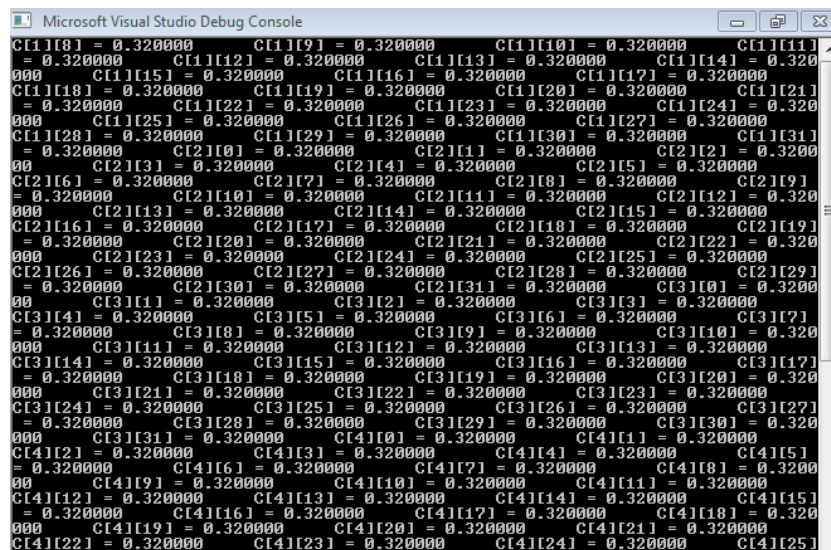


به نام خدا

گزارشکار آزمایش ششم

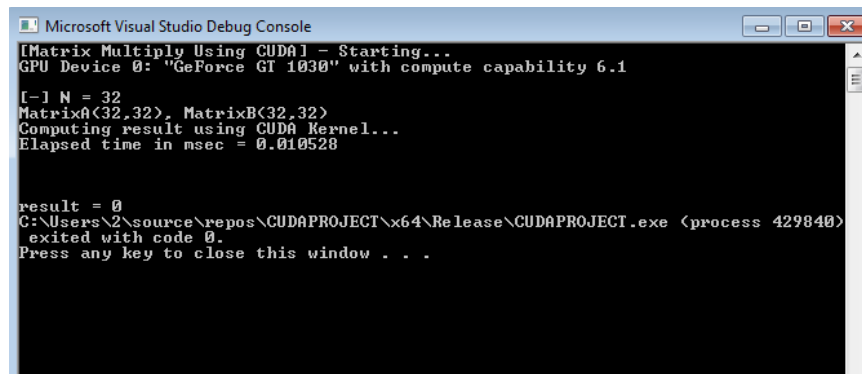
علی نوروزیگی ۹۹۳۱۰۶۲ - محمدمهدی نظری ۹۹۳۱۰۶۱

در گام اول، کد داده شده در دستورکار را بررسی می کنیم. در این کد، هر نخ مسئول محاسبه یک خانه در ماتریس نتیجه است. بعد از پیاده سازی تابع (که در کد پیوست شده با نام matrixMulCUDAPart01 وجود دارد)، مطابق با خواسته دستور کار، درستی ضرب ماتریس را بررسی می کنیم. چون ماتریس مربعی با اندازه 32 است و ماتریس A با مقادیر 1.0f و ماتریس b با مقادیر 0.01f پر شده اند، ماتریس نتیجه باید دارای مقادیر یکسان 0.32 باشد. در زیر تعدادی از درایه های ماتریس نتیجه را می بینیم:



```
Microsoft Visual Studio Debug Console
C[1][8] = 0.320000 C[1][9] = 0.320000 C[1][10] = 0.320000 C[1][11] = 0.320000
C[1][12] = 0.320000 C[1][13] = 0.320000 C[1][14] = 0.320000 C[1][15] = 0.320000
C[1][16] = 0.320000 C[1][17] = 0.320000 C[1][18] = 0.320000 C[1][19] = 0.320000
C[1][20] = 0.320000 C[1][21] = 0.320000 C[1][22] = 0.320000 C[1][23] = 0.320000
C[1][24] = 0.320000 C[1][25] = 0.320000 C[1][26] = 0.320000 C[1][27] = 0.320000
C[1][28] = 0.320000 C[1][29] = 0.320000 C[1][30] = 0.320000 C[1][31] = 0.320000
C[2][0] = 0.320000 C[2][1] = 0.320000 C[2][2] = 0.320000 C[2][3] = 0.320000
C[2][4] = 0.320000 C[2][5] = 0.320000 C[2][6] = 0.320000 C[2][7] = 0.320000
C[2][8] = 0.320000 C[2][9] = 0.320000 C[2][10] = 0.320000 C[2][11] = 0.320000
C[2][12] = 0.320000 C[2][13] = 0.320000 C[2][14] = 0.320000 C[2][15] = 0.320000
C[2][16] = 0.320000 C[2][17] = 0.320000 C[2][18] = 0.320000 C[2][19] = 0.320000
C[2][20] = 0.320000 C[2][21] = 0.320000 C[2][22] = 0.320000 C[2][23] = 0.320000
C[2][24] = 0.320000 C[2][25] = 0.320000 C[2][26] = 0.320000 C[2][27] = 0.320000
C[2][28] = 0.320000 C[2][29] = 0.320000 C[2][30] = 0.320000 C[2][31] = 0.320000
C[3][0] = 0.320000 C[3][1] = 0.320000 C[3][2] = 0.320000 C[3][3] = 0.320000
C[3][4] = 0.320000 C[3][5] = 0.320000 C[3][6] = 0.320000 C[3][7] = 0.320000
C[3][8] = 0.320000 C[3][9] = 0.320000 C[3][10] = 0.320000 C[3][11] = 0.320000
C[3][12] = 0.320000 C[3][13] = 0.320000 C[3][14] = 0.320000 C[3][15] = 0.320000
C[3][16] = 0.320000 C[3][17] = 0.320000 C[3][18] = 0.320000 C[3][19] = 0.320000
C[3][20] = 0.320000 C[3][21] = 0.320000 C[3][22] = 0.320000 C[3][23] = 0.320000
C[3][24] = 0.320000 C[3][25] = 0.320000 C[3][26] = 0.320000 C[3][27] = 0.320000
C[3][28] = 0.320000 C[3][29] = 0.320000 C[3][30] = 0.320000 C[3][31] = 0.320000
C[4][0] = 0.320000 C[4][1] = 0.320000 C[4][2] = 0.320000 C[4][3] = 0.320000
C[4][4] = 0.320000 C[4][5] = 0.320000 C[4][6] = 0.320000 C[4][7] = 0.320000
C[4][8] = 0.320000 C[4][9] = 0.320000 C[4][10] = 0.320000 C[4][11] = 0.320000
C[4][12] = 0.320000 C[4][13] = 0.320000 C[4][14] = 0.320000 C[4][15] = 0.320000
C[4][16] = 0.320000 C[4][17] = 0.320000 C[4][18] = 0.320000 C[4][19] = 0.320000
C[4][20] = 0.320000 C[4][21] = 0.320000 C[4][22] = 0.320000 C[4][23] = 0.320000
C[4][24] = 0.320000 C[4][25] = 0.320000
```

اکنون که از درستی برنامه مطمئن شدیم، زمان اجرای برنامه را بررسی می کنیم، اندازه بلاک ۱۶ در نظر گرفته شده:

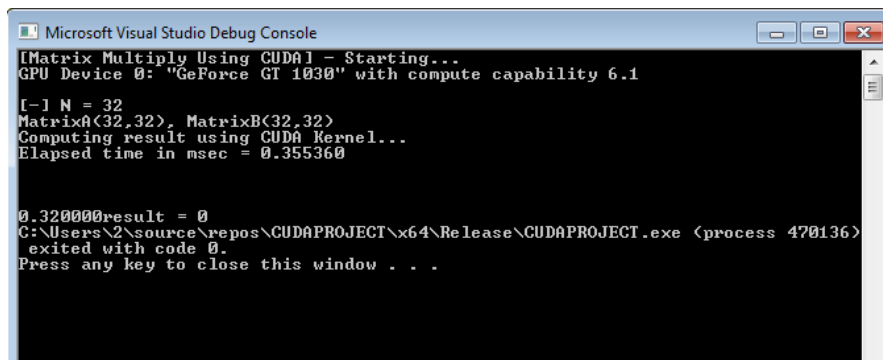


```
Microsoft Visual Studio Debug Console
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1

[-] N = 32
MatrixA(32,32), MatrixB(32,32)
Computing result using CUDA Kernel...
Elapsed time in msec = 0.010528

result = 0
C:\Users\2\source\repos\CUDAPROJECT\64\Release\CUDAPROJECT.exe (process 429840)
exited with code 0.
Press any key to close this window . . .
```

در گام دوم برنامه را توسعه می دهیم تا بتوانیم مقادیر بزرگتر را نیز بررسی کنیم. در مرحله اول، روشی را پیاده می کنیم که در آن هر نخ کار بیشتری را انجام دهد. پس مطابق با کد دستور کار، این کد را پیاده می کنیم و برای اطمینان از درستی برنامه تنها درایه اول ماتریس نتیجه را چاپ می کنیم. در کد این برنامه با تابعی با نام `matrixMulCUDApart02Case01` برنامه را اجرا می کند. لازم به ذکر است که اندازه بلاک در این حالت برابر با ۱۶ در نظر گرفته شده:

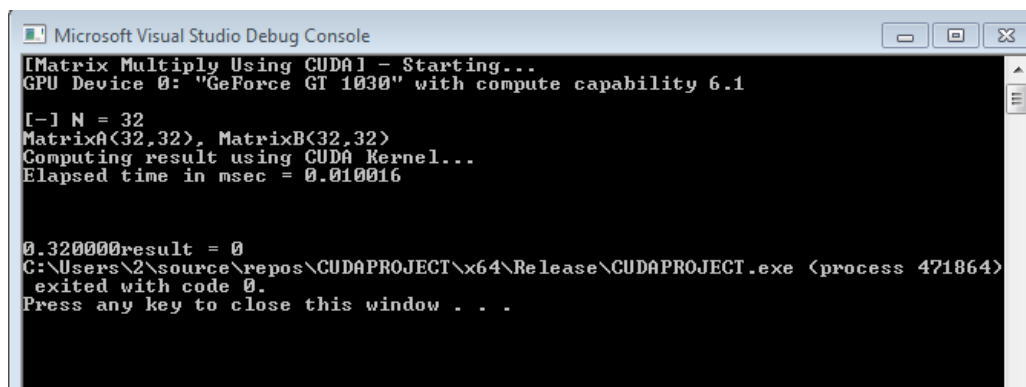


```
Microsoft Visual Studio Debug Console
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1

[-1] N = 32
MatrixA(32,32), MatrixB(32,32)
Computing result using CUDA Kernel...
Elapsed time in msec = 0.355360

0.320000result = 0
C:\Users\2\source\repos\CUDAPROJECT\x64\Release\CUDAPROJECT.exe (process 470136)
exited with code 0.
Press any key to close this window . . .
```

روش دوم این است که از چند بلاک استفاده کنیم. برای این منظور، باید تعداد `grid` و `threads` را متناسب با اندازه و تعداد بلاک در نظر بگیریم. به همین منظور از `dim3 grid2` و `dim3 threads2` برای این مرحله و مراحل بعدی استفاده می کنیم که جزییات آن در کد قابل مشاهده است. بعد از پیاده سازی کد موجود در دستور کار برای مرحله دوم (`matrixMulCUDApart02Case02`)، با اندازه بلاک ۱۶ نتیجه به صورت زیر است:



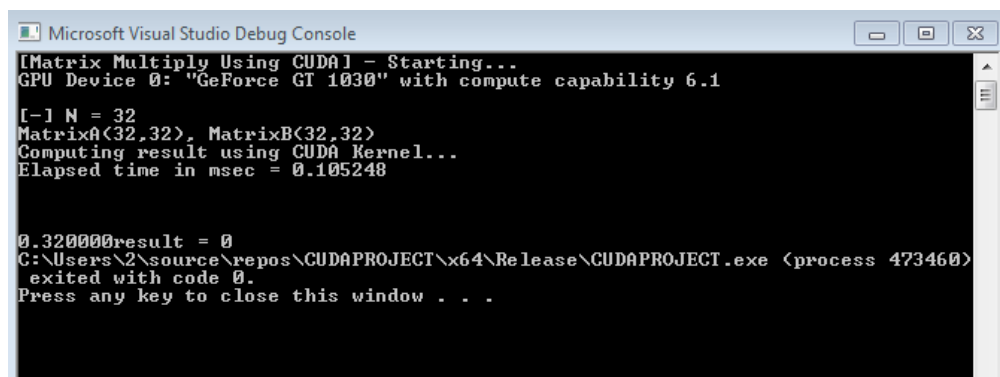
```
Microsoft Visual Studio Debug Console
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1

[-1] N = 32
MatrixA(32,32), MatrixB(32,32)
Computing result using CUDA Kernel...
Elapsed time in msec = 0.010016

0.320000result = 0
C:\Users\2\source\repos\CUDAPROJECT\x64\Release\CUDAPROJECT.exe (process 471864)
exited with code 0.
Press any key to close this window . . .
```

همانطور که مشاهده می شود، در این مرحله چون از تعداد نخهای بیشتری استفاده می کنیم، زمان اجرا بسیار سریع تر از مرحله قبل است.

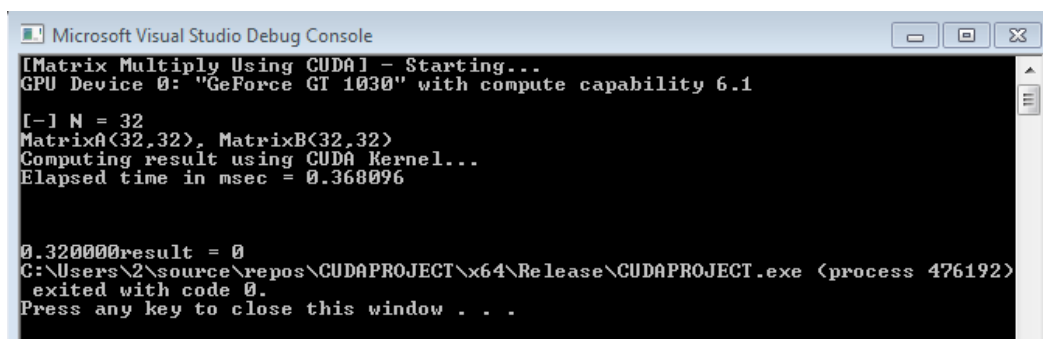
مرحله سوم از گام دوم، اجرا به وسیله کاشی کاری است. در این مرحله اندازه کاشی تاثیر زیادی دارد. به همین دلیل بعد از پیاده سازی کد موجود در دستور کار (matrixMulCUDAPart02Case03)، برنامه را با سه اندازه کاشی بررسی میکنیم. در ابتدا با اندازه کاشی ۸ نتیجه زیر را می گیریم:



```
Microsoft Visual Studio Debug Console
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1
[-] N = 32
MatrixA<32,32>, MatrixB<32,32>
Computing result using CUDA Kernel...
Elapsed time in msec = 0.105248

0.320000result = 0
C:\Users\2\source\repos\CUDAPROJECT\x64\Release\CUDAPROJECT.exe (process 473460)
exited with code 0.
Press any key to close this window . . .
```

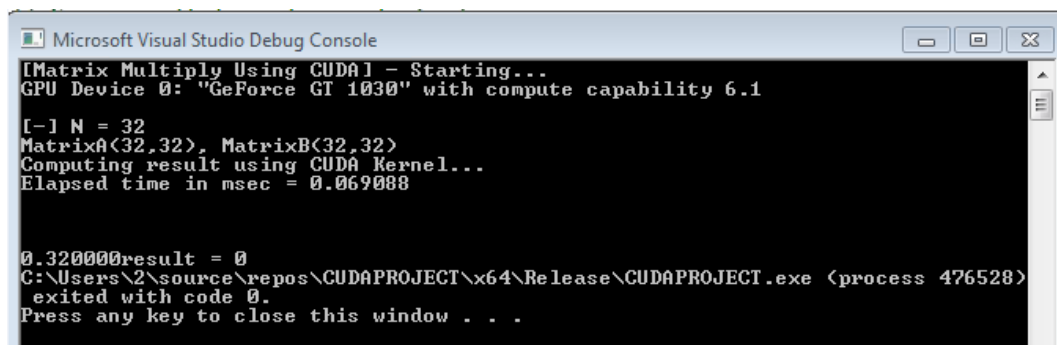
با اندازه کاشی ۱۶ نتیجه زیر بدست می آید که کمی طولانی تر از اندازه ۸ است:



```
Microsoft Visual Studio Debug Console
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1
[-] N = 32
MatrixA<32,32>, MatrixB<32,32>
Computing result using CUDA Kernel...
Elapsed time in msec = 0.368096

0.320000result = 0
C:\Users\2\source\repos\CUDAPROJECT\x64\Release\CUDAPROJECT.exe (process 476192)
exited with code 0.
Press any key to close this window . . .
```

با اندازه کاشی ۳۲ داریم:

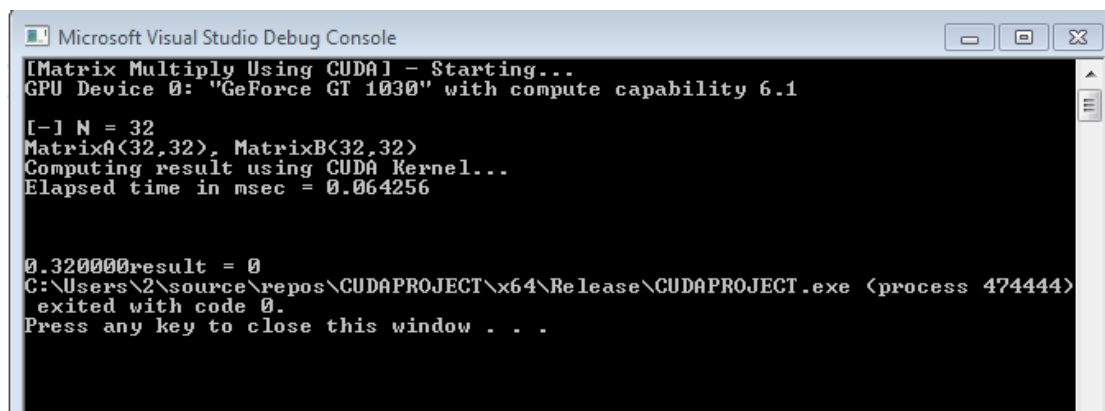


```
Microsoft Visual Studio Debug Console
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1
[-] N = 32
MatrixA<32,32>, MatrixB<32,32>
Computing result using CUDA Kernel...
Elapsed time in msec = 0.069088

0.320000result = 0
C:\Users\2\source\repos\CUDAPROJECT\x64\Release\CUDAPROJECT.exe (process 476528)
exited with code 0.
Press any key to close this window . . .
```

همانطور که مشاهده می شود، با اندازه ۳۲ سریع ترین حالت را داریم.

در گام سوم، با استفاده از حافظه مشترک برنامه را پیاده سازی می کنیم. همانطور که از اسلایدها می دانیم برای اینکه بتوانیم از حافظه مشترک بین نخهای موجود در یک بلاک استفاده کنیم، در کرنل می توانیم با نوشتن `__shared__` حافظه مشترک را مشخص کنیم. همانطور که در تابع `matrixMulCUDApart03` مشخص است، ابتدا به وسیله کلید واژه ذکر شده دو حافظه مشترک تعریف می کنیم. سپس طبق اسلاید، در مرحله اول همه نخ ها باید داده ها از داده اصلی به داده مشترک بارگذاری کنند. بعدا همه آنها با هم باید عملیات ضرب را شروع کنند، به همین دلیل باید نخ ها را با هم سینک کنیم. سپس عملیات ضرب انجام می شود اما بعد از آن برای اینکه از به پایان رسیدن محاسبات متناظر با آن کاشی اطمینان پیدا کنیم، دوباره باید نخ ها را با هم سینک کنیم و بعد نتیجه را ذخیره کنیم. نتیجه با اندازه ۳۲ بلاک به صورت زیر است:



```
Microsoft Visual Studio Debug Console

[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "GeForce GT 1030" with compute capability 6.1

[-] N = 32
MatrixA<32,32>, MatrixB<32,32>
Computing result using CUDA Kernel...
Elapsed time in msec = 0.064256

0.320000result = 0
C:\Users\2\source\repos\CUDAPROJECT\x64\Release\CUDAPROJECT.exe (process 474444)
exited with code 0.
Press any key to close this window . . .
```