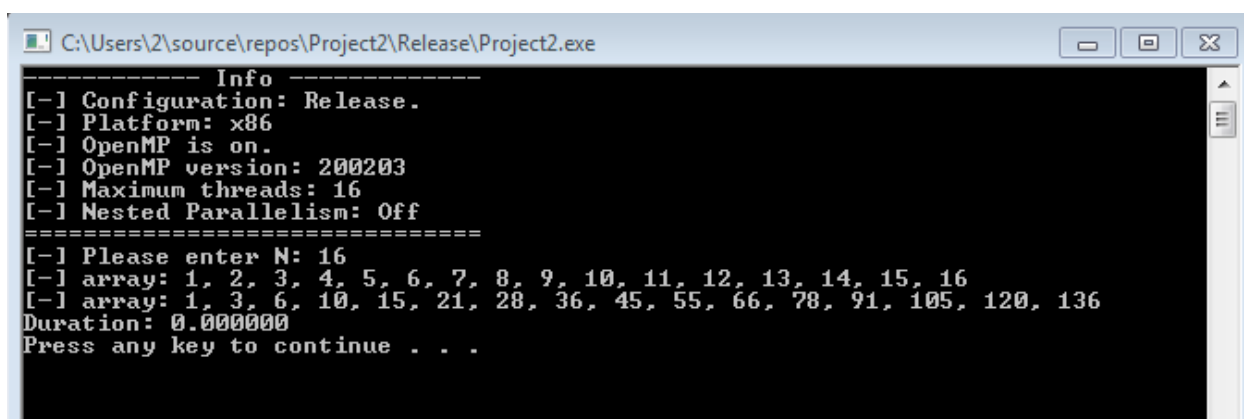


گزارش کار آزمایش ۴ برنامه نویسی چند هسته ای

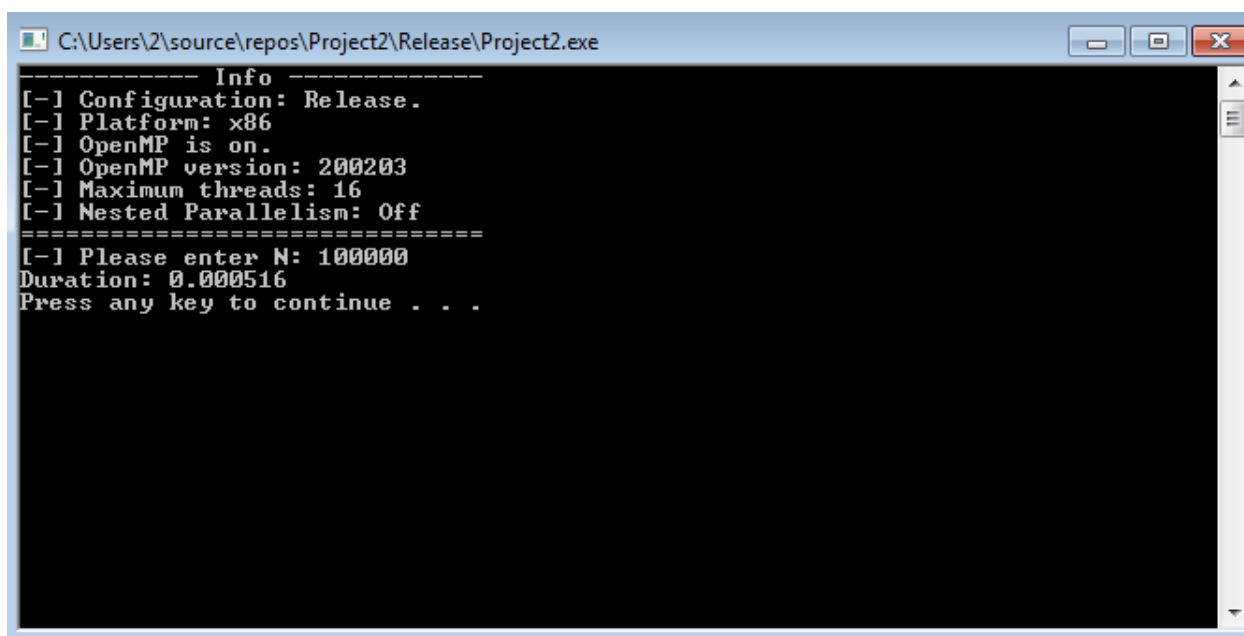
محمد مهدی نظری - ۹۹۳۱۰۶۱ و علی نوروزبیگی - ۹۹۳۱۰۶۲

* در هر مرحله درستی کد به ازای ۱۶ عضو و ۴ نخ آورده شده و سپس زمان اجرا به ازای ۱۰۰۰۰۰ عضو بررسی شده است.

(بخش سریال)



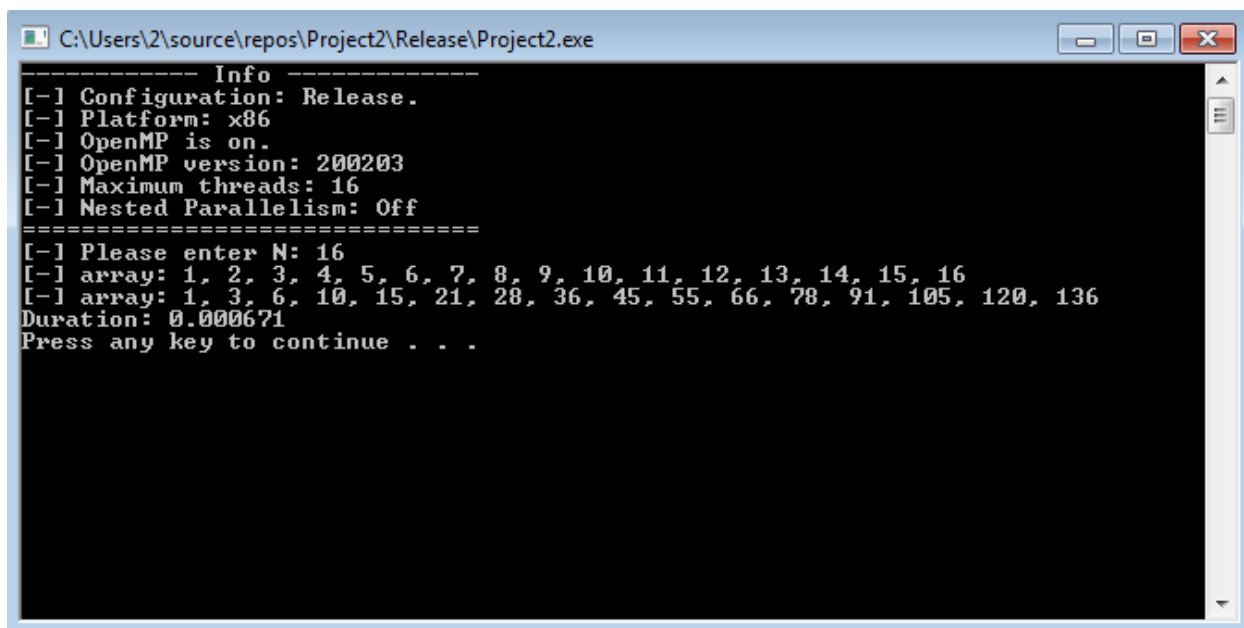
```
----- Info -----
[-] Configuration: Release.
[-] Platform: x86
[-] OpenMP is on.
[-] OpenMP version: 200203
[-] Maximum threads: 16
[-] Nested Parallelism: Off
=====
[-] Please enter N: 16
[-] array: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
[-] array: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136
Duration: 0.000000
Press any key to continue . . .
```



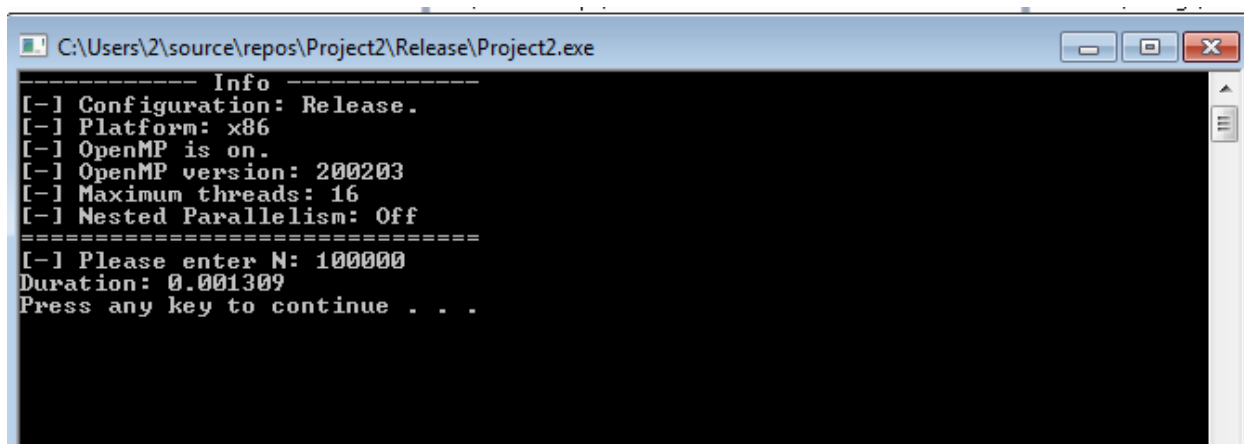
```
----- Info -----
[-] Configuration: Release.
[-] Platform: x86
[-] OpenMP is on.
[-] OpenMP version: 200203
[-] Maximum threads: 16
[-] Nested Parallelism: Off
=====
[-] Please enter N: 100000
Duration: 0.000516
Press any key to continue . . .
```

روش اول)

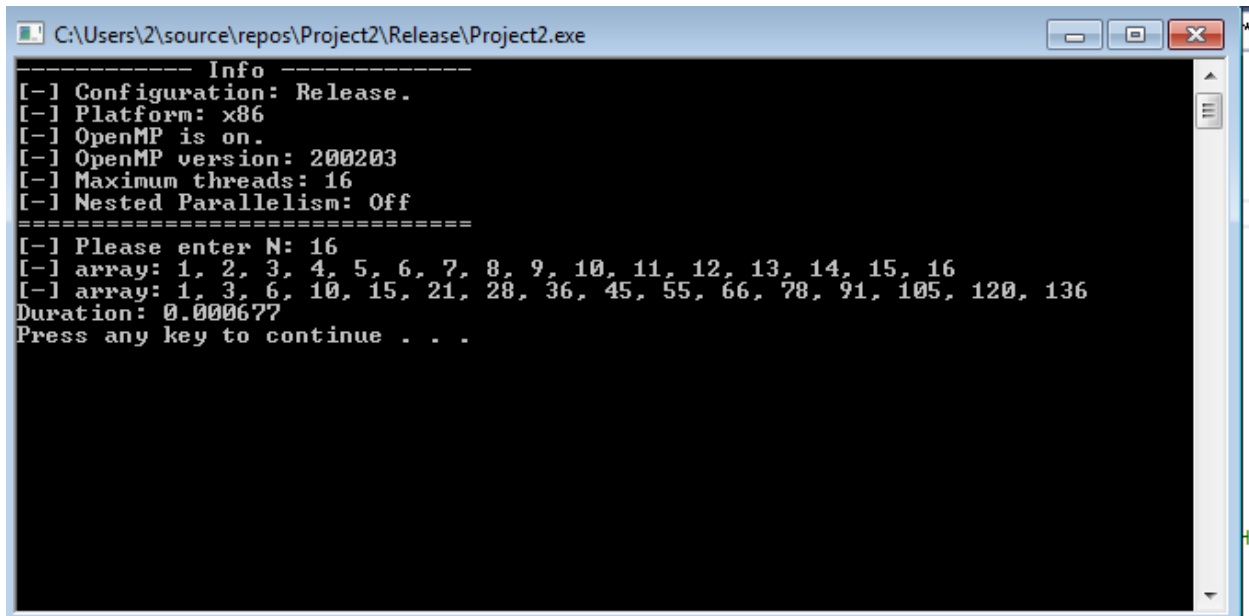
در این روش آرایه را به تعداد ترد ها تقسیم میکنیم و با استفاده از راهنمای schedule بصورت ایستا پردازش هر chunksize که در این کد n/# of threads تعریف کردیم را به یک نخ (به ترتیب) میدهیم. به عنوان مثال برای آرایه با ۱۶ عضو و ۴ نخ : خانه های ۰ تا ۳ را نخ ۰ ، خانه های ۴ تا ۷ را نخ ۱ ، خانه های ۸ تا ۱۱ را نخ ۲ و خانه های ۱۲ تا ۱۵ را نخ ۳ پردازش میکند . در نهایت برای اینکه آرایه نهایی را تولید کنیم باید آخرین خانه هر زیر آرایه را با همه خانه های زیر آرایه بعدی جمع کنیم.



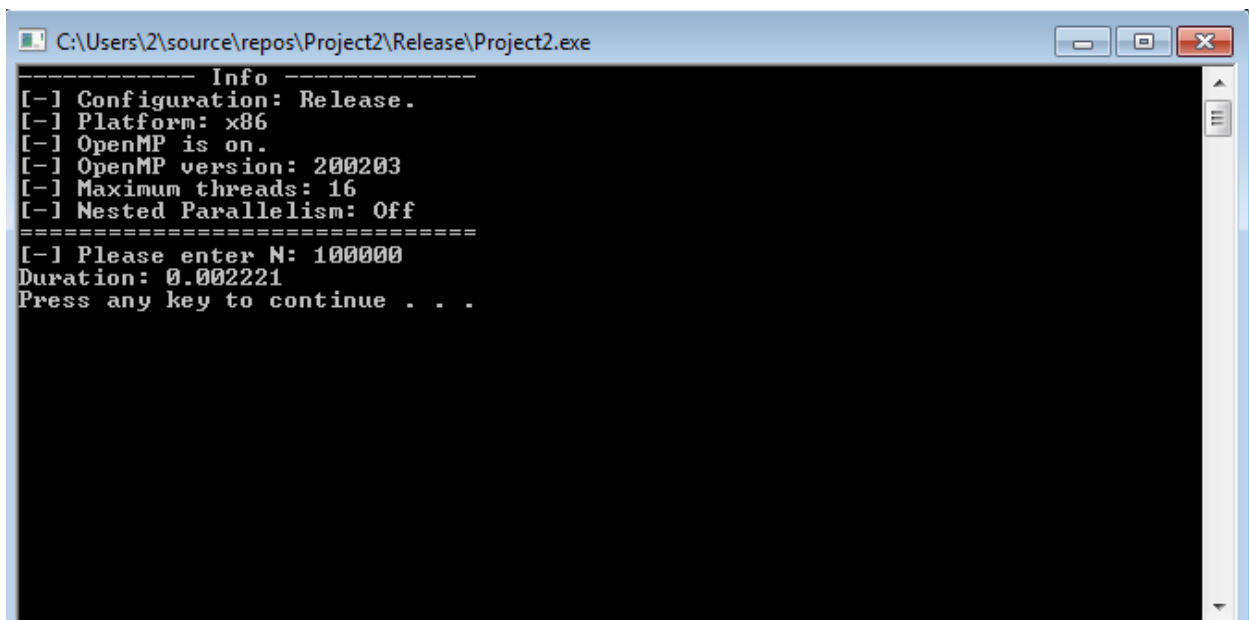
```
----- Info -----
[-] Configuration: Release.
[-] Platform: x86
[-] OpenMP is on.
[-] OpenMP version: 200203
[-] Maximum threads: 16
[-] Nested Parallelism: Off
=====
[-] Please enter N: 16
[-] array: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
[-] array: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136
Duration: 0.000671
Press any key to continue . . .
```



```
----- Info -----
[-] Configuration: Release.
[-] Platform: x86
[-] OpenMP is on.
[-] OpenMP version: 200203
[-] Maximum threads: 16
[-] Nested Parallelism: Off
=====
[-] Please enter N: 100000
Duration: 0.001309
Press any key to continue . . .
```



```
----- Info -----
[-] Configuration: Release.
[-] Platform: x86
[-] OpenMP is on.
[-] OpenMP version: 200203
[-] Maximum threads: 16
[-] Nested Parallelism: Off
=====
[-] Please enter N: 16
[-] array: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
[-] array: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136
Duration: 0.000677
Press any key to continue . . .
```



```
----- Info -----
[-] Configuration: Release.
[-] Platform: x86
[-] OpenMP is on.
[-] OpenMP version: 200203
[-] Maximum threads: 16
[-] Nested Parallelism: Off
=====
[-] Please enter N: 100000
Duration: 0.002221
Press any key to continue . . .
```

$$\text{SpeedUp(Parallel1)} = 0.000516/0.001309 = 0.39$$

$$\text{SpeedUp(Parallel2)} = 0.000516/0.002221 = 0.23$$

در الگوریتم دوم پردازش روی آرایه در هر حلقه ۲*۱ کاهش مییابد یعنی در مرحله اول خانه اول در مرحله دوم دو خانه اول در مرحله سوم ۴ خانه اول و به همین ترتیب . در الگوریتم اول تکرار آرایه فقط برای یک chunksize اتفاق میفتد پس از الگوریتم دوم سریعتر است. همچنین در الگوریتم دوم از آرایه موقت استفاده شده که ساختن و پرکردن آن زمان قابل ملاحظه ای میگیرد.

برای الگوریتم دوم و اول در OpenMP ما پردازش آرایه را بصورت سطری انجام دادیم بدلیل محدود بودن نخ ها اما در GPU بدلیل تعداد بالای ترد ها میتوان پردازش را بصورت عمودی انجام داد(یعنی هر ترد نتایج نهایی یک خانه را تولید کند) که در این صورت الگوریتم دوم سراسر تر و بهینه تر از الگوریتم اول است.