**고급소프트웨어실습 11주차 보고서**

**20191657 최세은**

이 보고서에서 모든 실행 시간은 release 모드에서 정확도를 위해 약 5번씩 반복해 측정하고, 이의 평균값을 사용하였다.

**실습 2**

|  |  |  |
| --- | --- | --- |
| **block size** | **cpu time** | **gpu time** |
| 32 | 34.294120 | 0.082156 |
| 64 | 35.014951 | 0.082450 |
| 128 | 36.059423 | 0.0940042 |

cpu보다 gpu time이 훨씬 적게 걸리는 것을 확인할 수 있었다. 그리고 거의 차이가 없긴 하지만 block size가 커질수록 cpu와 gpu time이 더 커지는 것을 확인할 수 있었다.

그리고 연산하는 코드를 더 복잡하게 하여 측정했더니 다음과 같은 결과를 얻을 수 있었다.

|  |  |  |
| --- | --- | --- |
| **block size** | **cpu time** | **gpu time** |
| 32 | 35.815000 | 0.085315 |
| 64 | 35.904825 | 0.086005 |
| 128 | 36.001281 | 0.086998 |

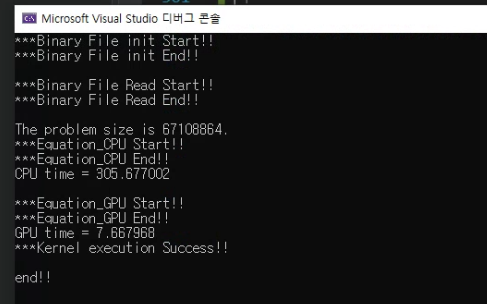
이전 코드에 비해 전체적으로 시간이 증가했지만 상대적인 크기는 이전 코드와 같았다.

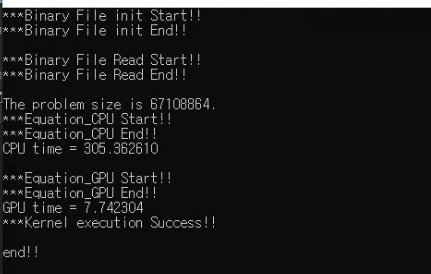
**실습 3**

|  |  |  |
| --- | --- | --- |
| **block size** | **cpu time** | **gpu time** |
| 16 | 0.642158 | 0.196548 |
| 32 | 0.643074 | 0.887142 |
| 64 | 0.652802 | 0.894004 |

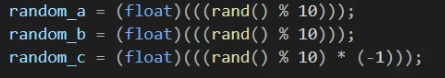
실습 2와 마찬가지로 대체로 block size가 증가하면 cpu와 gpu time 둘 다 증가하는 모습을 보여주었다. 그러나 이상하게도 예상과 달리 block size가 16일 때를 제외하면 gpu time이 cpu time보다 컸는데 결과를 보면 gpu 쪽에서 문제가 있는 것으로 보인다.

**과제 1**

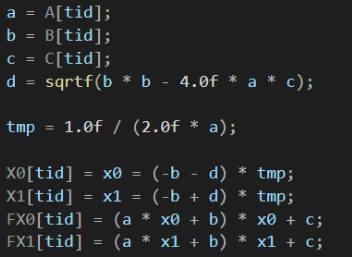
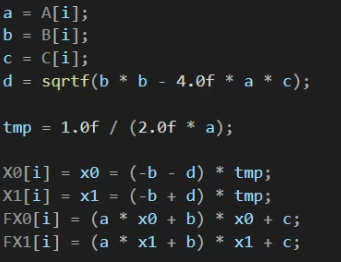




[기본 실행 결과]



우선 이차방정식이 두 개의 실근을 무조건 가지게 하기 위해서 랜덤하게 a, b, c를 생성했다. 그리고 x0 <= x1이 만족시키기 위해 다음과 같은 코드를 구현했다. 각각 cpu와 gpu 쪽의 코드이다.

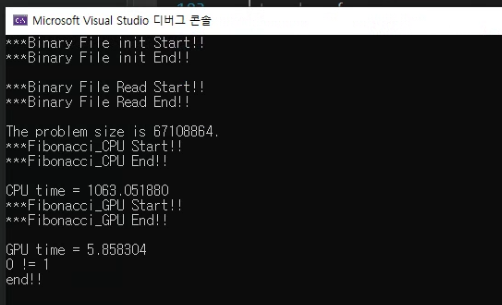


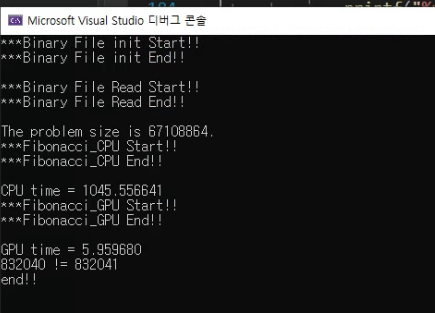
또한 수치 계산 시 계산 오차가 발생할 수 있기 때문에 이를 고려해 둘의 차이가 0.00001보다 작으면 알맞게 결과를 낸 것으로 간주했다.

|  |  |  |
| --- | --- | --- |
| **block size** | **cpu time** | **gpu time** |
| 32 | 305.621542 | 7.710532 |
| 64 | 305.561255 | 7.760024 |
| 128 | 305.412001 | 7.521681 |

block size를 바꿔가며 실행해 본 결과 cpu나 gpu time이나 거의 차이가 없었고 그나마 128일 때 근소하게 빠른 실행 시간을 보여주었다. 그리고 항상 cpu time이 gpu time 보다 훨씬 느리게 실행되었는데, 이는 이차방정식들의 해와 해에 따른 f(x)값을 구하는 연산을 cpu에서는 순차적으로 계산을 하지만 gpu에서는 병렬적으로 처리를 하기 때문이다.

**과제 2**

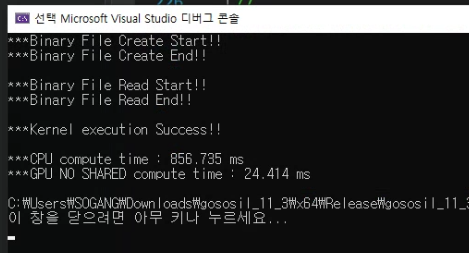


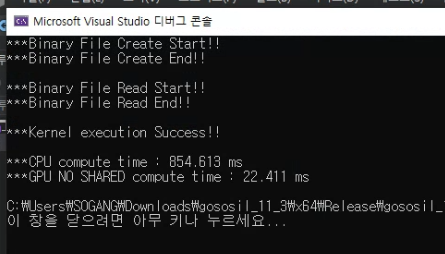


|  |  |  |
| --- | --- | --- |
| **block size** | **cpu time** | **gpu time** |
| 32 (기본) | 1065.002456 | 5.902365 |
| 64 | 1047.215627 | 3.466150 |
| 128 | 1046.056234 | 3.417525 |

우선 cpu와 gpu time을 비교했을 때 확연히 gpu time이 더 적게 나옴을 확인할 수 있다. 이는 cpu 에서는 피보나치 수를 구하는 연산을 한 개씩 계산하지만 gpu(cuda)에서는 병렬적으로 처리하기 때문이다. 또한 block size에 따라 걸리는 시간이 달라짐을 위 표로 확인할 수 있다.

**과제 3**





(1) N = 2^26

|  |  |
| --- | --- |
| **Nf** | **cpu time** |
| 1 | 364.258 |
| 4 | 510.300 |
| 16 | 544.163 |
| 64 | 884.041 |
| 256 | 2751.694 |
| 1024 | 10531.811 |

block size = 32

|  |  |
| --- | --- |
| **Nf** | **gpu time** |
| 1 | 6.551 |
| 4 | 6.806 |
| 16 | 9.252 |
| 64 | 26.011 |
| 256 | 100.058 |
| 1024 | 273.829 |

block size = 64

|  |  |
| --- | --- |
| **Nf** | **gpu time** |
| 1 | 3.346 |
| 4 | 3.215 |
| 16 | 6.299 |
| 64 | 22.078 |
| 256 | 83.920 |
| 1024 | 281.072 |

block size = 128

|  |  |
| --- | --- |
| **Nf** | **gpu time** |
| 1 | 2.534 |
| 4 | 3.098 |
| 16 | 5.455 |
| 64 | 21.041 |
| 256 | 89.370 |
| 1024 | 322.843 |

(2) N = 2^24

|  |  |
| --- | --- |
| **Nf** | **cpu time** |
| 1 | 70.102 |
| 4 | 115.004 |
| 16 | 125.946 |
| 64 | 207.810 |
| 256 | 675.984 |
| 1024 | 2628.462 |

block size = 32

|  |  |
| --- | --- |
| **Nf** | **gpu time** |
| 1 | 1.591 |
| 4 | 1.586 |
| 16 | 2.506 |
| 64 | 6.822 |
| 256 | 20.486 |
| 1024 | 62.918 |

block size = 64

|  |  |
| --- | --- |
| **Nf** | **gpu time** |
| 1 | 0.841 |
| 4 | 0.855 |
| 16 | 1.469 |
| 64 | 5.094 |
| 256 | 17.942 |
| 1024 | 69.058 |

block size = 128

|  |  |
| --- | --- |
| **Nf** | **gpu time** |
| 1 | 0.782 |
| 4 | 0.786 |
| 16 | 1.394 |
| 64 | 4.501 |
| 256 | 17.921 |
| 1024 | 78.069 |

우선 나머지 조건이 같을 때 Nf가 커질 때마다 cpu, gpu time이 점점 증가한다. 이는 Nf가 커질 때마다 원소를 포함하는 윈도우가 커져 더 많은 연산이 수행되었기 때문이다. 또한 block size와 Nf를 가지고 결과를 비교하면 Nf가 256, 1024일 때는 block size가 128일 때 가장 오래 걸렸고 Nf가 1024일 때를 제외하고 block size가 128일 때 가장 짧게 걸려서 각각 극대, 극소값임을 알 수 있었다.