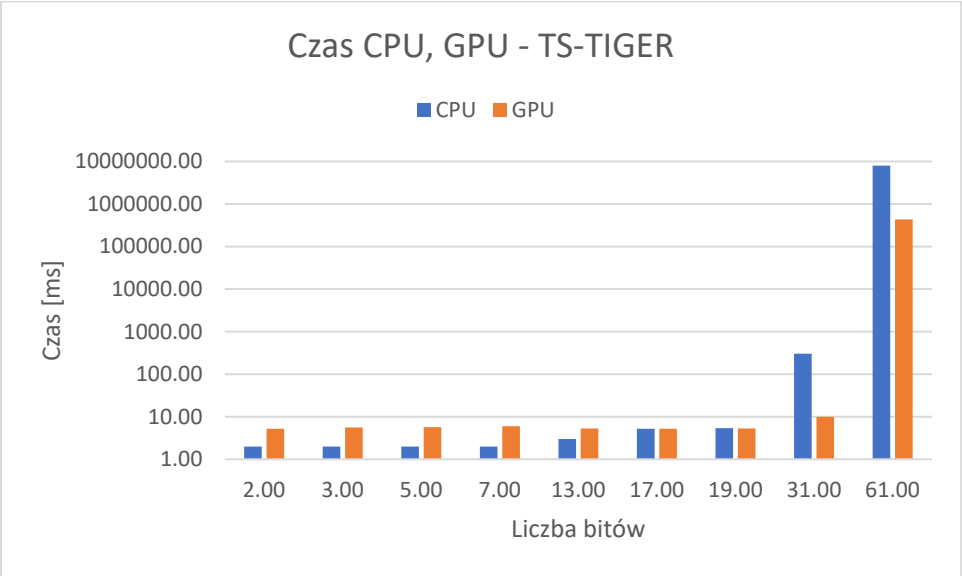
Współczesne Środowiska Programowania

Laboratorium 1 – Sprawdzanie pierwszości liczb

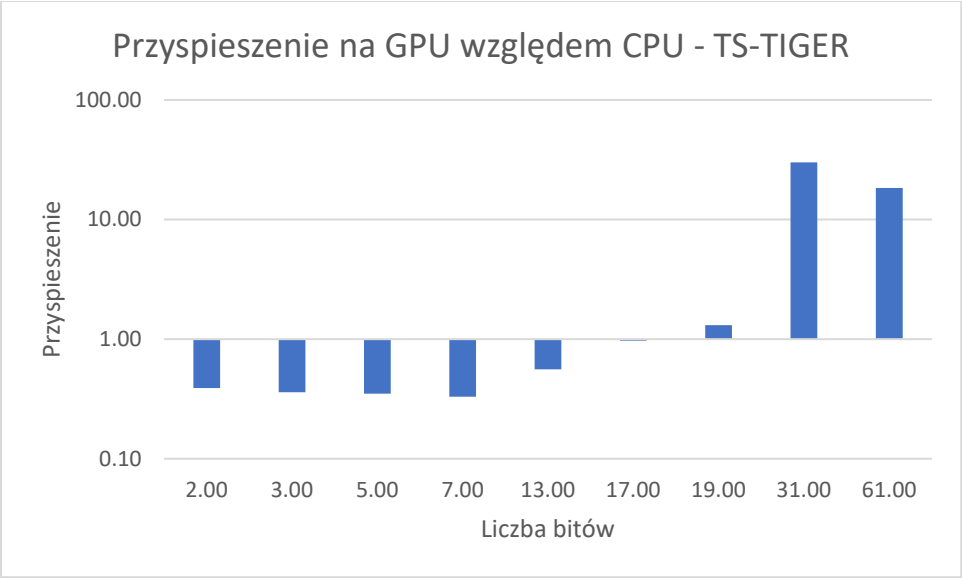
1. Celem laboratorium było zapoznanie się z programowaniem z wykorzystaniem kart graficznych firmy Nvidia i napisanie przykładowego programu sprawdzającego, czy dana liczba jest pierwsza, czy złożona oraz porównanie czasu wykonywania tego sprawdzania na CPU oraz GPU.

Pomiary zostały wykonywane na komputerze ts-tiger, na karcie graficznej Nvidia GTX 660 oraz na domowym komputerze z kartą GTX 1050 Ti.

2. Porównanie szybkości sprawdzania czy liczba jest pierwsza na CPU i GPU, w zależności od liczby bitów na Nvidia GTX 660:

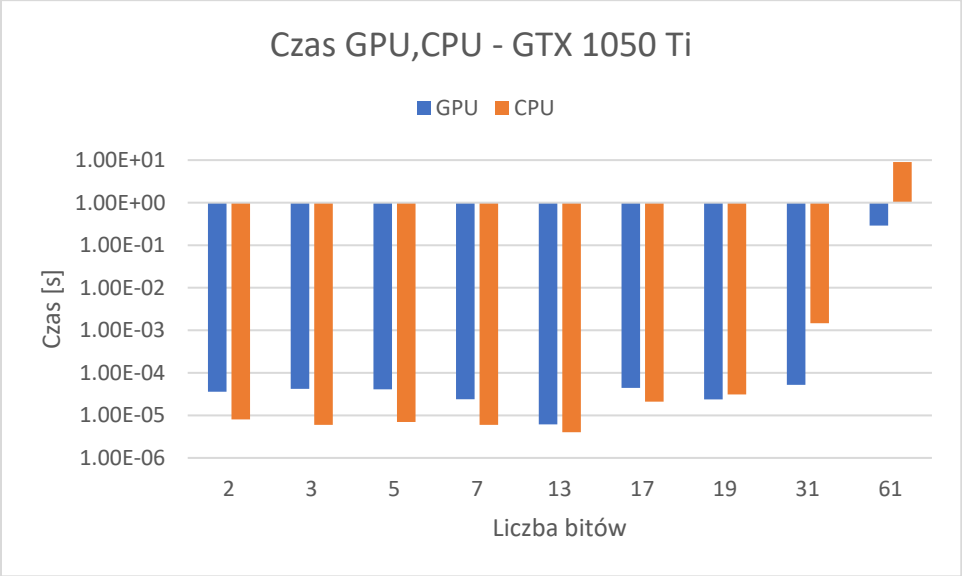


Z wykresu łatwo wyczytać, że sprawdzanie czy liczba jest pierwsza na GPU zaczyna być opłacalne dopiero od liczby 17 bitowej. Dla liczb o mniejszej ilości bitów algorytm jest znacznie wolniejszy, dzieje się tak, ponieważ przesyłanie danych na kartę graficzną zajmuje więcej czasu niż samo przetworzenie danych. Można wliczyć w to też samą inicjalizację kernela.



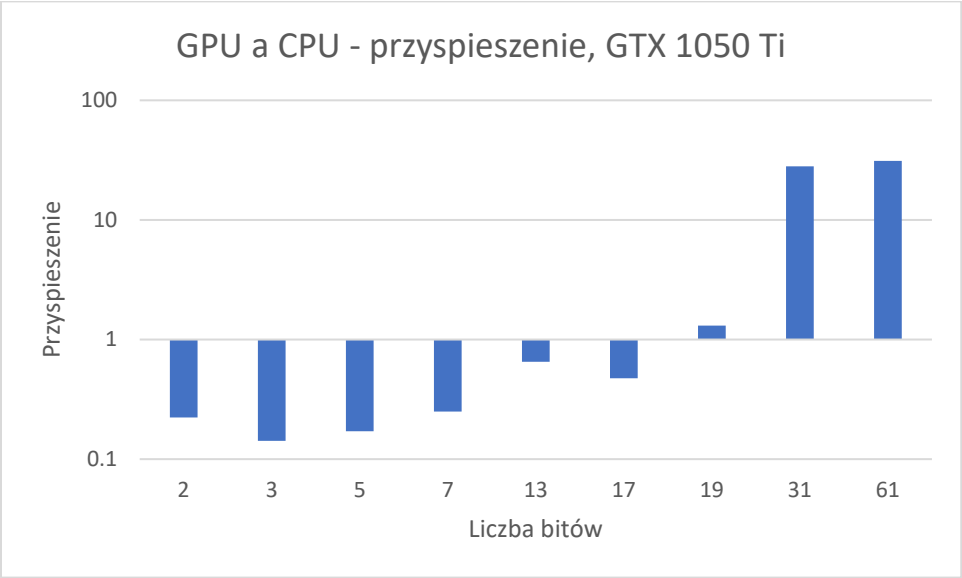
Podobnie do poprzedniego wykresu, tutaj również obserwujemy przyspieszenie dla liczb większych niż 17 bitowe podczas określania czy liczba jest pierwsza, czy złożona. Moglibyśmy spodziewać się większego przyspieszenia dla liczby 61 bitowej, jednak mniejsze przyspieszenie w porównaniu do 31 bitowej liczby może oznaczać, że jest jeszcze miejsce na optymalizację programu. Zastosowany algorytm jest jednym z najprostszych, jednak widać, że jeśli operujemy na większej ilości danych, możemy zacząć zastanawiać się nad optymalizacją w środowisku CUDA.

3. Porównanie szybkości sprawdzania czy liczba jest pierwsza na CPU i GPU, w zależności od liczby bitów na Nvidia GTX 1050 Ti:



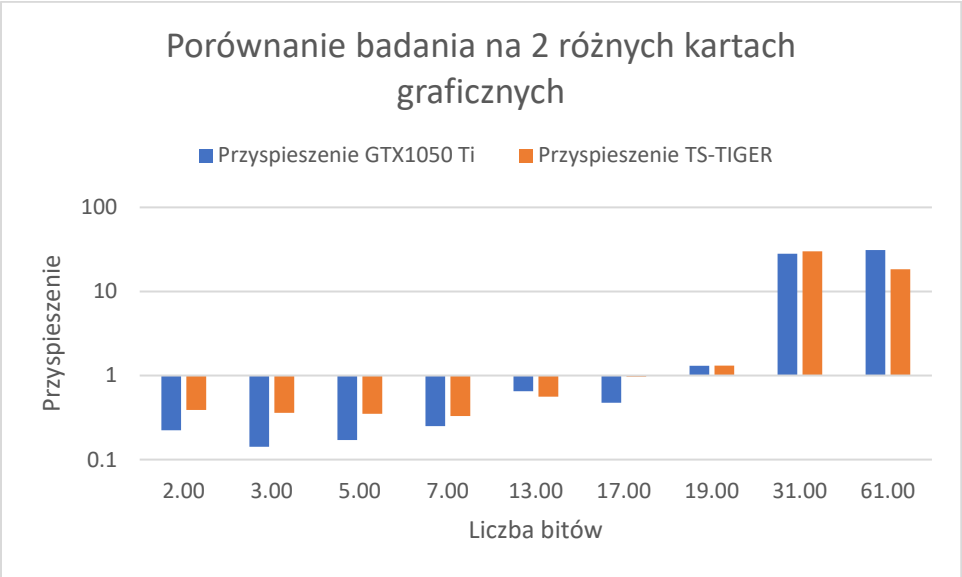
Sprzęt użyty do uruchomienia programu jest nowszy i bardziej wydajniejszy, z tego powodu obserwujemy w tym wypadku o wiele krótsze czasy zarówno dla GPU jak i CPU. Zaskakująco tym razem dopiero od liczby 31 bitowej obserwujemy znaczny wzrost przyspieszenia. Właśnie z powodu lepszego sprzętu ma to sens – podczas pisania programów dla kart

graficznych powinniśmy zwracać uwagę na inne dostępne dla nas narzędzia. Nie zawsze będziemy potrzebować kart graficznych do złożonych obliczeń, kiedy posiadamy odpowiednio szybkie CPU.



Po zorganizowaniu danych w inny sposób, okazuje się, że przyspieszenie dla karty GTX 1050 Ti zaczyna się właściwie od liczby 19 bitowej. Tym razem w porównaniu do GTX 660 w stosunku do liczby 61 bitowej, liczba 31 bitowa ma nieznacznie mniejsze przyspieszenie. Może to wynikać z innej architektury tej konkretnej karty graficznej.

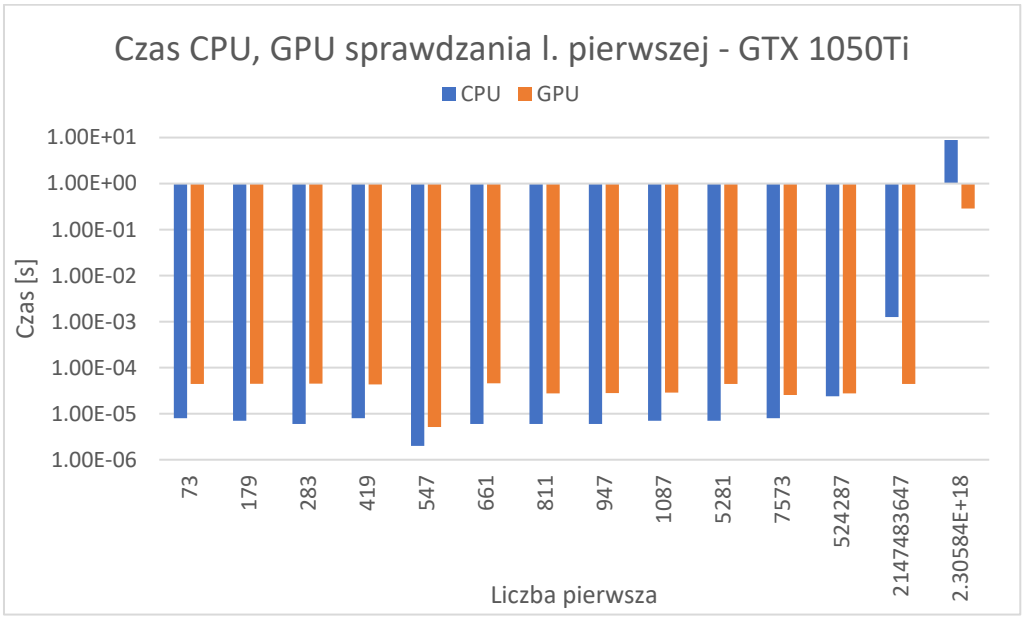
4. Porównanie osiągnieć GTX 660 i GTX 1050 Ti:



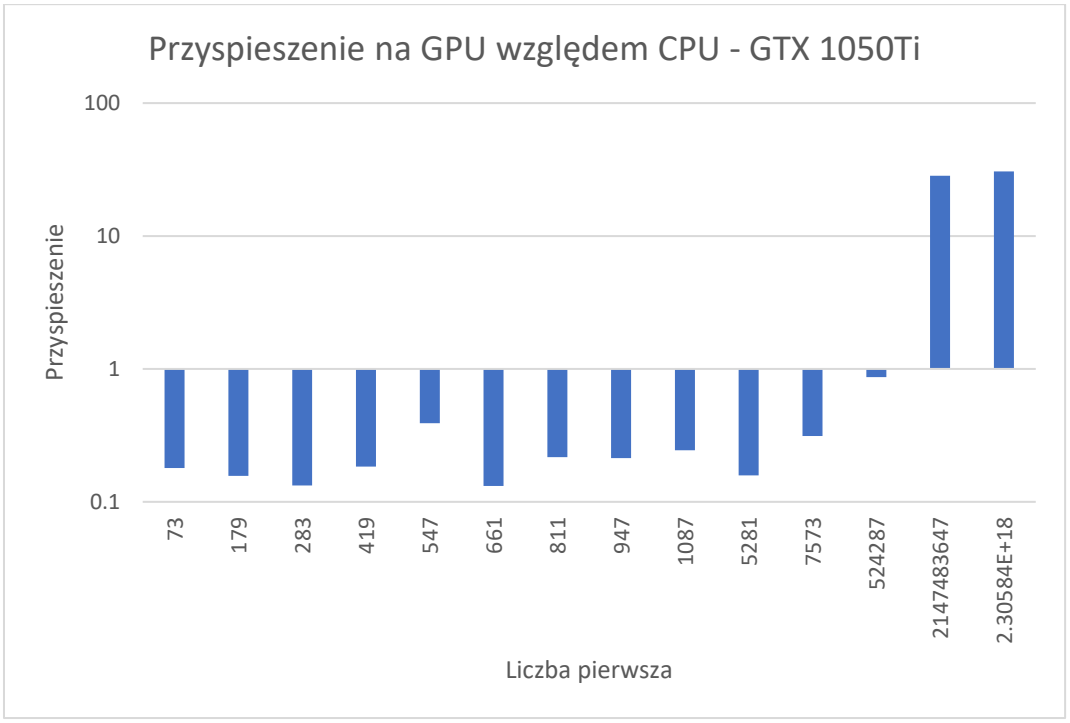
Łatwo zauważyć, że dla mniejszych liczb bitów GTX 660 właściwie osiąga lepsze przyspieszenie, wciąż wynika to z faktu różnicy w budowie obu kart i ich nieprzystosowania

do mniejszych ilości danych. Nowsze karty są tworzone z myślą o jak największych osiągach na ogromnych datasetach i możliwe, że z tego powodu operują gorzej na tych mniejszych.

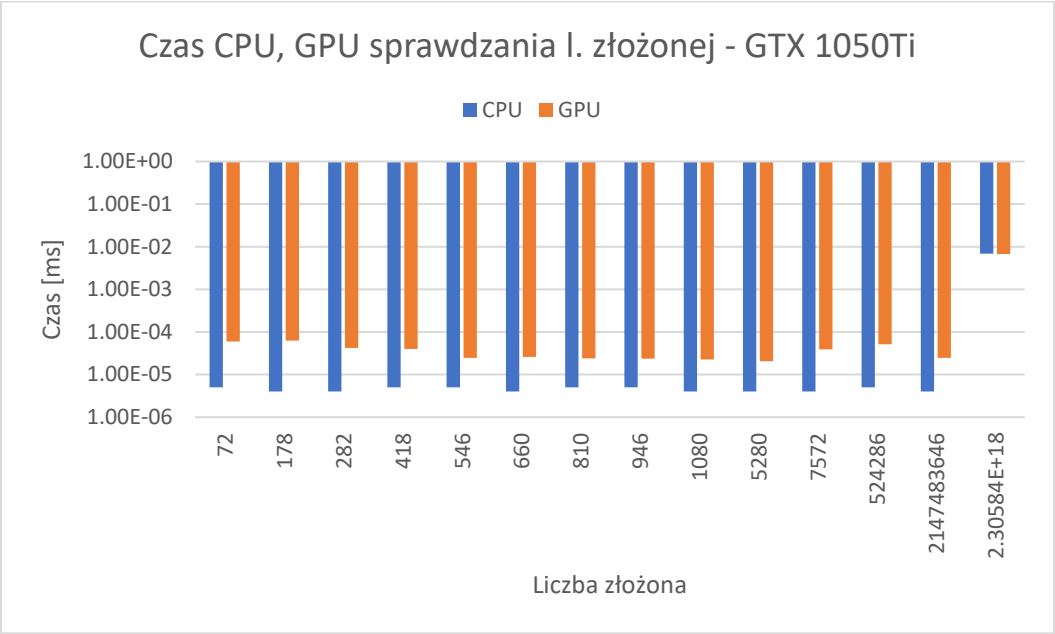
5. Czasy sprawdzania wybranych liczb pierwszych na CPU i GPU (GTX 1050 Ti):



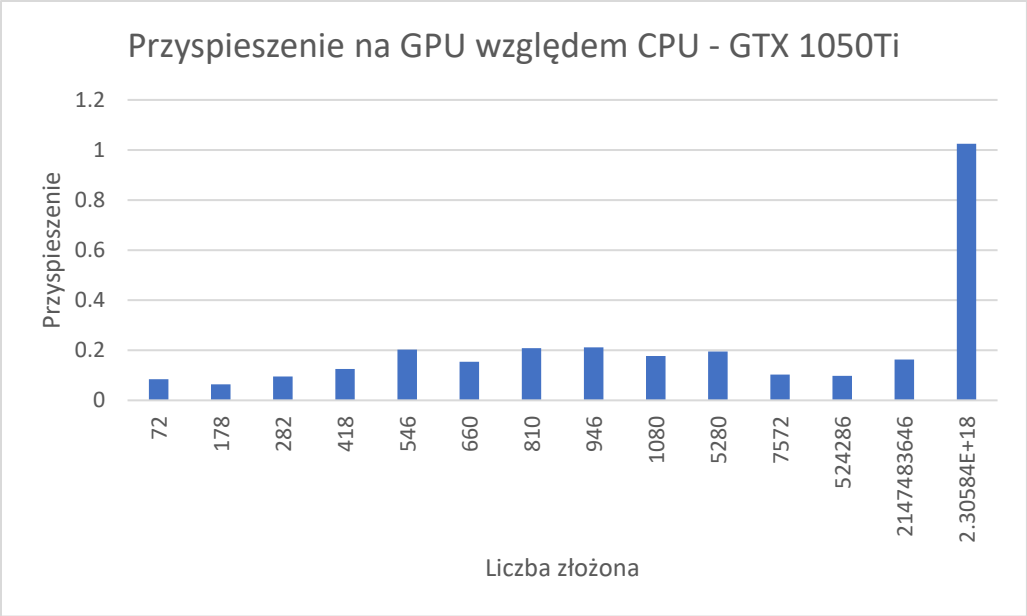
Czasy oczywiście są niewielkie, dlatego tak jak na poprzednich wykresach, tutaj również została zastosowana skala logarytmiczna. Znowu obserwujemy ten sam przypadek, który potwierdza wcześniejsze wnioski – większa liczba danych oznacza więcej korzyści z używania karty graficznej.



6. Czasy sprawdzania wybranych liczb złożonych na CPU i GPU (GTX 1050 Ti):



W porównaniu do sprawdzania liczb pierwszych, sprawdzanie liczb złożonych odbywa się szybciej zarówno dla CPU jak i GPU. Dzieje się tak prawdopodobnie, ponieważ algorytm jest dopasowany do znajdywania liczb pierwszych i jest to trudniejsza operacja niż uznanie liczby za złożoną.



Z tego też powodu przyspieszenie na GPU względem CPU nie jest na tyle zauważalne co w pozostałych przypadkach.

7. Napotkane trudności:

‘Przestawienie się’ na programowanie równoległe było dla mnie największą trudnością, mimo wiedzy w jaki sposób działa pisanie programu na karcie graficznej, bardzo długo nie potrafiłem wymyślić stosownego sposobu na zrównoleglenie algorytmu. Wybranie odpowiedniej liczby bloków i wątków również sprawiło mi problemy, jednak ostatecznie uważam, że dobrze poradziłem sobie z napotkanymi problemami.