

۱ مرحله اول

ابتدا تغییراتی در کد برنامه سریال می‌دهیم و امکاناتی مانند اندازه‌گیری زمان اجرا و زمان میانگین چند اجرا را به آن اضافه می‌کنیم. پس از انجام این تغییرات، برنامه را به صورت سریال اجرا می‌کنیم. یک نمونه از اجرای کد سریال را برای ماتریس ۸ در ۸ در شکل ۱ می‌بینیم.

```
C:\Users\Radin\source\repos\ConsoleApplication1\Release\ConsoleApplication1.exe
[-] Dataset size is: 256 bytes

[-] Matrix A
30 48 52 72 62 10 49 53
84 90 90 6 44 45 68 18
39 24 99 65 12 65 19 45
27 5 12 90 69 64 62 63
30 21 37 35 6 51 35 34
40 61 85 9 18 19 33 49
59 10 27 96 40 68 56 58
20 14 65 86 75 76 22 70

[-] Matrix B
25 66 76 45 18 27 39 78
45 10 71 40 65 3 83 7
71 99 71 82 26 98 97 49
64 25 11 93 0 53 62 54
38 6 8 34 16 84 87 40
5 22 12 86 41 66 9 95
23 66 71 25 62 6 17 96
43 56 63 41 97 27 88 56

[-] Matrix C
55 114 128 117 80 37 88 131
129 100 161 46 109 48 151 25
110 123 170 147 38 163 116 94
91 30 23 183 69 117 124 117
68 27 45 69 22 135 122 74
45 83 97 95 59 85 42 144
82 76 98 121 102 74 73 154
63 70 128 127 172 103 110 126

[-] Time Elapsed: 0.008462 Secs

[-] The average running time was: 0.008462
Press any key to continue . . .
```

شکل ۱: نمونه اجرای کد سریال

۲ مرحله دوم

برای تجزیه یک بعدی، با استفاده از راهنمای زیر، می‌توانیم اجراهای حلقه بیرونی را موازی کنیم. همچنین برای بهتر دیدن تجزیه، شماره نخ را به جای جمع ماتریس‌ها داخل ماتریس خروجی می‌نویسیم. حاصل نوشتن شماره نخ داخل ماتریس خروجی را در شکل ۲ می‌بینیم. پس از اطمینان از درستی تجزیه، جهت اندازه‌گیری درست زمان اجرا مجدد برنامه را به حالت قبلی (جمع ماتریس) برمی‌گردانیم. همچنین یک نمونه از جمع صحیح دو ماتریس با تجزیه یک بعدی در شکل ۳ نشان داده شده است.

```

1 #pragma omp parallel for
2 for (i = 0; i < dataSet.n; i++) {
3     for (j = 0; j < dataSet.m; j++) {
4         dataSet.C[i * dataSet.m + j] = omp_get_thread_num();
5     }
6 }

```

```

[-] Matrix C
0      0      0      0      0      0      0      0
1      1      1      1      1      1      1      1
2      2      2      2      2      2      2      2
3      3      3      3      3      3      3      3
4      4      4      4      4      4      4      4
5      5      5      5      5      5      5      5
6      6      6      6      6      6      6      6
7      7      7      7      7      7      7      7

```

شکل ۲: نحوه تخصیص نخ‌ها در حالت تجزیه یک بعدی برای یک ماتریس ۸ در ۸

```

C:\Users\Radin\source\repos\ConsoleApplication1\Release\ConsoleApplication1.exe
[-] Dataset size is: 256 bytes

[-] Matrix A
24 14 98 88 45 92 80 35
64 14 41 81 3 25 21 20
86 66 39 6 1 42 15 51
79 64 82 4 13 54 5 73
1 36 87 35 87 3 71 79
96 31 81 1 30 70 78 18
33 55 56 15 81 52 23 79
62 24 31 17 62 9 43 4

[-] Matrix B
55 53 98 19 43 6 45 32
76 1 67 18 36 25 72 58
23 48 26 70 18 35 96 40
60 98 95 60 0 50 4 46
4 85 27 2 29 98 31 99
88 55 48 67 51 29 91 53
62 3 74 17 38 98 14 44
45 34 46 97 52 45 59 58

[-] Matrix C
79 67 196 107 88 98 125 67
140 15 108 99 39 50 93 78
109 114 65 76 19 77 111 91
139 162 177 64 13 104 9 119
5 121 114 37 116 101 102 178
184 86 129 68 81 99 169 71
95 58 130 32 119 150 37 123
107 58 77 114 114 54 102 62

[-] Time Elapsed: 0.008798 Secs

[-] The average running time was: 0.008798
Press any key to continue . . .

```

شکل ۳: یک نمونه از جمع ماتریس با تجزیه یک بعدی

پس از تجزیه یک بعدی، این بار به سراغ تجزیه دوبعدی می‌رویم. برای این کار تابع دیگری به نام `add2D` را به کد اضافه می‌کنیم. کد این تابع در فایل `matadd.cpp` آمده است. مشابه حالت قبل، ابتدا برای دیدن

جدول ۲: نتایج روش اول

تسریع	میانگین زمان اجرا (ثانیه)				تعداد نخ‌ها
	1GB	100MB	10MB	1MB	
۰/۹۴	۰/۴۵۴۹۱۴	۰/۰۴۷۸۴۵	۰/۰۰۴۲۸۸	۰/۰۰۰۵۶۷	۱
۱/۳۹	۰/۲۸۳۴۲۸	۰/۰۲۷۶۲۳	۰/۰۰۲۸۲۶	۰/۰۰۰۵۴۵	۲
۱/۵۰	۰/۲۴۰۸۵۴	۰/۰۲۳۸۳۱	۰/۰۰۲۶۱۶	۰/۰۰۰۶۴۲	۴
۲/۰۰	۰/۲۳۰۹۵۰	۰/۰۲۱۱۷۸	۰/۰۰۲۲۸۹	۰/۰۰۰۲۳۷	۸

جدول ۳: نتایج روش دوم

تسریع	میانگین زمان اجرا (ثانیه)				تعداد نخ‌ها
	1GB	100MB	10MB	1MB	
۱/۰۱	۰/۴۲۶۹۲۳	۰/۰۳۸۵۹۹	۰/۰۰۴۱۴۲	۰/۰۰۰۵۹۸	۱
۱/۳۹	۰/۲۷۵۳۶۵	۰/۰۲۵۹۹۳	۰/۰۰۲۹۵۹	۰/۰۰۰۵۶۶	۲
۱/۴۰	۰/۲۵۷۶۳۰	۰/۰۲۵۴۰۳	۰/۰۰۳۲۱۰	۰/۰۰۰۵۴۸	۴
۱/۷۷	۰/۲۱۳۱۸۳	۰/۰۲۲۴۳۱	۰/۰۰۲۸۰۹	۰/۰۰۰۳۱۷	۸