

## مرحله اول:

بدون بهینه سازی:

```
→ lab1 ./a.out
Serial Timings for 50000 iterations

Time Elapsed: 7.434777 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.372481 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.413630 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.424001 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.418277 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.424945 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.461292 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.383861 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.392344 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.394958 Secs, Total = 30.656747, Check Sum = 50000
d
→ lab1 █
```

با بهینه سازی:

```
→ lab1 gcc Lab-1-1.cpp -fopenmp -lm -O2
→ lab1 ./a.out
Serial Timings for 50000 iterations

Time Elapsed: 3.300523 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.258078 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.262199 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.261960 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.262722 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.257854 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.265194 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.265169 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.260344 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 3.259613 Secs, Total = 30.656747, Check Sum = 50000
s
→ lab1 §█
```

## سوال ۱:

اکنرا برنامه درون حلقه‌ی به اندازه‌ی **VERYBIG** در حال اجرا می‌باشد و خطوط ۳۳ و ۳۷ که درون

حلقه درونی به طول  $\mathfrak{z}$  اجرا می‌شوند در اکثر برنامه در حال اجرا هستند زیرا این خطوط

$O(VERYBIG)$  اجرا می‌شوند اما بقیه خطوط حداکثر  $O(VERYBIG^2)$  اجرا می‌شوند.

بدنه حلقه‌ی **VERYBIG** فقط دو متغیر مشترک دارد که آنها `total` و `sumy` و `sumx` و `k` می‌باشند

که چون فقط این متغیرها با یک مقدار دیگر جمع می‌شوند در هر اجرای بدنه و ترتیب این

جمع شدن مهم نیست و به هم وابسته نیستند، بدنه‌ی حلقه **VERYBIG** کاملاً قابل موازی سازی

می‌باشد.

همچنین حلقه‌های خطوط ۳۲ و ۳۶ مانند حلقه VERYBIG می‌توانند تبدیل به parallel loop شوند.

## سوال : ۲

برای اجرای چندین بار آزمایش و اندازه گیری چندین بار پارامترها که خطای احتمالی از اندازه گیری حذف شود، تنها مقداری که در تمام اجراهای متفاوت است مقدار Time elapsed می‌باشد، دلیل این اتفاق این است که زمان اجرا به وضعیت کامپیوتر، برنامه‌های دیگر اجرا شده و عوامل محیطی دیگر وابسته است.

## سوال : ۳

در حالت DEBUG توانایی debug وجود دارد، برای مثال می‌توان breakpoint بر روی خطوط مختلف گذاشت و یا روند حرکت برنامه را مرحله مشاهده کرد، همچنین وضعیت پردازنده و حافظه نیست به صورت دلخواه قابل ملاحظه هستند، برای بدست آوردن این امکانات، کامپایلر در حالت DEBUG تکه کدهایی به بخش‌های مختلف برنامه اضافه می‌کند و همچنین بعضی بخش‌ها از برنامه را تفسیر (interpret) می‌کند و این موارد باعث کند شدن اجرای برنامه نسبت به کامپایل ساده و نهایی در حالت RELEASE می‌باشد.

#### سوال ۴:

می‌توان درون حلقه VERYBIG می‌توان task parallelism بین محاسبه‌ی sumx و sumy انجام داد.

همچنین برای هر سه حلقه (حلقه VERYBIG، دو حلقه‌ی ز تایی درون حلقه VERYBIG) می‌توان divide and conquer انجام داد.

## مرحله ۲:

```
→ lab1 ./a.out
OpenMp is good :)
Serial Timings for 50000 iterations
^C
→ lab1 []
```

چک کردن وجود openmp

```
→ lab1 ./a.out
OpenMp is good :)
Serial Timings for 50000 iterations
Time Elapsed: 55.905303 Secs, Total = inf, Check Sum = 44303
Time Elapsed: 66.154218 Secs, Total = inf, Check Sum = 44384
^C
→ lab1 []
```

حالت مسابقه

دلیل این مشکل به وجود آمدن race condition می باشد.

```
→ lab1 ./a.out
OpenMp is good :)
Serial Timings for 50000 iterations
Time Elapsed: 1.931332 Secs, Total = 30.628844, Check Sum = 49915
Time Elapsed: 1.862462 Secs, Total = 30.633554, Check Sum = 49917
Time Elapsed: 1.860233 Secs, Total = 30.616026, Check Sum = 49893
Time Elapsed: 1.884076 Secs, Total = 30.617522, Check Sum = 49914
Time Elapsed: 1.866981 Secs, Total = 30.610758, Check Sum = 49928
Time Elapsed: 1.740801 Secs, Total = 30.627878, Check Sum = 49900
Time Elapsed: 1.736156 Secs, Total = 30.623503, Check Sum = 49896
Time Elapsed: 1.825911 Secs, Total = 30.629247, Check Sum = 49902
Time Elapsed: 2.008650 Secs, Total = 30.639670, Check Sum = 49943
Time Elapsed: 1.929498 Secs, Total = 30.636557, Check Sum = 49932
```

حالت درست کردن private ها ولی بدون critical section

```
→ lab1 gcc Lab-1-1.cpp -fopenmp -lm
→ lab1 ./a.out
OpenMp is good ;)
Serial Timings for 50000 iterations

Time Elapsed: 1.928027 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.871777 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.870221 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.868658 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.870784 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.870275 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.870492 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.874153 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.872066 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.874101 Secs, Total = 30.656747, Check Sum = 50000
```

بعد از وارد کردن `critical` مقدار `sum` درست می شود.

```
→ lab1 ./a.out
OpenMp is good ;)
Serial Timings for 50000 iterations

Time Elapsed: 1.892185 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.919961 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.857724 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.759912 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.808521 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.829484 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.902719 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.845759 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.888369 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.881452 Secs, Total = 30.656747, Check Sum = 50000
d
→ lab1 □
```

اجرا با استفاده از `reduction` که سریع تر است.

```
→ lab1 ./a.out
OpenMp is good ;)
Serial Timings for 50000 iterations

Time Elapsed: 1.904946 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.865419 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.856059 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.860303 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.860013 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.861598 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.769626 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.857157 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.847912 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.866207 Secs, Total = 30.656747, Check Sum = 50000
```

در حالت ذخیره `sum` و `total` در یک آرایه و در نهایت جمع زدن آرایه.

## سوال ۱:

در سیستم ۴ هسته ای / ۸ نخ مورد استفاده، برنامه تعداد ۸ نخ ایجاد کرد در مجموع (با احتساب نخ اصلی) که نشانه آن است که رابطه مستقیم دارند با هم.

### سوال :۲

خیر زیرا atomic محدود به چند عملیات است و تقسیم را شامل نمیشود که در محاسبه total به آن نیاز داریم.

### سوال :۳

در حالت عادی، تفاوت زمانی معنی داری مشاهده نشد (1.28 برای حالت critical، 1.29 برای reduction). سپس تعداد نخ ها را به 4 تغییر دادیم. در حالت critical به 2.26 رسیدیم، و در حالت reduction به 2.23 که باز هم تفاوت معنی داری پیدا نشد. سپس در حالت ۸ نخی، تعداد ۵۰۰۰، ۵۰۰۰۰ و ۵۰۰۰۰۰ تست شد که در تعداد ۵۰۰۰ تایی، در هر دو حالت ۰.۰۲۵ ثانیه، در تعداد ۵۰۰۰۰ هر دو به ۱۰.۲۸ ثانیه رسیدند، و در تعداد ۵۰۰۰۰۰ هر دو حالت critical به ۱۲۳ ثانیه و reduction به زمان ۱۲۵ ثانیه رسیدیم که باز هم عملا هیچ تفاوتی را نشان نمیدهد.