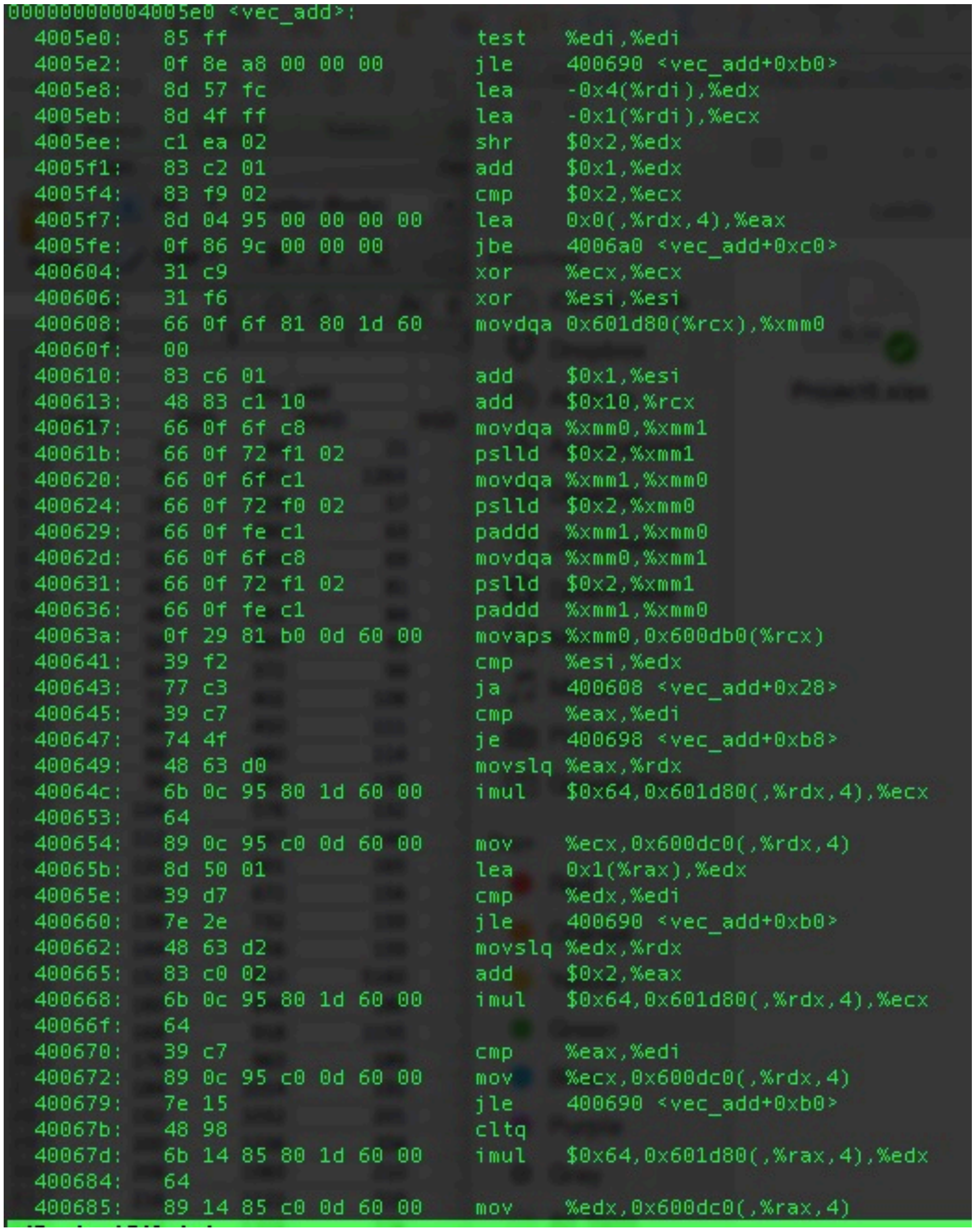


```
00000000004005e3 <vec_add>:
  4005e3:    55                      push   %rbp
  4005e4:    48 89 e5                mov    %rsp,%rbp
  4005e7:    89 7d ec                mov    %edi,-0x14(%rbp)
  4005ea:    c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  4005f1:    eb 1f                   jmp    400612 <vec_add+0x2f>
  4005f3:    8b 45 fc                mov    -0x4(%rbp),%eax
  4005f6:    48 98                   cltq
  4005f8:    8b 04 85 c0 1b 60 00    mov    0x601bc0(,%rax,4),%eax
  4005ff:    6b d0 64                imul   $0x64,%eax,%edx
  400602:    8b 45 fc                mov    -0x4(%rbp),%eax
  400605:    48 98                   cltq
  400607:    89 14 85 00 0c 60 00    mov    %edx,0x600c00(,%rax,4)
  40060e:    83 45 fc 01             addl   $0x1,-0x4(%rbp)
  400612:    8b 45 fc                mov    -0x4(%rbp),%eax
  400615:    3b 45 ec                cmp    -0x14(%rbp),%eax
  400618:    7c d9                   jl     4005f3 <vec_add+0x10>
  40061a:    5d                      pop    %rbp
  40061b:    c3                      retq
```

1) imul $0x64, %eax, %edx is the place where c[i]*100 takes place. Explaining the where the add takes place.

2) With snapshot or code explaining vec_mult. Similar to the above code imul %eax, %edx is the code where the multiplication actually takes place.

SIMD:



```
00000000004005e0 <vec_add>:
  4005e0:   85 ff                   test   %edi,%edi
  4005e2:   0f 8e a8 00 00 00       jle    400690 <vec_add+0xb0>
  4005e8:   8d 57 fc                lea    -0x4(%rdi),%edx
  4005eb:   8d 4f ff                lea    -0x1(%rdi),%ecx
  4005ee:   c1 ea 02                shr    $0x2,%edx
  4005f1:   83 c2 01                add    $0x1,%edx
  4005f4:   83 f9 02                cmp    $0x2,%ecx
  4005f7:   8d 04 95 00 00 00 00    lea    0x0(,%rdx,4),%eax
  4005fe:   0f 86 9c 00 00 00       jbe    4006a0 <vec_add+0xc0>
  400604:   31 c9                   xor    %ecx,%ecx
  400606:   31 f6                   xor    %esi,%esi
  400608:   66 0f 6f 81 80 1d 60    movdqa 0x601d80(%rcx),%xmm0
  40060f:   00
  400610:   83 c6 01                add    $0x1,%esi
  400613:   48 83 c1 10             add    $0x10,%rcx
  400617:   66 0f 6f c8             movdqa %xmm0,%xmm1
  40061b:   66 0f 72 f1 02          pslld  $0x2,%xmm1
  400620:   66 0f 6f c1             movdqa %xmm1,%xmm0
  400624:   66 0f 72 f0 02          pslld  $0x2,%xmm0
  400629:   66 0f fe c1             paddd  %xmm1,%xmm0
  40062d:   66 0f 6f c8             movdqa %xmm0,%xmm1
  400631:   66 0f 72 f1 02          pslld  $0x2,%xmm1
  400636:   66 0f fe c1             paddd  %xmm1,%xmm0
  40063a:   0f 29 81 b0 0d 60 00    movaps %xmm0,0x600db0(%rcx)
  400641:   39 f2                   cmp    %esi,%edx
  400643:   77 c3                   ja     400608 <vec_add+0x28>
  400645:   39 c7                   cmp    %eax,%edi
  400647:   74 4f                   je     400698 <vec_add+0xb8>
  400649:   48 63 d0                movslq %eax,%rdx
  40064c:   6b 0c 95 80 1d 60 00    imul   $0x64,0x601d80(,%rdx,4),%ecx
  400653:   64
  400654:   89 0c 95 c0 0d 60 00    mov    %ecx,0x600dc0(,%rdx,4)
  40065b:   8d 50 01                lea    0x1(%rax),%edx
  40065e:   39 d7                   cmp    %edx,%edi
  400660:   7e 2e                   jle    400690 <vec_add+0xb0>
  400662:   48 63 d2                movslq %edx,%rdx
  400665:   83 c0 02                add    $0x2,%eax
  400668:   6b 0c 95 80 1d 60 00    imul   $0x64,0x601d80(,%rdx,4),%ecx
  40066f:   64
  400670:   39 c7                   cmp    %eax,%edi
  400672:   89 0c 95 c0 0d 60 00    mov    %ecx,0x600dc0(,%rdx,4)
  400679:   7e 15                   jle    400690 <vec_add+0xb0>
  40067b:   48 98                   cltq
  40067d:   6b 14 85 80 1d 60 00    imul   $0x64,0x601d80(,%rax,4),%edx
  400684:   64
  400685:   89 14 85 c0 0d 60 00    mov    %edx,0x600dc0(,%rax,4)
```

1) Identifying the equivalent code here. imul $0x64,0x601d80(,%rax,4),%edx is the place where c[i]*100 takes place and paddd %xmm1,%xmm0 is the place where addition takes place. Brief explanation of xmm0 and xmm1 registers referring to the lectures.
2) Similarly for vec_mult.

```
pmuludq %xmm2,%xmm1
```

These lines give the crux. pmuldq gives the multiplied value of the vectors.

XMM0-XMM8 are the vector registers specially used for vector operations. They are 128 bits in length and can stream 4 words at a time. So this the reason when the vector length increases the time taken by vectorized code in less as it uses vector registers.