|  |  |
| --- | --- |
|  | **AGH – UNIVERSITY OF SCIENCE AND TECHNOLOGY** |

Dokumentacja projektowa dla

**Operacje macierzowe z wykorzystaniem technologii CUDA**

**Architektura I Programowanie GPU**

Informatyka i Systemy Inteligentne, I rok

*Adrian Smoła, Kacper Godula*

Prowadzący: Jędrzej Byrski

09.06.2022

1. **Opis Projektu**  
   Celem projektu jest zaprezentowanie wybranych operacji macierzowych, wykonanych w technologii CUDA. Głównym zamierzeniem jest lepsze zrozumienie operacji zrównoleglonych używanych w działaniach na macierzach, a także znaczące przyspieszenie ich działania w stosunku do klasycznych implementacji sekwencyjnych.
2. **Kompilacja**W celu uruchomienia projektu, zalecane jest uruchomienie poszczególnych projektów w Visual Studio 2022, lub po prostu uruchomić załączony plik exe.
3. **Architektura projektu**
   1. Matrix\_x\_scalar – Służy do podstawowego mnożenia macierzy przez skalar używając technologii CUDA. Zawiera 3 funkcje i jednego kernela.
      1. Void main() – Jest to główna funkcja programu, w niej deklarujemy pamięć, wywołujemy inne funkcje oraz kernele.
      2. Void printResults(float\*\* A, float\*\* B, int width) – Jest to funkcja użyta w celu wypisania elementów macierzy A oraz macierzy B o szerokości width
      3. Void random\_ints(float\*\* matrix, size\_t N, size\_t M) - wypełnia macierz matrix pseudo losowymi liczbami, parametry N oraz M określają wymiary macierzy.
      4. \_\_global\_\_ void Matrix\_multiplication(float \*A, float\* B, float mul, int width) – Kernel używany do mnożenia Macierzy A przez wartość mul, i zapisujący jej zawartość do macierzy B. width oznacza szerokość macierzy.
   2. Matrix\_x\_matrix – Służy do mnożenia macierzy przez macierz. Obsługuje również macierze niebędące macierzami kwadratowymi. Zawiera 3 funkcje i jeden kernel.
      1. Void main() – Jest to główna funkcja programu, w niej deklarujemy pamięć, wywołujemy inne funkcje oraz kernele.
      2. Void printResults(float\*\* A, float\*\* B, float \*\* C, size\_t N, size\_t M, size\_t Csize) – Jest to funkcja użyta w celu wypisania elementów macierzy A oraz macierzy B, oraz macierzy C, gdzie N i M to rozmiary macierzy wejściowych, a Csize to rozmiar macierzy wynikowej.
      3. Void random\_ints(float\*\* matrix, size\_t N, size\_t M) - wypełnia macierz matrix pseudo losowymi liczbami, parametry N oraz M określają wymiary macierzy.
      4. \_\_global\_\_ void MatMul(float\* A, float\* B, float\* C, int ARows, int ACols, int BRows, int BCols, int CRows, int CCols) – Macierze A I B to macierze wejściowe, macierz C to macierz wynikowa, pozostałe parametry to ilość kolumn i rzędów w poszczególnych macierzach. Służy do wykonywania mnożenia macierzy A przez macierz B, wynik jest zapisywany do macierzy C.
   3. Matrix\_transpose – Służy do wykonywania transpozycji macierzy. Zawiera 3 funkcje i jeden kernel.
      1. Void main() – Jest to główna funkcja programu, w niej deklarujemy pamięć, wywołujemy inne funkcje oraz kernele.
      2. Void printResults(float\*\* A, float\*\* B, size\_t N, size\_t M) – Jest to funkcja użyta w celu wypisania elementów macierzy A oraz macierzy B, gdzie N to rozmiar macierzy wejściowej A, a M to rozmiar macierzy wyjściowej B.
      3. Void random\_ints(float\*\* matrix, size\_t N, size\_t M) - wypełnia macierz matrix pseudo losowymi liczbami, parametry N oraz M określają wymiary macierzy.
      4. \_\_global\_\_ void Matrix\_transpose(float\* A, float\* B, int A\_rows, int A\_cols) – Kernel używany do wykonywania transpozycji, Macierz A to macierz wejściowa, macierz B to macierz wyjściowa, a pozostałe dwa parametry to ilość kolumn i wierszy macierzy wejściowej.
   4. Matrix\_gauss – Służy do obliczenia rozwiązań zbioru równań za pomocą eliminacji gaussa. Składa się z 4 funkcji i jednego kernela. Do wykonania programu potrzebna jest przygotowana macierz w pliku tekstowym.
      1. Void main() – Jest to główna funkcja programu, w niej deklarujemy pamięć, wywołujemy inne funkcje oraz kernele.
      2. \_\_global\_\_ void Kernel(float\* A\_, float\* B\_, int size) – Kernel służący do obliczania metodą eliminacji gaussa.
      3. void copyvalue(int newchar, int\* i, FILE\* data, float\* temp\_h) – funkcja do kopiowania wartości z pliku do tabeli.
      4. void DeviceFunc(float\* temp\_h, int variablesNo, float\* temp1\_h) – zawiera wszystkie metody związane z alokowaniem pamięci w GPU, kopiowaniem z i do pamięci GPU, oraz wywoływaniem kernela.
      5. void getvalue(float\*\* temp\_h, int\* variablesNo) – funkcja służąca do odczytania danych z pliku.
   5. Matrix\_jordan – Służy do obliczania macierzy odwrotnej do macierzy wejściowej korzystając z metody gaussa-jordana. Funkcja ta zapisuje dwie wartości do dwóch różnych plików. W pliku input\_matrix.txt możemy znaleźć macierz wejściową, wygenerowaną w kodzie, a w pliku inverse.txt – obliczoną macierz odwrotną dla macierzy wejściowej.
      1. Void main() – Jest to główna funkcja programu, w niej deklarujemy pamięć, wywołujemy inne funkcje oraz kernele.
      2. \_\_global\_\_ void gaussjordan(float\* A, float\* I, int n, int i) – kernel wykonujący eliminacje gaussa-jordana w konkretnym wierszu
      3. \_\_global\_\_ void dev(float\* d\_A, float\* dI, int h) – Kernel odpowiedzialny za dzielenie wierszy zawierających element przestawny przez ten element przestawny.
      4. void savetofile(float\* A, string s, int n, int h) – funkcja odpowiedzialna za zapisanie macierzy A, o rozmiarze n na h, w pliku o nazwie zapisanej parametrze s.
      5. void random\_floats(float\* vect, int N) – funkcja odpowiedzialna za wypełnienie macierzy pseudo losowymi wartościami typu float.
4. **Problemy**
   1. We wstępnych założeniach projektu było również wykonanie kodu obliczającego równolegle macierz dopełnień i macierzy dołączonej dla dowolnej macierzy, niestety w związku problematyką liczenia wyznaczników macierzy wyższego stopnia - czyli rzeczy niezbędnej dla obliczania macierzy dopełnień i dołączonej, potrzebna jest duża ilość rekurencyjnych wywołań, a także wygenerowanie dla macierzy wyższego stopnia milionów niewielkich tymczasowych tabel.