**Zbiór Mandelbrota w CUDA**

Napisz program w CUDA implementujący 2 różne wersje algorytmu tworzenia graficznej reprezentacji zbioru Mandelbrota i jego najbliższego otoczenia.
Porównaj czas wykonania tego obrazka w wersji CPU i wersji CUDA.
Dla każdej wersji zbadaj zależność czasu wykonania programu od konfiguracji kernela, zgodnie z załączonymi tabelami.

W katalogu ~w.rudnicki/Public/Mandelbrot znajdują się dwa pliki Mandelbrot_cuda.h oraz Mandelbrot_new_skel.cu, które mogą stanowić punkt wyjścia do własnej implementacji.
W pliku Mandelbrot_cuda.h znajdują się deklaracje potrzebnych funkcji i definicja liczb rzeczywistych.

W pliku Mandelbrot.cu znajduje się funkcja main(), funkcje pomocnicze do generowania obrazków, funkcje generujące zbiór Mandelbrota dla CPU oraz szkielety dwóch wersji kernela generującego zbiór Mandelbrota na GPU a także funkcja pomocnicza do porównania wyników. W funkcji main są dwa dodatkowe parametry - flagi sterujące generowaniem obrazka i wykonaniem porównania z wersją CPU.
Ponadto w funkcji main() są komentarze wskazujące na miejsce w którym trzeba uzupełnić kod, jest podana przykładowa konfiguracja uruchomienia kernela jednowymiarowego, oraz zakomentowane konfiguracje i wywołania kernela dwuwymiarowego.

W ramach zadania należy:

1. Uzupełnić kod w kernelach i funkcji main, tak aby stworzyć funkcjonalny program generujący zbiór Mandelbrota i jego graficzną reprezentację. Uwaga - dopuszczalne i mile widziane jest napisanie programu samodzielnie, lub dowolnie głębokie modyfikacje dostarczonego kodu.
2. Uruchomić kod i porównać wydajność wersji CPU i GPU. Należy wykonać co najmniej 11 uruchomień dla każdej wersji kodu. W raporcie należy zamieścić dyskusję o źródłach fluktuacji wyników czasowych dla GPU i dla CPU.
3. Porównanie wydajności różnych wersji kodu z różnymi wartościami parametrów uruchomienia (liczba wątków w bloku w kernelu jednowymiarowym, wymiary bloku w wersji dwuwymiarowej) dla następującego wycinak płaszczyzny zespolonej: {-2. -1.25} {0.5 1.25
4. Zastanowić się skąd wynikają różnice (lub ich brak) w wersjach GPU. W raporcie neleży wykorzystać poniższe tabele.
5. Do porównań wydajności różnych wersji i i różnych parametrów uruchomienia kernela należy wziąć sam czas wykonania kernela. Dla porównania czasów wykonania różnych wariantów GPU należy użyć polecenia rozdzielczości 10 000 punktów.
6. Jako referencyjnego czasu CPU należy użyć czas uzyskany przy generowaniu zbioru z 10 razy mniejszą rozdzielczością (1000x1000 zamiast 10000x10000) przemnożony przez 100.
7. **Uwaga:** karta GPU na węzłach w chmurze gdy nie jest używana do obliczeń ma obniżaną częstotliwość zegara dla obniżenia zużycia energii. System zarządzający zwiększa częstotliwość zegara do wartości maksymalnej dopiero po kilku-(kilkunasto-)sekundowym pełnym obciążeniu. Poza tym maszyna wirtualna ma duże fluktacje przydzielonych zasobów. W związku z tym porównania wydajności należy robić według następującej procedury:
   1. Porównywane wywołania wywołania różnych wersji kodu i różnych konfiguracji powinny być uruchomione w ramach jednego wywołania.
   2. Należy umieścić wywołanie wersji referencyjnej wraz z pomiarem czasu wielokrotnie w ramach jednego wywołania programu np. według następującego wzorca:
      wersja_CPU
      GPU1D_32
      GPU1D_64
      ...
      GPU1D_1024
      wersja_CPU
      GPU2D_par1
      GPU2D_par2
      ...
      GPU2D_par4
      wersja_CPU
   3. Należy do analizy wziąć tylko te uruchomienia dla których czas wersji CPU jest stabilny (różnice czasu wywołania mniejsze niż 1% ).

8. **UWAGA:** sposób prezentacji wyników:
    1. Dla każdego wariantu liczymy średni czas wykonania $\bar{T} = \frac{\sum_i T_i}{N}$, gdzie N jest liczbą pomiarów (np 11). wariancję $V = \frac{\sum_i (T_i - \bar{T})^2}{N-1}$ i odchylenie standardowe $S = \sqrt{V}$. Obliczamy odchylenie standardowe wartości średniej jako $S_m = \frac{S}{\sqrt{N}}$.
    2. Zaokrąglamy $S_m$ do dwóch cyfr znaczących **w górę!**
    Przykłady: 0.783456 - 0.79; 11.456 - 12, 99.01 -> 100; 100.01 -> 110, 0.00879345 - 0.0088
    3. Podajemy wynik średni w postaci $\bar{T} \pm S_m$ z precyzją wynikającą z zaokrąglenia $S_m$. Zaokrąglamy wynik wg standardowych reguł. Przykłady: (1159,9090, 13,1609): 1160 ± 14 (1160,9510, 10,6509): 1161 ± 14, (1476,9220, 0,4992): 1476.92 ± 0.50.
    4. Analogicznie liczymy przyspieszenie względem CPU, i również podajemy jako wartość średnią z odchyleniem standardowym średniej.
    5. Wartości typowego czasu wykonania (mediana) i minimalny czas wykonania podajemy w zaokrągleniu, z tą samą precyzją co wartość średnią czasu.

UWAGA
Warto przygotować schemat obliczeń, który wywołuje w pętli kilka kolejnych wersji/konfiguracji kernela w jednym wywołaniu funkcji, zapisuje wyniki do tablicy pomocniczej, a następnie oblicza wszystkie wartości potrzebne do raportu w wersji niemal gotowej do prezentacji - jedynie zaokrąglenie zostawiłbym do poprawienia ręcznego.
W kolejnych zadaniach raporty będą przygotowywane według tego samego schematu.

Raport powinien składać się z następujących części:

Sekcja tytułowa (Tytuł opracowania, Autor, nr albumu).

Wstęp - krótka informacja o badanym problemie (np. co to jest zbiór Mandelbrota) plus co jest przedmiotem badania (różnice w wydajności w zależności od konfiguracji kernela oraz rodzaju kernela)

Opis implementacji (krótko, włączyć/zacytować rdzeń kernela).

Wyniki - tabele wraz z omówieniem.

Podsumowanie - wnioski

Tabela 1. Podsumowanie wyników – wersja 1D

| Liczba wątków w bloku | Typowy czas wykonania (ms) | Minimalny czas wykonania (ms) | Średni czas wykonania (ms) | Przyspieszenie względem wersji referencyjnej |
|---|---|---|---|---|
| **CPU** | | | | 1 |
| **32** | | | | |
| **64** | | | | |
| **128** | | | | |
| **256** | | | | |
| **512** | | | | |
| **1024** | | | | |

Tabela 2. Podsumowanie wyników - wersja 2D 256 wątków

| Konfiguracja kernela | | Typowy czas wykonania (ms) | Minimalny czas wykonania (ms) | Średni czas wykonania (ms) | Przyspieszenie względem wersji referencyjnej |
|---|---|---|---|---|---|
| **X** | **Y** | | | | |
| **CPU** | | | | | 1 |
| **256** | **1** | | | | |
| **128** | **2** | | | | |
| **64** | **4** | | | | |
| **32** | **8** | | | | |
| **16** | **16** | | | | |
| **8** | **32** | | | | |
| **4** | **64** | | | | |
| **2** | **128** | | | | |
| **1** | **256** | | | | |

Tabela 3. Podsumowanie wyników - wersja 2D 1024 wątki

| Konfiguracja kernela | | Typowy czas wykonania (ms) | Minimalny czas wykonania (ms) | Średni czas wykonania (ms) | Przyspieszenie względem wersji referencyjnej |
|---|---|---|---|---|---|
| X | Y | | | | |
| CPU | | | | | 1 |
| 1024 | 1 | | | | |
| 512 | 2 | | | | |
| 256 | 4 | | | | |
| 128 | 8 | | | | |
| 64 | 16 | | | | |
| 32 | 32 | | | | |
| 16 | 64 | | | | |
| 8 | 128 | | | | |
| 4 | 256 | | | | |
| 2 | 512 | | | | |
| 1 | 1024 | | | | |

Tabela 4. Podsumowanie wyników - wersja 2D: inne przypadki

| Konfiguracja kernela | | Typowy czas wykonania (ms) | Minimalny czas wykonania (ms) | Średni czas wykonania (ms) | Przyspieszenie względem wersji referencyjnej |
|---|---|---|---|---|---|
| X | Y | | | | |
| CPU | | | | | 1 |
| 32 | 32 | | | | |
| 16 | 16 | | | | |
| 8 | 8 | | | | |
| 32 | 16 | | | | |
| 64 | 8 | | | | |
| 8 | 64 | | | | |
| 16 | 32 | | | | |