علیرضا آخوندی
۹۷۳۱۱۰۷
HW5

سوال اول : C

سوال دوم : باید حداقل ۸۰۰۰ نخ داشته باشیم و از آنجایی که تعداد نخ های گرید مضربی از ۱۰۲۴ باید باشد آنگاه در مجموع ۱۰۲۴ * ۸ = ۸۱۹۲ نخ نیاز است.

سوال سوم :
Compute capability عددی است که نشان می دهد که معماری gpu مورد استفاده متعلق به چه نسلی است.در واقع هر نسلی که آمده است قابلیت هایی به آن اضافه شده است که gpu های دیگر ندارند.

سوال چهارم :
PTX یا parallel thread execution یک ماشین مجازی و ISA است که در محیط برنامه نویسی کودا از آن استفاده می شود. در واقع کامپایلر nvcc کد کودا یا c++ نوشته شده را به این مجموعه دستورات ترجمه کرده و در graphic driver ها این کد نوشته شده را به کد باینری در سطح ماشین ترجمه می کنند.در واقع PTX این امکان را به ما می دهد که کد خود را بر روی gpu ها مختلف با ساختار متفاوت اجرا کنیم بدون آن که کد خود را تغییر دهیم.
در واقع syntax ptx شامل یک سری دستور عمل های شبیه assembly و یک سری directive است (مانند .version و .global و .reg). نمونه ای از کد PTX در مثال زیر آمده است:

```
        .reg     .b32 r1, r2;
        .global  .f32  array[N];

start:  mov.b32   r1, %tid.x;
        shl.b32   r1, r1, 2;          // shift thread id by 2 bits
        ld.global.b32 r2, array[r1];  // thread[tid] gets array[tid]
        add.f32   r2, r2, 0.5;        // add 1/2
```

سوال ۵ :

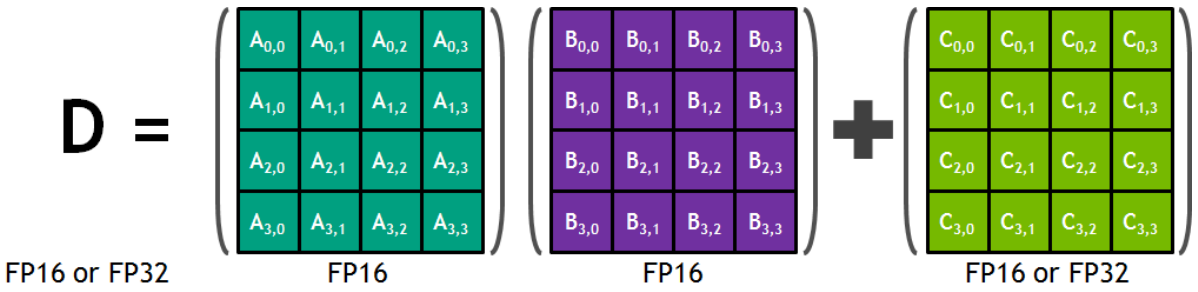
هسته های cuda در واقعا ساده ترین واحد های پردازشی هستند که یک stream از instruction هارا برای ما اجرا میکنند. این واحد به global memory ، shared memory و file register داخل sm دسترسی دارد.

طبق داکیومنت زیر از nvidia امکان عملیات mixed precision از cuda 8 اضافه شده است.

/https://developer.nvidia.com/blog/mixed-precision-programming-cuda-8

سوال ۶ :

Tensor core یک واحد محاسباتی است که وظیفه آن انجام عملیات ضرب و انباشت ماتریس های ۴ در ۴ است. این واحد محاسباتی برای اولین بار در معماری Volta از کارت های گرافیکی معرفی شد. در ابتدا این واحد ها فقط می توانستند روی FP16 این عملیات را انجام دهند ، اما با آمدن معماری های بعدی امکان عملیات بر رو precision های دیگر نیاز مهیا شد.با اومدن این سخت افزار ، از کودای ۹ به بعد نیز توابعی اضافه شدند که با استفاده از آن ها می توان از tensor core ها استفاده کرد.



/https://www.nvidia.com/en-us/data-center/tensor-cores

/https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9

سوال ۷ :
Openmp برای موازی سازی کد در سیستم های چند هسته ای است در حالی کد کودا برای موازی سازی عملیات های محاسباتی با کمک gpu است. در واقع لینکر کامپایلر nvcc این امکان را به ما میدهد که از openmp نیز است استفاده کنیم (–fopenmp Xcompiler باید فعال شوند تا بتوانیم از openmp در کد سمت host

استفاده کنیم.در واقع بهتر است بخشی از کد که در آن محاسبات ریاضی بر روی ساختمان داده های خطی مانند ماتریس ها ، وکتور ها و تنسور هارا با کمک کرنل های gpu (یا همان device) انجام دهیم و بقیه کد که برای ما latency اهمیت بیشتری نسبت به throughput دارد به کمک openmp موازی سازی کنیم.حتی می توانیم از معماری لایه ای استفاده کنیم و کد های مربوط به کرنل را در لایه پایین نوشته و یک موازی سازی نیز بر روی فراخوانی های این توابع در لایه بالاتر توسط openmp انجام دهیم.

سوال ۸ : در داخل فایل print_thread_info.cu

سوال ۹ : تمام پیاده سازی های این بخش در فایل vector_add.cu آمده است.
(a

پیاده سازی این بخش در تابع serial_vector_add آمده است.
زمان اجرا : time for serial: 0.026330 s

(b

پیاده سازی این بخش در تابع omp_parallel_vector_add آمده است.
زمان اجرا : time for openMP: 0.006384 s

c) پیاده سازی در تابع cuda_parallel_vector_add آمده است.
زمان اجرا : time for CUDA: 0.001217 s

در این سوال از آنجایی که تعداد بلاک ها(یا همان اندازه گرید) را طوری تنظیم کرده ایم که تعداد نخ ها به اندازه دو آرایه نزدیک باشه ، با زیاد شدن اندازه بلاک ها ، اندازه گرید کوچکتر شده و با کوچکتر شدن آن نیز ابعاد گرید بیشتر می شود.

Block size 16 : 0.004201 s
Block size 32 : 0.002133 s
Block size 128 : 0.001225 s
Block size 256 : 0.001221 s
Block size 512 : 0.001217 s

همانطور که دیده می شود با بزرگتر شدن اندازه بلاک ها و کوچکتر شدن اندازه گرید ها تسریع بیشتری گرفتیم.البته با بزرگتر شدن بیش از حد اندازه بلاک ها ممکن است.occupancy نیز پایین بیاید.

D)داخل کد یک متغیر به نام STRIDE در نظر میگیریم که نشان دهنده تعداد عناصری هست که هر ترد باید جمع آن ها را محاسبه کند.حال زمان اجرا را به ازای مقادیر stride متفاوت به دست می آوریم.block size را نیز 256 در نظر می گیریم.

Stride 1 : 0.001221 s

Stride 10 : 0.015311 s

Stride 20 : 0.047919 s

Stride 30 : 0.097765 s

همانطور که دیده می شود با بالا رفتن stride ، کم شدن تعداد نخ ها و درشت دانه تر شدن زمان اجرا بالا رفته است. دلیل آن این است که در gpu ما تعداد زیادی نخ داریم که قدرت پردازشی آن ها کم است پس بهتر است از مکانیزم ریزدانگی استفاده کنیم و سعی کنیم به هر نخ کار کمتری اختصاص دهیم و به جای آن تعداد نخ های مورد استفاده را بیشتر کنیم.

اندازه مناسب grid و block ها باید بر اساس تعداد sm ها و تعداد نخ ها و بلاک های هر sm تعیین شود تا بتوانیم occupancy را بیشینه کنیم .

سوال ۱۰ :
الف) ptx در واقع یک ماشین مجازی و ISA است که برای برنامه نویسی سطح پایین طراحی شده است در حالی که sass یک زبان اسمبلی سطح پایین تر است که به کد باینری کامپایل می شود. در واقع ptx امکان برنامه نویس موازی را برای برنامه نویس مهیا می کند.

ب) از آنجایی که این تابع یک error code برمی گرداند و ما می خواهیم که یه یک پوینتر حافظه بدهیم و توابع در زبان c به صورت pass by value کار میکنند ، نمی‌تونیم یک pointer یگانه بفرستیم. بنابراین لازم است که آدرس مربوط به آن پوینتر را بفرستیم.حال آن که تابع malloc برای حل این مشکل مقدار پوینتر مورد نیاز را برمی گرداند.