

## مقدمه

در این آزمایش شما باید یک برنامه سریال جمع دو ماتریس را با تنظیمات مناسب در محیط Visual Studio کامپایل و اجرا کنید. سپس به کمک رهنمودهای OpenMP برنامه را موازی و از صحت عملکرد آن اطمینان حاصل کنید. در نهایت تسریع به دست آمده محاسبه می‌شود.

## آزمایش

### ❖ مرحله اول: ساخت پروژه در ویژوال استودیو، تنظیم پروژه، کامپایل و اجرای کد

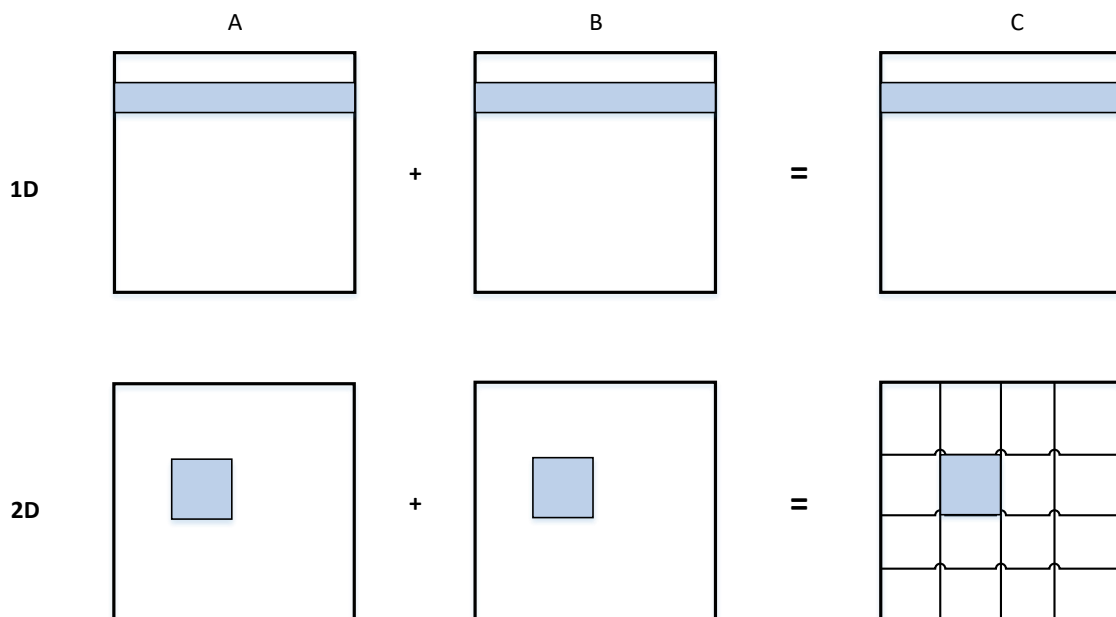
- کد برنامه درون فایل zip که در اختیارتان قرار گرفته موجود است. نام فایل matadd.cpp است.
- به حالت کامپایل (Debug/Release) توجه کنید.

### ❖ مرحله دوم: فعال‌سازی OpenMP و موازی‌سازی برنامه

برنامه را به کمک OpenMP به سه صورت زیر موازی کنید و خروجی آن را با خروجی کد سریال مقایسه کنید:

1. یک بعدی سطری
2. یک بعدی ستونی
3. دوبعدی

قسمت حاشور زده شده حجم کاری است که یک نخ انجام می‌دهد (شکل اول روش یک بعدی سطری را نشان می‌دهد). توجه داشته باشید که صرفاً موازی‌سازی عمل جمع مد نظر است. برای موازی‌سازی دوبعدی می‌توانید از nested parallelism استفاده کنید. توجه داشته باشید باید این ویژگی فعال شود.



<sup>1</sup> Directive

## ❖ مرحله سوم: اندازه‌گیری

برای سادگی ماتریس‌های A و B را مربعی فرض کنید و پس از پیاده‌سازی، زمان عمل جمع را با تابع مناسب اندازه گرفته و برای هر سه روش موازی‌سازی جدول ذیل را پر کرده و گزارش کنید. تکرار هر اجرا و میانگین گرفتن از زمان‌های اجرا به افزایش دقت اندازه‌گیری کمک می‌کند. ابعاد هر ماتریس ورودی را به گونه‌ای بگیرید که حجم آن برابر مقدار ورودی خواسته شده باشد. هر  $int$  را چهار بایت فرض کنید.

## نتایج روش اول

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1					-
4					
8					
16					

## نتایج روش دوم

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1					-
4					
8					
16					

## نتایج روش سوم

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1					-
4					
8					
16					

## ❖ نکته

پس از آنکه از صحت خروجی برنامه اطمینان حاصل کردید، برای ماتریس‌های بزرگ و اندازه‌گیری زمان می‌توانید تابع چاپ ماتریس‌ها را فراخوانی نکنید.

## ❖ گزارش

پس از اتمام آزمایش به سوال‌های زیر پاسخ دهید:

1. کدام یک از دو روش تجزیه یک بعدی کندتر اجرا می‌شود؟ چرا؟
2. افزایش اندازه ماتریس ورودی چه تاثیری بر میزان تسريع دارد؟
3. بیشترین میزان تسريع برای هر روش مربوط به چه تعداد نخ است؟ این تعداد چه ارتباطی با ساختار پردازنده شما دارد؟
4. از نظر شما روش ایستا برای تقسیم کار بین نخ‌ها در این برنامه مناسب‌تر است یا روش پویا؟ چرا؟
5. به طور کلی تجزیه دو بعدی و یک بعدی چه مزایایی نسبت به هم دارند؟ به طور خاص در این برنامه کدام روش بهتر عمل می‌کند؟

در نظر داشته باشید که لزومی بر اتمام آزمایش در جلسه آزمایشگاه نیست و امکان ارسال نتایج بدون کسر نمره تا آخر هفته وجود دارد.

## بخش امتیازی

پاسخ این قسمت را می توانید پس از آزمایشگاه در قسمتی که برای ارسال کدها و نیز گزارشکارها باز شده است ارسال نمایید.

علاوه بر روش های بالا روش زیر نیز می تواند به ما در بهینگی عملکرد برنامه موازی داده شده کمک کند.

۴. تجميع حلقه های for

یک مشکل برای روش یک بعدی سطری این است که تمام عناصر یک سطر توسط یک نخ پردازش می شوند. می دانیم ماتریس های دوبعدی در حافظه به صورت یک آرایه یک بعدی ذخیره می شوند بنابراین می توانیم با تجميع دو حلقه for موجود برای پیمایش آرایه دو بعدی به یک حلقه عمل جمع ماتریس ها را به صورت بهتر و نزدیک تری به نحوه ی پیاده سازی آن توسط سخت افزار انجام دهیم. هم چنین نسبت به روش دو بعدی سربار کمتری را نیز به سیستم تحمیل کنیم. این کار را به صورت زیر می توانیم انجام دهیم:

```
void add_collapse(DataSet dataSet, int chunkSize) {
#pragma omp parallel for schedule(static, chunkSize)
    for (int ij = 0; ij < dataSet.n * dataSet.m; ij++) {
        dataSet.C[ij] = dataSet.A[ij] + dataSet.B[ij];
    }
}
```

در این روش به اصطلاح دو حلقه for موجود را در هم collapse کردیم. دقت کنید که پس از collapse کردن همچنان می توانیم i را از طریق انجام عملیات  $ij / \text{dataSet.m}$  و j را از طریق  $ij \% \text{dataSet.m}$  به دست آوریم. این عملیات در openMP های ورژن ۳ به بالا اضافه شد تا بتوان بدون تغییر ظاهر کد نیز از روش collapse کردن استفاده نمود:

```
void add_collapse(DataSet dataSet, int chunkSize) {
#pragma omp parallel for collapse(2) schedule(static, chunkSize)
    for (int i = 0; i < dataSet.n; i++) {
        for (int j = 0; j < dataSet.m; j++) {
            dataSet.C[i * dataSet.m + j] =
                dataSet.A[i * dataSet.m + j] + dataSet.B[i * dataSet.m + j];
        }
    }
}
```

با پیاده سازی روش بالا جدول زیر را تکمیل کنید و زمان ها را با سایر روش ها مقایسه نمایید.

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1					-
4					
8					
16					

میزان تسریع این روش را نسبت به سایر روش ها تحلیل نمایید.