Computer Architecture Lab6 CS322 Report

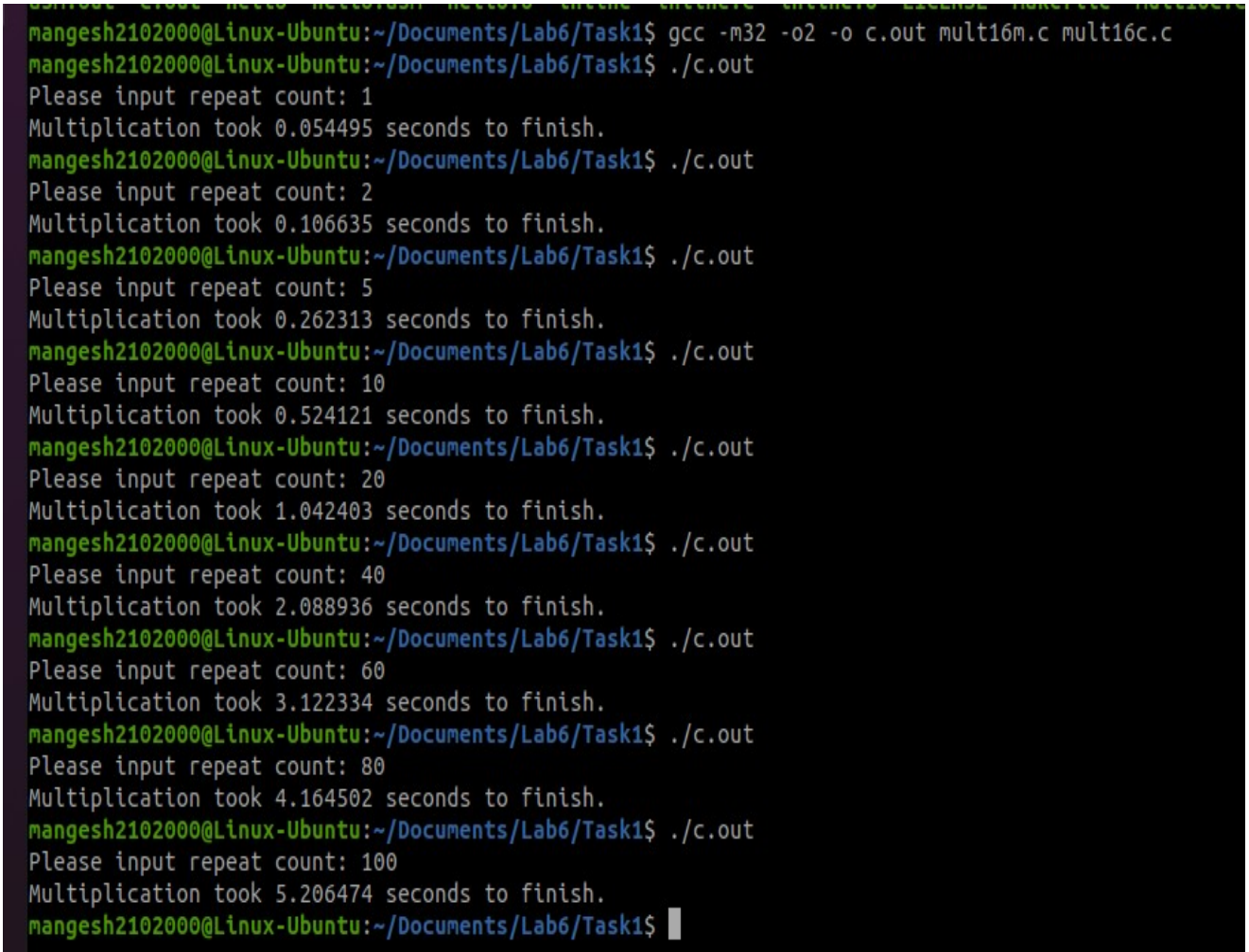
Name : **Chandrawanshi Mangesh Shivaji**
Roll No : **1801CS16**
Date : 23/10/2020

Task 1a : Comparison of C and Assembly version of Multiplication Functions

gcc -m32 -o2 -o c.out mult16m.c mult16c.c
./c.out

C version

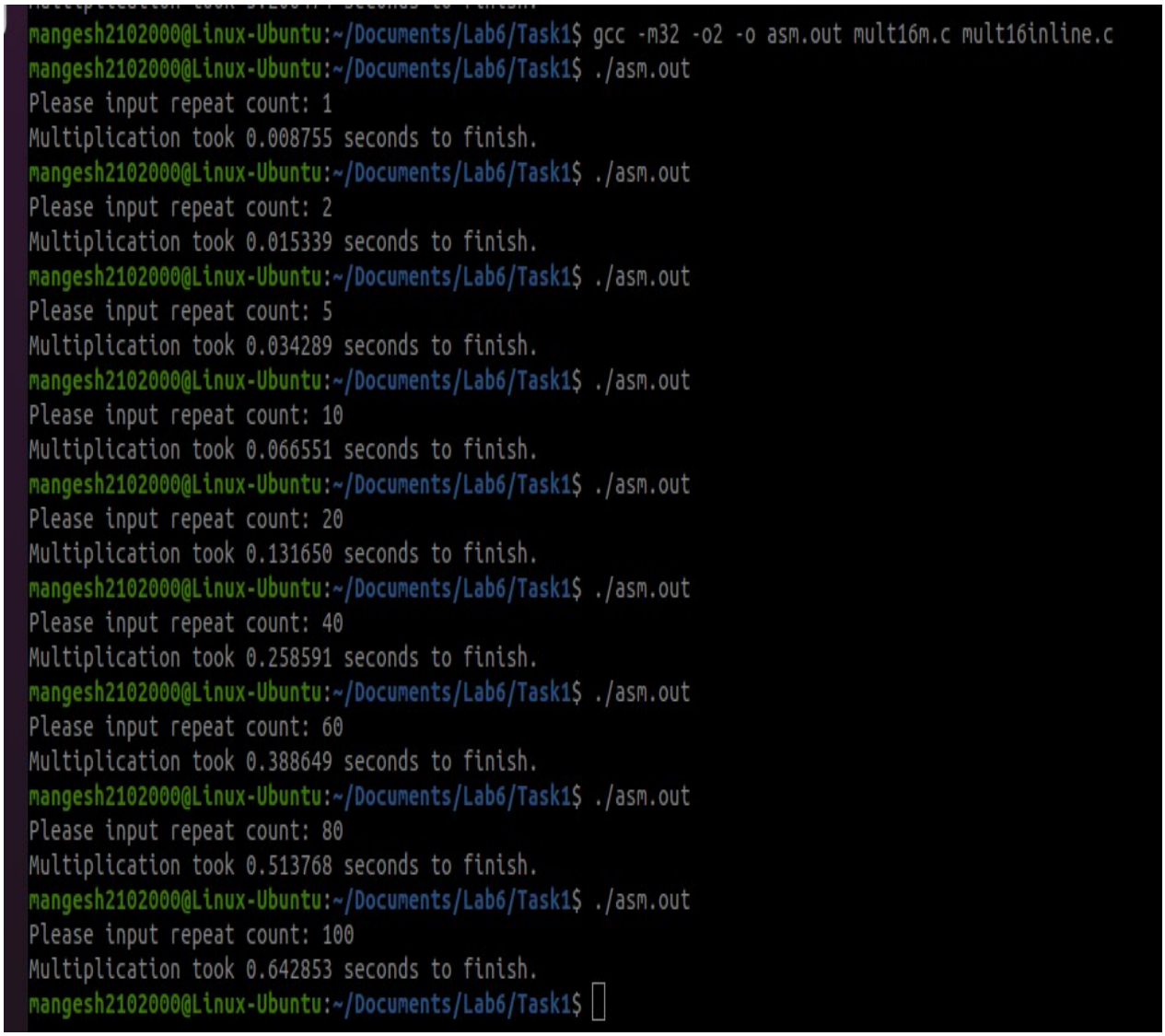
Number of Calls (in millions)	Time (in seconds)
1	0.054495
2	0.106635
5	0.262313
10	0.524121
20	1.042403
40	2.088936
60	3.122334
80	4.164502
100	5.206474



gcc -m32 -o2 -o asm.out mult16m.c mult16inline.c
./asm.out

AL version

Number of Calls (in millions)	**Time (in seconds)**
1	0.008755
2	0.015339
5	0.034289
10	0.066551
20	0.131650
40	0.258591
60	0.388649
80	0.513768
100	0.642853



Graph 1 a

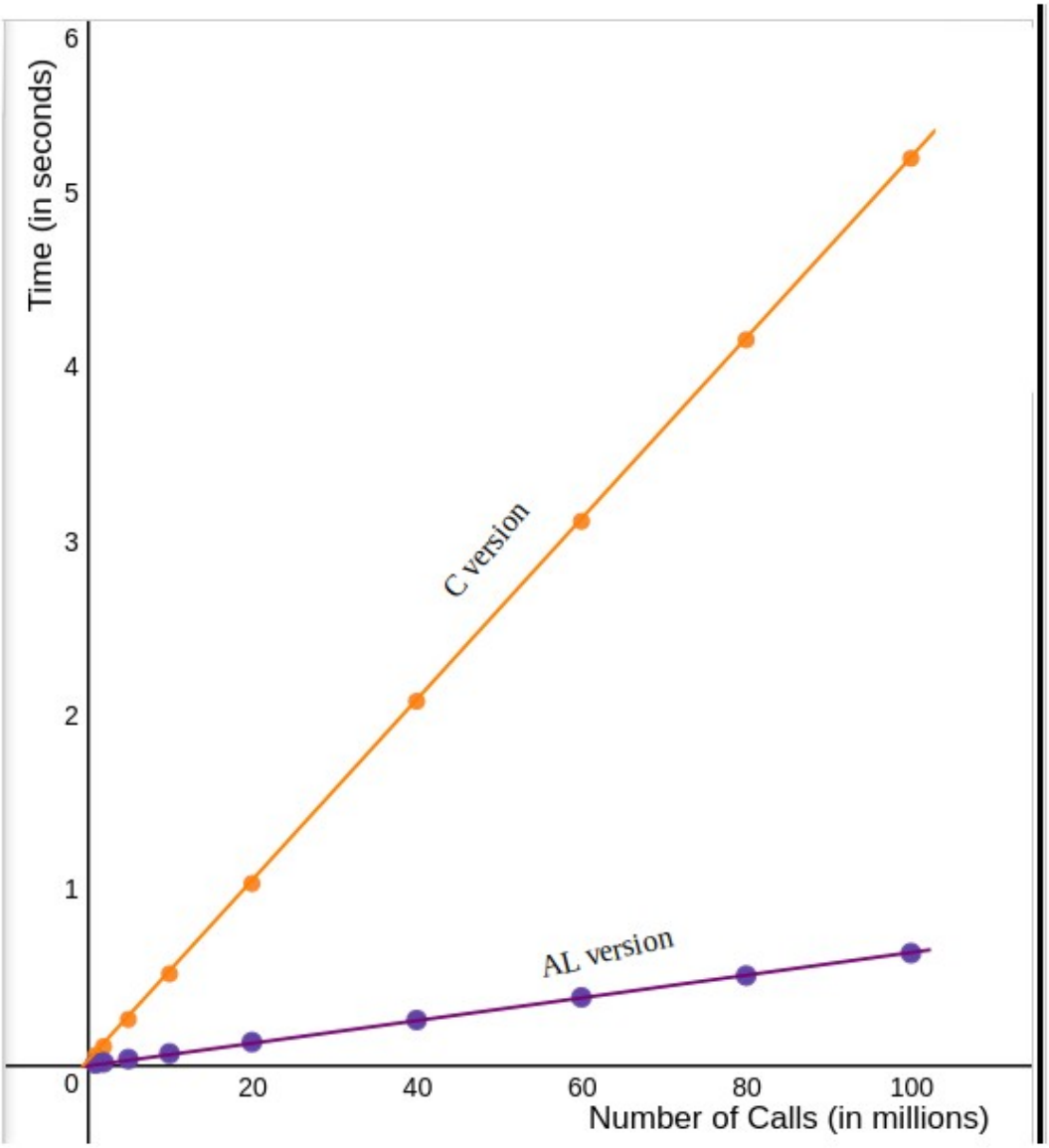


Fig. Comparision_Graph_C_and_AL_mult

So, from above graph we can conclude that assembly works faster than its equivalent C implementation. Here, both curves are linear in nature. But slope for C is greater is than corresponding Assembly implementation.

Task 1 b : Comparison of C and Assembly version of Factorial Functions

```
gcc -std=c99 L6_factorial.o L6_factorial_main_t_c.c
./a.out
```

C version

Number(upto which factorials are calculated)	Time (in seconds)
1	0.000003
2	0.000004
5	0.000005
10	0.000005
25	0.000010
50	0.000023
100	0.000091
500	0.001515
1000	0.005546
5000	0.063451
10000	0.235433
20000	0.954791
30000	2.083967
40000	3.731651
50000	5.835595
60000	8.288919
70000	11.385861
80000	14.715557
90000	18.967658
100000	23.573817

```
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ gcc -std=c99 L6_factorial.o L6_factorial_main_t_c.c
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 1
Factorial took 0.000003 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 2
Factorial took 0.000004 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 5
Factorial took 0.000005 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 10
Factorial took 0.000005 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 25
Factorial took 0.000010 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 50
Factorial took 0.000023 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 100
Factorial took 0.000091 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 500
Factorial took 0.001515 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 1000
Factorial took 0.005546 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 5000
Factorial took 0.063451 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 10000
Factorial took 0.235433 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 20000
Factorial took 0.954797 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 30000
Factorial took 2.083967 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 40000
Factorial took 3.731651 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 50000
Factorial took 5.835595 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 60000
Factorial took 8.288919 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 70000
Factorial took 11.385861 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 80000
Factorial took 14.715557 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 90000
Factorial took 18.967658 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 100000
Factorial took 23.573817 second
```

```
nasm -f elf64 L6_factorial.asm
gcc -std=c99 L6_factorial.o L6_factorial_main_t.c
./a.out
```

AL version

Number(upto which factorials are calculated)	Time (in seconds)
1	0.000004
2	0.000005
5	0.000005
10	0.000005
25	0.000007
50	0.000013
100	0.000039
500	0.000999
1000	0.003493
5000	0.036828
10000	0.135437
20000	0.533520
30000	1.190139
40000	2.114971
50000	3.303128
60000	4.762004
70000	6.501823
80000	8.520762
90000	10.740420
100000	13.278020

```
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ nasm -f elf64 L6_factorial.asm
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ gcc -std=c99 L6_factorial.o L6_factorial_main_t.c
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 1
Factorial took 0.000004 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 2
Factorial took 0.000005 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 5
Factorial took 0.000005 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 10
Factorial took 0.000005 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 25
Factorial took 0.000007 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 50
Factorial took 0.000013 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 100
Factorial took 0.000039 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 500
Factorial took 0.000999 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 1000
Factorial took 0.003493 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 5000
Factorial took 0.036828 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 10000
Factorial took 0.135437 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 20000
Factorial took 0.533520 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 30000
Factorial took 1.190139 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 40000
Factorial took 2.114971 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 50000
Factorial took 3.303128 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 60000
Factorial took 4.762004 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 70000
Factorial took 6.501823 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 80000
Factorial took 8.520762 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 90000
Factorial took 10.740420 second
mangesh2102000@Linux-Ubuntu:~/Documents/Lab6/Lab6$ ./a.out
Enter input number : 100000
Factorial took 13.278020 second
```

Graph 1 b

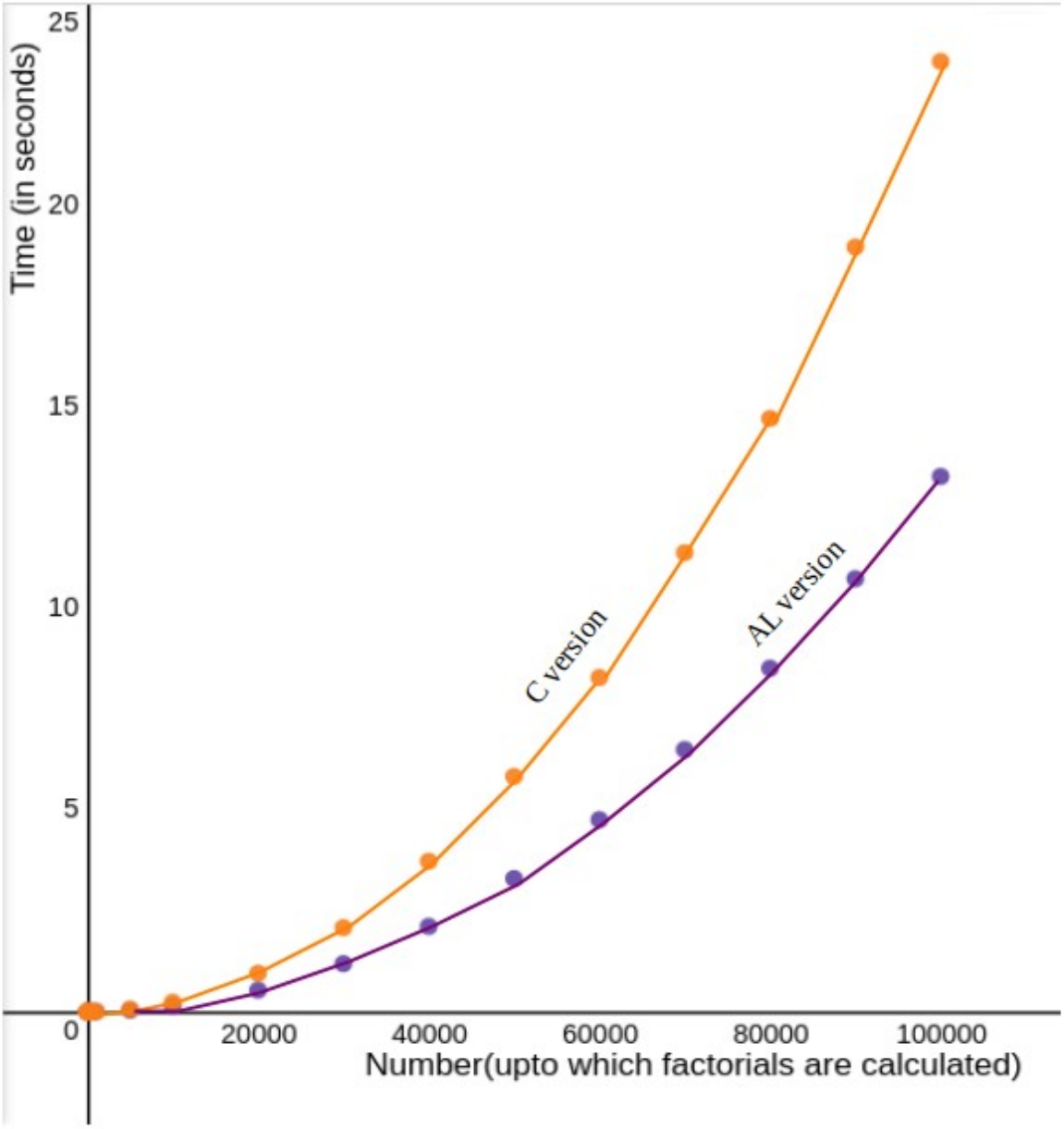


Fig. Comparision_Graph_C_and_AL_fact

So, from above graph we can conclude that assembly works faster than its equivalent C implementation for factorial function. Here, both curves are exponential in nature, But the constant factor for Assembly is lesser than its C counterpart.

From 1a and 1b, we can infer that for speed optimization it will be better if we write important functions in Assembly than any other high level language like C. Though it comes with a difficult implementation and also reduces code readability.

Task 2 : A 64-bit NASM program and a C program for implementing x^y functionality

Algorithm :

if y < 0, tell user that negative exponent is not allowed
if y == 0, ans is always 1
if y > 0, First store x in some variable (let us say in ans)
Then loop (y-1) times, in each iteration multiply ans by x
Now, we have x^y stored in ans

Command for running 64-bit NASM Program :

$ nasm -felf64 lab6.asm && gcc -o lab6 lab6.o –static
$ ./lab6 num1 num2
where num1 is base and num2 is exponent

```
mangesh2102000@Linux-Ubuntu:~/Documents$ nasm -felf64 lab6.asm && gcc -o lab6 lab6.o --static
mangesh2102000@Linux-Ubuntu:~/Documents$ ./lab6 5 3
125
mangesh2102000@Linux-Ubuntu:~/Documents$ ./lab6 25 0
1
mangesh2102000@Linux-Ubuntu:~/Documents$ ./lab6 35 -2
The exponent may not be negative
```

Command for running C Program :

$ gcc lab6.c
$ ./a.out
after this user will be prompted for input accordingly

```
mangesh2102000@Linux-Ubuntu: ~/Documents 80x24
mangesh2102000@Linux-Ubuntu:~/Documents$ gcc lab6.c
mangesh2102000@Linux-Ubuntu:~/Documents$ ./a.out
Enter Number : 5
Enter Exponent : 3
Answer is : 125
mangesh2102000@Linux-Ubuntu:~/Documents$ ./a.out
Enter Number : 25
Enter Exponent : 0
Answer is : 1
mangesh2102000@Linux-Ubuntu:~/Documents$ ./a.out
Enter Number : 35
Enter Exponent : -2
The exponent may not be negative
```