به نام خدا

برنامه‌نویسی چندهسته‌ای

دستور کار آزمایشگاه 2

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

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 رهنمود[1]های OpenMP برنامه را موازی و از صحت عملکرد آن اطمینان حاصل کنید. در نهایت تسریع به‌دست‌آمده محاسبه می‌شود.

آزمایش

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

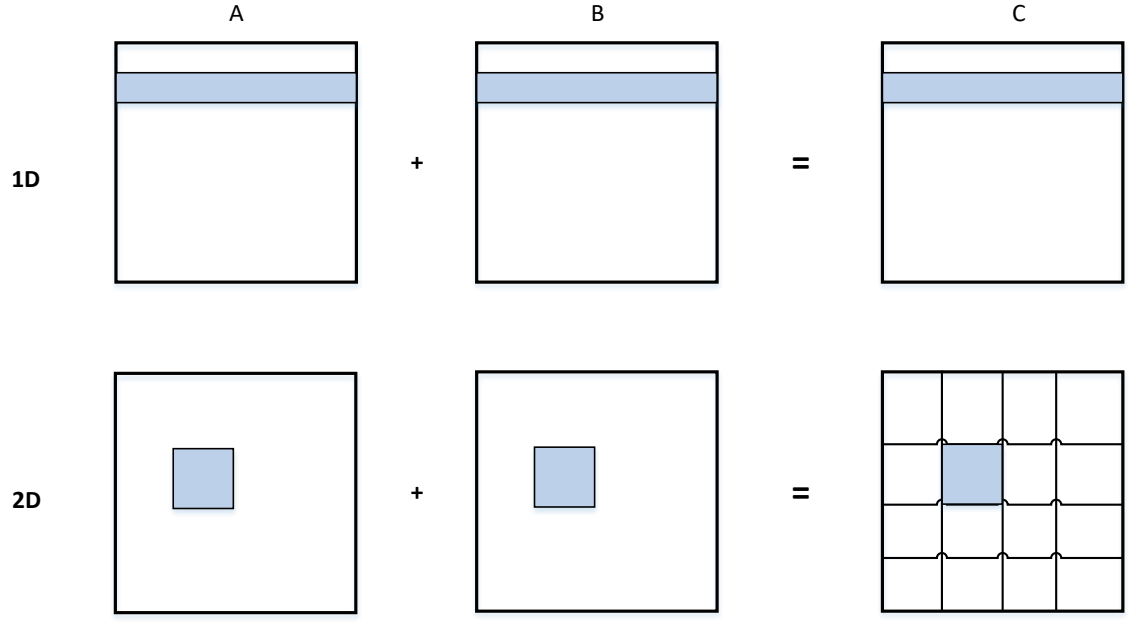
- کد برنامه درون فایل zip که در اختیارتان قرار گرفته موجود است. نام فایل matadd.cpp است.
- به حالت کامپایل (Debug/Release) توجه کنید.

❖ **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 هاشور خورده 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 موازی سازی دوبعدی می‌توانید از nested parallelism استفاده کنید. توجه داشته باشید باید این ویژگی فعال شود.



❖ **مرحله سوم: اندازه‌گیری**

[1] Directive

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				میانگین تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	0.000328	0.000309	0.31890	0.29043	-
4	0.000306	0.000339	0.29205	0.28987	1.065
8	0.000286	0.000290	0.29199	0.28872	1.077
16	0.000300	0.000315	0.29230	0.28576	1.045

نتایج روش دوم

تعداد نخ‌ها	اندازه هر ماتریس ورودی				میانگین تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	0.000613	0.000685	0.35149	0.30338	-
4	0.000640	0.000677	0.34993	0.29556	1.000
8	0.002655	0.002653	0.34419	0.29613	0.633
16	0.002696	0.002947	0.34590	0.29374	0.627

نتایج روش سوم

تعداد نخ‌ها	اندازه هر ماتریس ورودی				میانگین تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	0.000767	0.000546	0.439824	0.435928	-
4	0.019690	0.018630	0.392011	0.347871	0.610
8	0.072224	0.042422	0.584253	0.436889	0.443
16	0.299609	0.087670	1.197199	0.742058	0.241

❖ **گزارش**

پس از اتمام آزمایش به سوال‌های زیر پاسخ دهید:

1. کدام یک از دو روش تجزیه یک بعدی کندتر اجرا می‌شود؟ چرا؟

روش ستونی. به این دلیل که ماتریس به صورت سطری در حافظه ذخیره شده و با استفاده‌ی سیستم از تکنیک Caching و Spatial Locality دسترسی به خانه‌های نزدیک به هم سریع‌تر می‌شود.

2. افزایش اندازه ماتریس ورودی چه تاثیری بر میزان تسریع دارد؟

از نتایج گرفته‌شده در این آزمایش نمی‌توان نتیجه‌ی کلی‌ای گرفت. در تفکیک سطری به وضوح با افزایش سایز، تسریع کم‌تر می‌شود اما در روش‌های دیگر در سایز کوچک‌تر تسریع منفی‌تری داریم و در سایز بزرگ‌تر می‌توان تسریع خوبی را مشاهده کرد.

3. بیشترین میزان تسریع برای هر روش مربوط به چه تعداد نخ است؟ این تعداد چه ارتباطی با ساختار پردازنده شما دارد؟

روش اول: ۸ نخ. روش دوم و سوم ۴ نخ. احتمالا با تعداد هسته‌های منطقی پردازنده‌ی من (که ۴ عدد است) ارتباط مستقیم دارد.

4. از نظر شما روش ایستا برای تقسیم کار بین نخ‌ها در این برنامه مناسب‌تر است یا روش پویا؟ چرا؟

روش ایستا. زیرا حجم کار بین نخ‌ها به صورت موازی تقسیم شده و در روش ایستا overhead کم‌تری داریم به همین دلیل استفاده از آن بهینه‌تر است.

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؟

تغییر و فهم کد عموما در تجزیه‌های یک بعدی آسان‌تر است ولی تجزیه‌های دو بعدی می‌توانند به کوچک‌تر کردن تسک‌ها کمک خوبی بکنند. در این برنامه تجزیه یک بعدی خیلی بهتر عمل می‌کند.

بخش امتیازی

با پیاده سازی روش بالا جدول زیر را تکمیل کنید و زمان‌ها را با سایر روش‌ها مقایسه نمایید.

تعداد نخ‌ها	اندازه هر ماتریس ورودی				تسریع
	1MB		1GB		
	Chunk size 1	Chunk size 128	Chunk size 1	Chunk size 128	
1	0.002883	0.002981	2.402808	2.416258	-
4	0.002260	0.002038	1.144161	1.098649	1.759
8	0.002252	0.002125	1.674122	1.044179	1.608
16	0.002791	0.002311	1.729891	1.027632	1.515

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

چون base این روش همان روش سوم است نمی‌توان توقع تسریع زیادی (نسبت به روش سطری) داشت اما به دلیل استفاده‌ی مستقیم از دستورات openmp و optimization های آن، باید از خود روش سوم بهتر عمل می‌کرد اما در همان ابتدا در optimization به مشکل می‌خورد و صرفا با افزایش تعداد نخ می‌توانیم تسریع تخریب‌شده‌ی آن (نسبت به روش سطری) را کمی بهبود بخشیم.