어셈블리 설계 및 실습 결과 보고서

Bilinear Interpolation

학　　과: 컴퓨터공학과

담당교수: 이준환 교수님

학　　번: 2015722025

성　　명: 정용훈

결과 보고서 목차

1. Introduction

1) Simple introduction of project

Project의 간단한 설명으로는 N값의 N X N 정방행렬을 2^k만큼 확대하는 작업이 가능한 프로그램을 어셈블리언어를 통하여 구현하는 것이 목적입니다. 해당 프로그램의 응용으로는 영상이나 이미지 처리를 하는데 쓰이며, 해당프로그램으로는 영상이나 이미지 확대를 가능할 수 있도록 도와주는 프로그램입니다. 프로그램을 작성할 수 있는 방법을 Project를 구현하기전 배우게 되었는데 개념을 살펴보면 선형 보간 법과 양선형 보간 법이 있습니다. 하지만 해당 방법을 사용하지 않고 중점 정리와 점 사이의 거리를 직접 구하여 더하는 방법을 채택하여 Project를 구현하게 되었습니다.

2) Schedule

11	12	13	14	15	16	17
				제안서 제출	프로젝트의 대한 알고리즘 정리	Project ver1 구현
18	19	20	21	22	23	24
Project ver1 구현	Project ver2 구현		Project ver3 구현		Project ver3 최적화	
25	26	27	28	29	30	
보고서 기본 틀 작성	결과 보고서 작성			프로젝트 제출		

해당 일정표는 이번 프로젝트를 구현하고 제출하는 과정을 간단하게 나타낸 이미지입니다. 제안서 제출이후 알고리즘과 k, n값의 대한 알고리즘을 생각했으며 바로 고안한 방법을 토대로 프로젝트를 구현하게 되었습니다. Project version 1에서는 중점 정리를 이용하여 Project를 구현하였으며, Project version 2는 점과 점사이의 거리를 구하여 k값에 따라 거리를 균등하게 나누어 주어 계속해서 더해주어 사이 값을 구할 수 있는 알고리즘입니다. Project version 3은 중점 정리를 토대로 기존 code를 좀 더 Unrolling하게 바꾸어 code size를 늘리고 state를 줄이는 방법을 채택하게 되었습니다.

2. Project Specification

Pseudo code

----------Row function----------

(1) Matrix data주소 load, 계산된 값을 저장할 주소 load, 각 count를 저장할 주소 load

(2) n값과 k값 Load

(3) 계산할 값을 r1, r2에 load

(4) Floating point 계산 실행(해당 Pseudo code는 전 실습에서 수행함)

(5) Save branch호출, r1, r2의 값과 계산된 mid-point의 값 저장

(6) row한줄이 모두 계산 완료되었는지 판단, 아니라면 3번 문항으로 맞으면 7번으로

(7) Padding branch를 call할지 Nonpadding을 call 할지 k값의 따라 결정

(8) 모든 row가 계산을 완료했는지 판단, 완료되었다면 9번, 아니면 3번 문항 호출

(9) k비교 후 1이면 column함수로 넘어가고 2이면 row함수 한번 더 실행, 3이면 두 번 더 실행

----------**Column function**----------

(10) Row함수를 통과 후 저장된 data주소 load, 결과를 저장할 주소 load

(11) Column계산을 위하여 다음 데이터를 가리킬 수 있도록 next pointer의 값을 지정해줌 (next pointer를 위한 이동 횟수 = N * 2^k)

(12) 계산할 값을 r1, r2에 load

(13) Floating Point 계산 실행

(14) Save branch호출, r1, r2의 값과 mid-point의 값을 계산한 next pointer 만큼의 값만큼 이동 후 각각의 값 저장

(15) column한줄이 모두 계산이 완료되었는지 판단, 아니면 12번 문항으로 이동 맞으면 16번 문항으로 이동

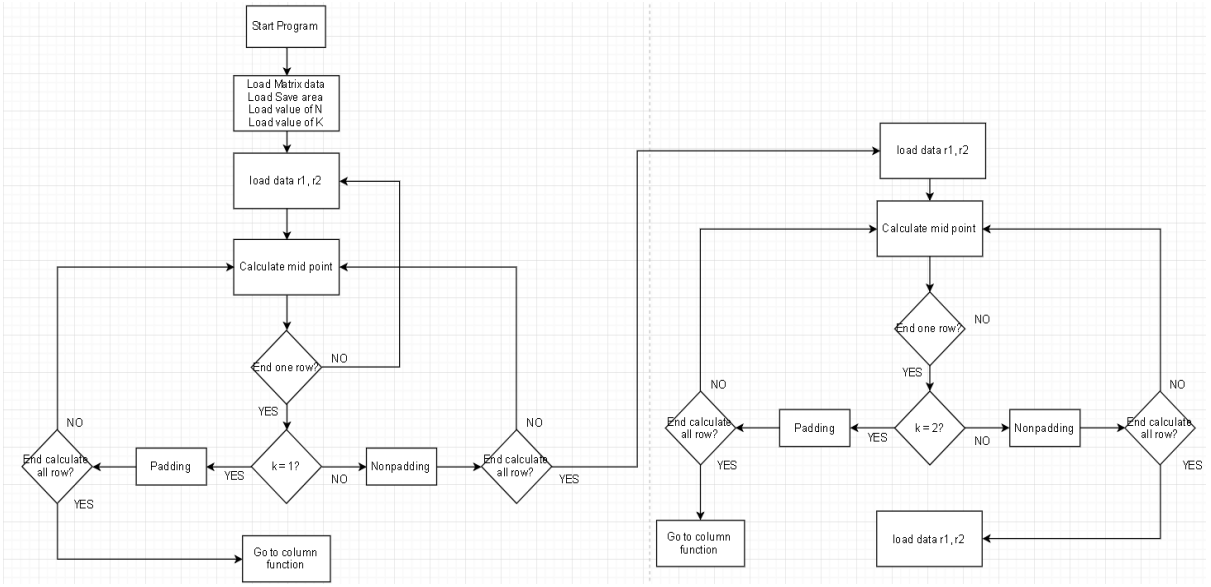
(16) k값에 따라 padding함수를 call할지 Nonpadding함수를 call할지 결정

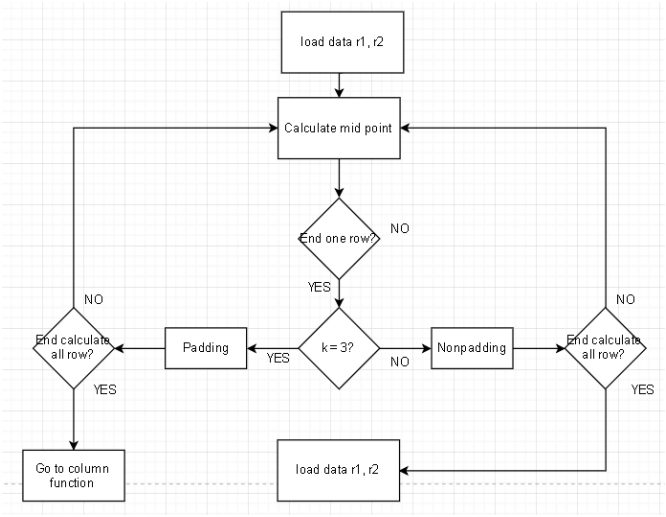
(17) 모든 column이 계산이 완료되었는지 판단, 완료되었다면 프로그램을 종료, 아니면 18번으로 이동

(18) k값의 따라 Column function의 반복수행 횟수를 결정 (k=1, 이면 17번에서 종료, k=2이면 한번 더 반복 후 17번에서 종료, k=3이면 2번 더 반복 후 17번에서 종료)

Flow chart

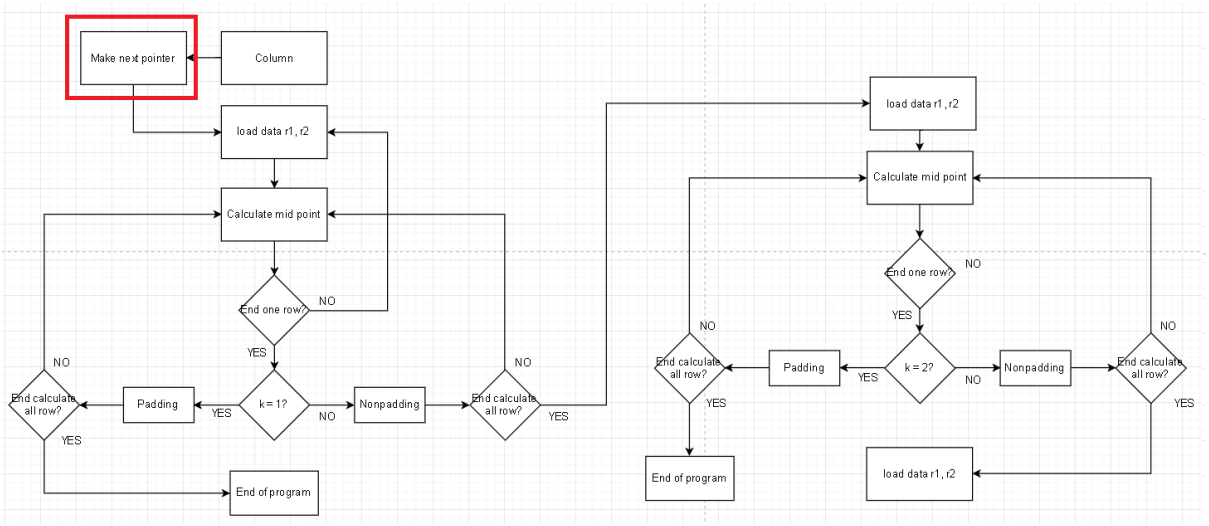
(row function)

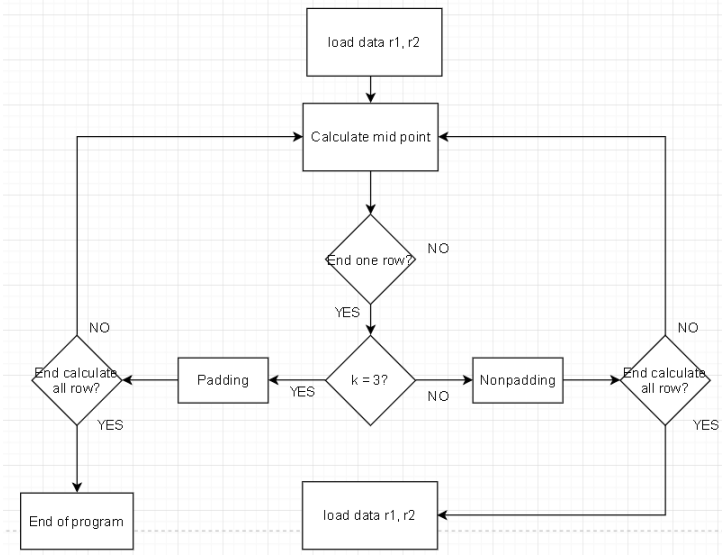




해당 Flow chart는 row를 구하는 함수로 k의 값의 따라 column함수로 넘어가는 타이밍이 다르게 됩니다. K는 1일 때 왼쪽 위 함수만 실행하고 column함수로 넘어가게 되며 2이면 오른쪽위에서 3이면 해당 그림에 있는 함수들을 모두 수행 후 column함수로 넘어가게 됩니다.

(column)

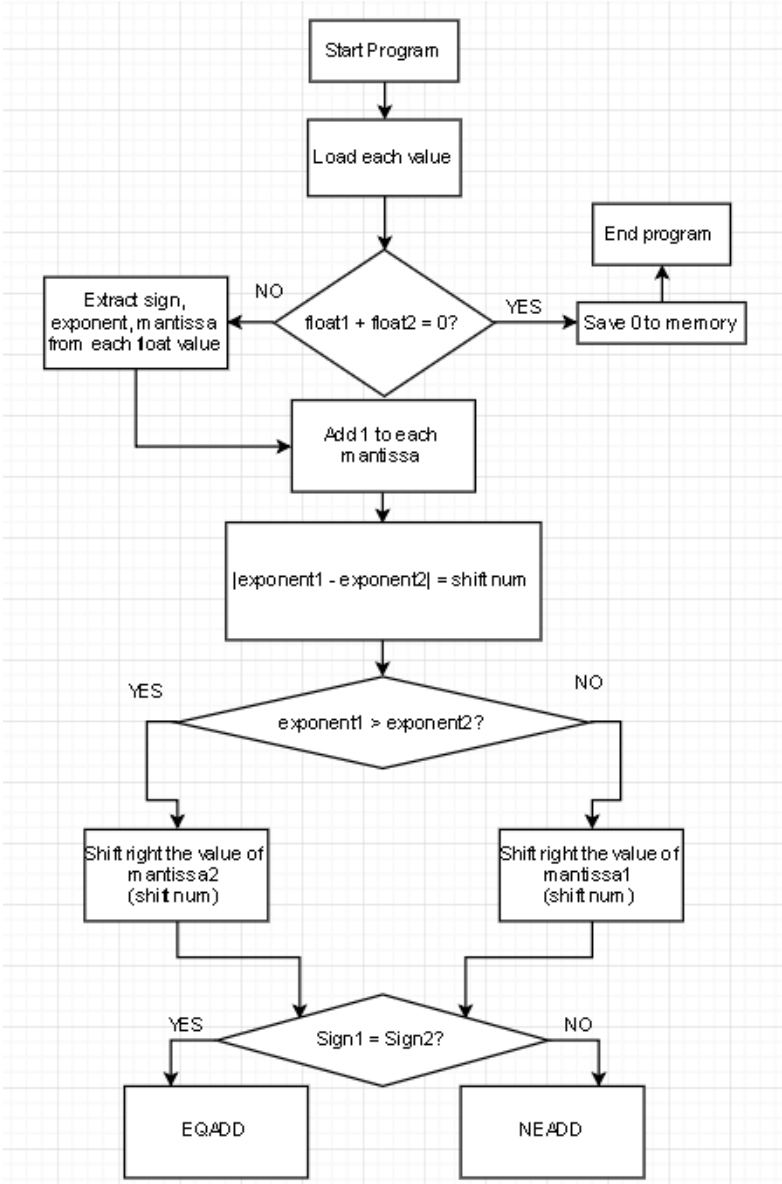




전체적인 흐름은 Row flowchart와 다르지 않습니다. 한가지 다른 부분은 불러오는 주소 값이 4차이가 아니라 column이기 때문에 N의 값을 왼쪽으로 K만큼 shift해준 값의 횟수가 다음 column값을 가리키기 때문에 위 flow chart에서 next pointer를 가리킬 수 있도록 하는 값을 만들어 주어야 합니다. 그 후 전체 알고리즘은 모든 동작이 끝났을 경우 프로그램을 종료해주는 조건이 있을 뿐 다르지 않습니다.

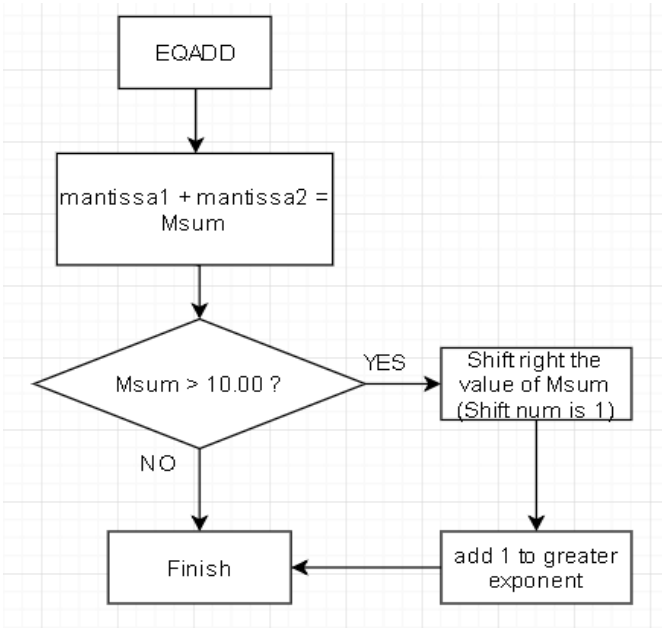
3. Algorithm

Floating Point

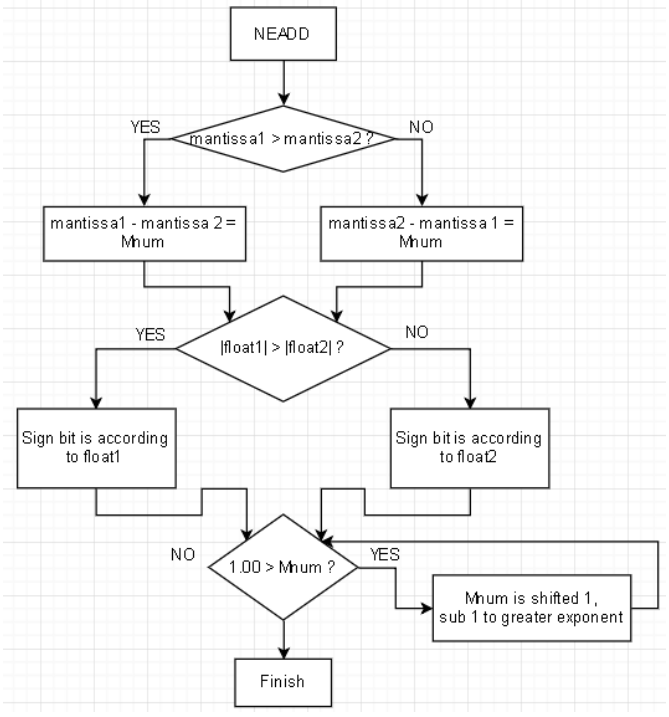


해당 flow chart는 floating point계산의 대한 알고리즘입니다. 해당 chart를 수행한 다음 부호가 같을 때와 부호가 다른 경우로 나뉘게 되는데 그 경우의 함수는 다음 그림과 같습니다.

EQADD

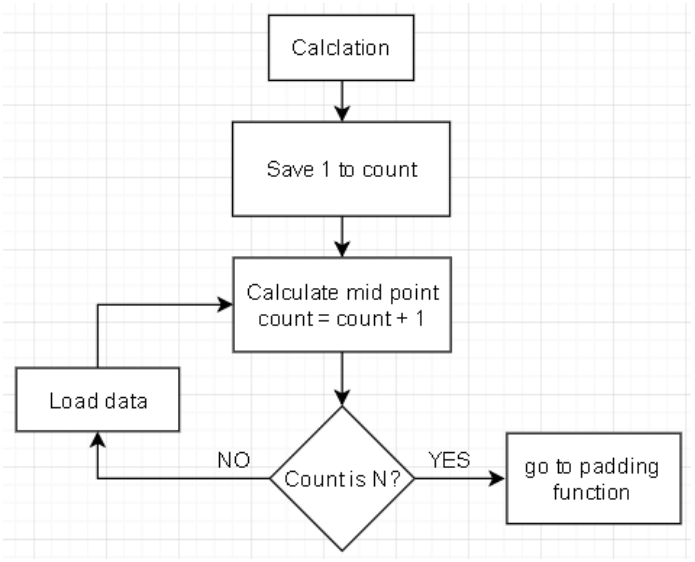


NEADD

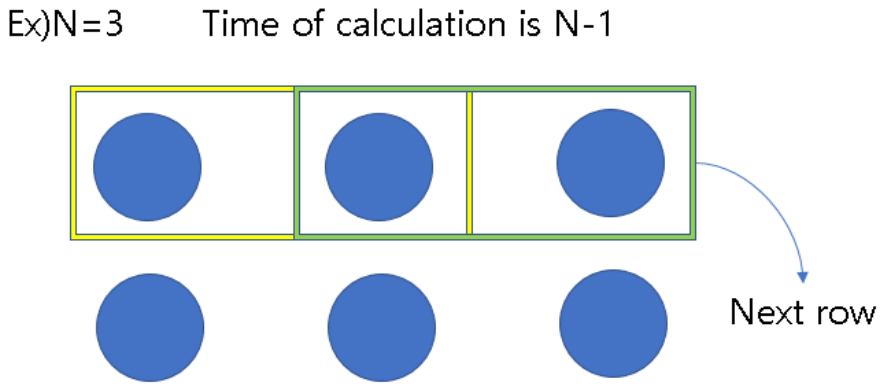


위 그림은 계산하는 값의 부호가 같은 경우와 다른 경우를 나눈 것이다. 위 공통적인 flow chart를 지난 후 부호에 따라 해당 함수를 통과하면 최종적인 덧셈 값이 나오게 된다. 최종 값이 나오기 전에 exponent에서 1값을 빼면 두 값 사이의 중점이 나오므로 해당 알고리즘을 이용해서 중점을 구할 수 있다.

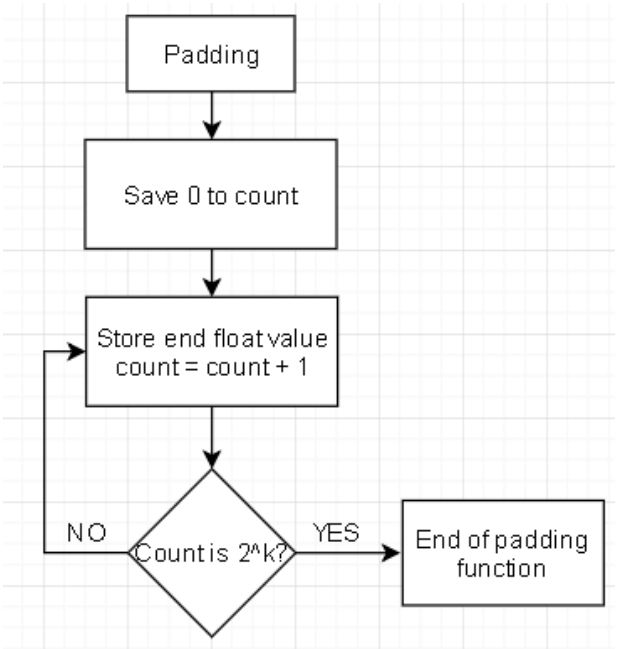
Calculation



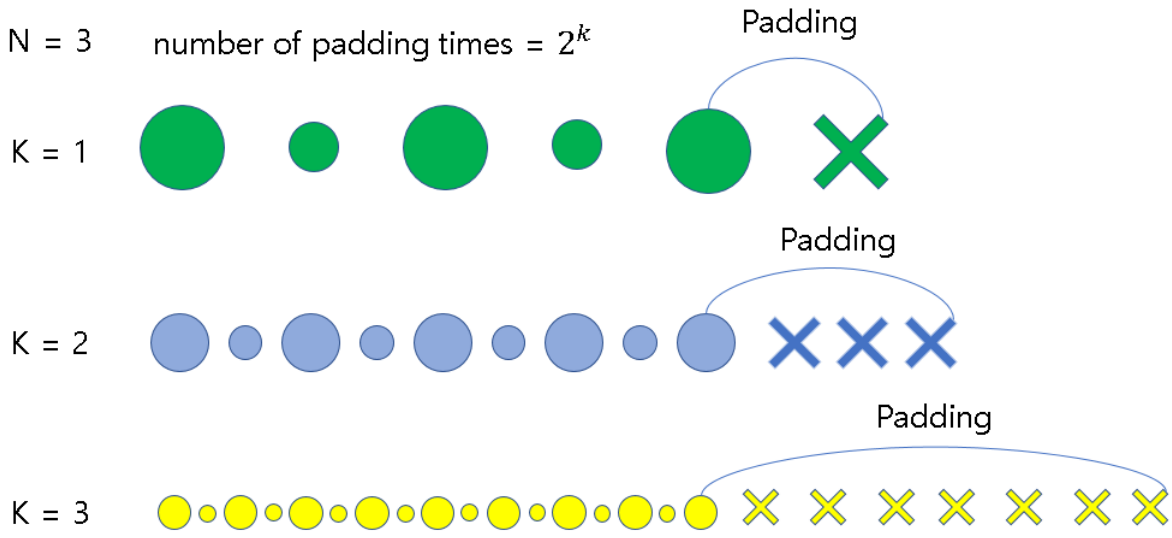
해당 함수는 data의 한 줄을 계산하는 알고리즘입니다. 한 줄의 mid point를 모두 계산 후 padding을 하는 조건으로 계산을 N-1번만큼 한 후 padding함수로 넘어가 row나 column의 한 줄을 padding할 수 있도록 하는 함수입니다. 단 k의 값이 다르면 Non padding함으로써 다음 계산의 Base가 될 수 있도록 해줍니다.



Padding



Padding함수를 실행하기 전 위 Calculation함수를 거친 후 조건을 만족하면 Padding함수로 넘어오게 됩니다. K의 값이 다르게 되면 padding을 하지 않고 넘어가게 됩니다.



또한 row padding함수의 횟수가 N번 실행되었다면 column을 구하기 위한 함수로 넘어가며, column을 구하는 방법은 row를 구하는 방법과 동일합니다.

4. Performance & Result

Input 1

N=3, K=1

Address: 0x60000000

0x60000000: 110	179	248	116	-16	-16
0x60000018: 13	65.125	117.25	55.6094	-6.03125	-6.03125
0x60000030: -84	-48.75	-13.5	-4.78125	3.9375	3.9375
0x60000048: -42.1758	-9.46289	23.25	-179.391	-382.031	-382.031
0x60000060: -0.351563	29.8242	60	-354	-768	-768
0x60000078: -0.351563	29.8242	60	-354	-768	-768
0x60000090: 0	0	0	0	0	0
0x600000A8: 0	0	0	0	0	0
0x600000C0: 0	0	0	0	0	0

Input 2

N=3, K=2

Address: 0x60000000

0x60000000: 58	43.0234	28.0469	13.0703	-1.90625	-2.92969
0x60000018: -3.95313	-4.97656	-6	-6	-6	-6
0x60000030: 15.5	19.2676	23.0352	26.8027	30.5703	24.4902
0x60000048: 18.4102	12.3301	6.25	6.25	6.25	6.25
0x60000060: -27	-4.48828	18.0234	40.5352	63.0469	51.9102
0x60000078: 40.7734	29.6367	18.5	18.5	18.5	18.5
0x60000090: -69.5	-28.2441	13.0117	54.2676	95.5234	79.3301
0x600000A8: 63.1367	46.9434	30.75	30.75	30.75	30.75
0x600000C0: -112	-52	8	68	128	106.75
0x600000D8: 85.5	64.25	43	43	43	43
0x600000F0: 14	49.5	85	120.5	156	126.313
0x60000108: 96.625	66.9375	37.25	37.25	37.25	37.25
0x60000120: 140	151	162	173	184	145.875
0x60000138: 107.75	69.625	31.5	31.5	31.5	31.5
0x60000150: 266	252.5	239	225.5	212	165.438
0x60000168: 118.875	72.3125	25.75	25.75	25.75	25.75
0x60000180: 392	354	316	278	240	185
0x60000198: 130	75	20	20	20	20
0x600001B0: 392	354	316	278	240	185
0x600001C8: 130	75	20	20	20	20
0x600001E0: 392	354	316	278	240	185
0x600001F8: 130	75	20	20	20	20
0x60000210: 392	354	316	278	240	185
0x60000228: 130	75	20	20	20	20
0x60000240: 0	0	0	0	0	0
0x60000258: 0	0	0	0	0	0
0x60000270: 0	0	0	0	0	0

Input3

N=2, K=3

Address: 0x60000000

0x60000000:	2.03125	0.902344	-0.226563	-1.35547	-2.48438	-3.61328
0x60000018:	-4.74219	-5.87109	-7	-7	-7	-7
0x60000030:	-7	-7	-7	-7	34.2773	29.4927
0x60000048:	24.708	19.9233	15.1387	10.354	5.56934	0.784668
0x60000060:	-4	-4	-4	-4	-4	-4
0x60000078:	-4	-4	66.5234	58.083	49.6426	41.2021
0x60000090:	32.7617	24.3213	15.8809	7.44043	-1	-1
0x600000A8:	-1	-1	-1	-1	-1	-1
0x600000C0:	98.7695	86.6733	74.5771	62.481	50.3848	38.2886
0x600000D8:	26.1924	14.0962	2	2	2	2
0x600000F0:	2	2	2	2	131.016	115.264
0x60000108:	99.5117	83.7598	68.0078	52.2559	36.5039	20.752
0x60000120:	5	5	5	5	5	5
0x60000138:	5	5	163.262	143.854	124.446	105.039
0x60000150:	85.6309	66.2231	46.8154	27.4077	8	8
0x60000168:	8	8	8	8	8	8
0x60000180:	195.508	172.444	149.381	126.317	103.254	80.1904
0x60000198:	57.127	34.0635	11	11	11	11
0x600001B0:	11	11	11	11	227.754	201.035
0x600001C8:	174.315	147.596	120.877	94.1577	67.4385	40.7192
0x600001E0:	14	14	14	14	14	14
0x600001F8:	14	14	260	229.625	199.25	168.875
0x60000210:	138.5	108.125	77.75	47.375	17	17
0x60000228:	17	17	17	17	17	17
0x60000240:	260	229.625	199.25	168.875	138.5	108.125
0x60000258:	77.75	47.375	17	17	17	17
0x60000270:	17	17	17	17	260	229.625
0x60000288:	199.25	168.875	138.5	108.125	77.75	47.375
0x600002A0:	17	17	17	17	17	17
0x600002B8:	17	17	260	229.625	199.25	168.875
0x600002D0:	138.5	108.125	77.75	47.375	17	17
0x600002E8:	17	17	17	17	17	17
0x60000300:	260	229.625	199.25	168.875	138.5	108.125
0x60000318:	77.75	47.375	17	17	17	17
0x60000330:	17	17	17	17	260	229.625
0x60000348:	199.25	168.875	138.5	108.125	77.75	47.375
0x60000360:	17	17	17	17	17	17

Address: 0x60000000

0x60000378:	17	17	260	229.625	199.25	168.875
0x60000390:	138.5	108.125	77.75	47.375	17	17
0x600003A8:	17	17	17	17	17	17
0x600003C0:	260	229.625	199.25	168.875	138.5	108.125
0x600003D8:	77.75	47.375	17	17	17	17
0x600003F0:	17	17	17	17	0	0
0x60000408:	0	0	0	0	0	0
0x60000420:	0	0	0	0	0	0
0x60000438:	0	0	0	0	0	0

Input4

N=10, K=1

Address: 0x60000000

0x60000000: 1.0625	-32.4688	-66	8	82	-51
0x60000018: -184	-91.4531	1.09375	-16.4531	-34	255
0x60000030: 544	255.75	-32.5	-26.75	-21	-73.5
0x60000048: -126	-126	-31.4688	-134.234	-237	-96.625
0x60000060: 43.75	55.875	68	34.1758	0.351563	-8.25
0x60000078: -16.8516	257.574	532	113.875	-304.25	-157.68
0x60000090: -11.1094	-36.7031	-62.2969	-62.2969	-64	-236
0x600000A8: -408	-201.25	5.5	162.75	320	159.805
0x600000C0: -0.390625	-0.046875	0.296875	260.148	520	-28
0x600000D8: -576	-288.609	-1.21875	0.09375	1.40625	1.40625
0x600000F0: -240	-221.484	-202.969	-88.6094	25.75	92.7656
0x60000108: 159.781	-40.207	-240.195	-117.211	5.77344	94.8867
0x60000120: 184	140	96	48.1406	0.28125	-4.25781
0x60000138: -8.79688	-8.79688	-416	-206.969	2.0625	24.0313
0x60000150: 46	22.7813	-0.4375	-240.219	-480	-234.375
0x60000168: 11.25	-70.375	-152	308	768	384.891
0x60000180: 1.78125	-8.60938	-19	-19	-212	-73.4844
0x60000198: 65.0313	44.5313	24.0313	-2.09375	-28.2188	-133.484
0x600001B0: -238.75	-121.438	-4.125	-41.25	-78.375	-79.1875
0x600001C8: -80	-54.0547	-28.1094	-194.805	-361.5	-361.5
0x600001E0: -8	60	128	65.0313	2.0625	-26.9688
0x600001F8: -56	-26.75	2.5	-8.5	-19.5	-12.125
0x60000210: -4.75	-466.375	-928	-493	-58	-381
0x60000228: -704	-704	-8.625	157.688	324	163.766
0x60000240: 3.53125	-16.1719	-35.875	-16.9375	2	-4.3125
0x60000258: -10.625	-170.5	-330.375	-366.188	-402	-67.5
0x60000270: 267	-42.668	-352.336	-352.336	-9.25	255.375
0x60000288: 520	262.5	5	-5.375	-15.75	-7.125
0x600002A0: 1.5	-0.125	-1.75	-328.875	-656	-266
0x600002B8: 124	358	592	295.664	-0.671875	-0.671875
0x600002D0: -2.75	128.938	260.625	299.563	338.5	165.094
0x600002E8: -8.3125	-3.89844	0.515625	-2.08594	-4.6875	-158.094
0x60000300: -311.5	-124.5	62.5	172	281.5	140.324
0x60000318: -0.851563	-0.851563	3.75	2.5	1.25	336.625
0x60000330: 672	335.563	-0.875	-0.671875	-0.46875	-4.04688
0x60000348: -7.625	12.6875	33	17	1	-14
0x60000360: -29	-15.0156	-1.03125	-1.03125	3.625	-49.875

0x60000378: -103.375	234.313	572	285.508	-0.984375	-5.98438
0x60000390: -10.9844	-8.33594	-5.6875	-54.5938	-103.5	-62
0x600003A8: -20.5	7.5	35.5	18.0078	0.515625	0.515625
0x600003C0: 3.5	-102.25	-208	132	472	235.453
0x600003D8: -1.09375	-11.2969	-21.5	-12.625	-3.75	-121.875
0x600003F0: -240	-141	-42	29	100	51.0313
0x60000408: 2.0625	2.0625	169.75	32.7773	-104.195	66.168
0x60000420: 236.531	118.406	0.28125	-4.57813	-9.4375	-4.59375
0x60000438: 0.25	8.125	16	-32	-80	-12.75
0x60000450: 54.5	61.7656	69.0313	69.0313	336	167.805
0x60000468: -0.390625	0.335938	1.0625	1.35938	1.65625	2.14063
0x60000480: 2.625	3.4375	4.25	138.125	272	77
0x60000498: -118	-54.5	9	72.5	136	136
0x600004B0: 174.75	86.7148	-1.32031	7.10547	15.5313	9.19531
0x600004C8: 2.85938	-7.41406	-17.6875	200.219	418.125	310.063
0x600004E0: 202	74.125	-53.75	-24.5313	4.6875	23.3438
0x600004F8: 42	42	13.5	5.625	-2.25	13.875
0x60000510: 30	17.0313	4.0625	-16.9688	-38	397
0x60000528: 832	482	132	71.25	10.5	5.4375
0x60000540: 0.375	-25.8125	-52	-52	6.17188	2.99219
0x60000558: -0.1875	-240.594	-481	-239.656	1.6875	-12.0313
0x60000570: -25.75	196.063	417.875	242	66.125	37.1875
0x60000588: 8.25	-71.7813	-151.813	-131.906	-112	-112
0x600005A0: -1.15625	0.359375	1.875	-495.063	-992	-496.344
0x600005B8: -0.6875	-7.09375	-13.5	-4.875	3.75	2
0x600005D0: 0.25	3.125	6	-149	-304	-238
0x600005E8: -172	-172	-1.15625	0.359375	1.875	-495.063
0x60000600: -992	-496.344	-0.6875	-7.09375	-13.5	-4.875
0x60000618: 3.75	2	0.25	3.125	6	-149
0x60000630: -304	-238	-172	-172	0	0

Input5

N=20, K=2

First page

Address: 0x60000000

0x60000000: 3.5	4.59375	5.6875	6.78125	7.875	20.6563
0x60000018: 33.4375	46.2188	59	47	35	23
0x60000030: 11	11.375	11.75	12.125	12.5	-18.625
0x60000048: -49.75	-80.875	-112	-85.625	-59.25	-32.875
0x60000060: -6.5	7.625	21.75	35.875	50	189.5
0x60000078: 329	468.5	608	372	136	-100
0x60000090: -336	-238	-140	-42	56	41.8867
0x600000A8: 27.7734	13.6602	-0.453125	35.6602	71.7734	107.887
0x600000C0: 144	107.172	70.3438	33.5156	-3.3125	-1.79688
0x600000D8: -0.28125	1.23438	2.75	1.92969	1.10938	0.289063
0x600000F0: -0.53125	-88.3984	-176.266	-264.133	-352	-261.875
0x60000108: -171.75	-81.625	8.5	60.375	112.25	164.125
0x60000120: 216	190	164	138	112	112
0x60000138: 112	112	-27.375	-19.4219	-11.4688	-3.51563
0x60000150: 4.4375	23.8906	43.3438	62.7969	82.25	30.75
0x60000168: -20.75	-72.25	-123.75	-92.4688	-61.1875	-29.9063
0x60000180: 1.375	-26.2188	-53.8125	-81.4063	-109	-93.2188
0x60000198: -77.4375	-61.6563	-45.875	6.46875	58.8125	111.156
0x600001B0: 163.5	236.645	309.789	382.934	456.078	279.039
0x600001C8: 102	-75.0391	-252.078	-200.559	-149.039	-97.5195
0x600001E0: -46	14.415	74.8301	135.245	195.66	173.835
0x600001F8: 152.01	130.185	108.359	80.5605	52.7617	24.9629
0x60000210: -2.83594	-5.61133	-8.38672	-11.1621	-13.9375	-10.3848
0x60000228: -6.83203	-3.2793	0.273438	-67.2949	-134.863	-202.432
0x60000240: -270	-201.328	-132.656	-63.9844	4.6875	44.0664
0x60000258: 83.4453	122.824	162.203	137.652	113.102	88.5508
0x60000270: 64	64	64	64	-58.25	-43.4375
0x60000288: -28.625	-13.8125	1	27.125	53.25	79.375
0x600002A0: 105.5	14.5	-76.5	-167.5	-258.5	-196.313
0x600002B8: -134.125	-71.9375	-9.75	-33.8125	-57.875	-81.9375
0x600002D0: -106	-100.813	-95.625	-90.4375	-85.25	5.3125
0x600002E8: 95.875	186.438	277	283.789	290.578	297.367
0x60000300: 304.156	186.078	68	-50.0781	-168.156	-163.117
0x60000318: -158.078	-153.039	-148	-13.0566	121.887	256.83
0x60000330: 391.773	312.01	232.246	152.482	72.7188	53.9492
0x60000348: 35.1797	16.4102	-2.35938	-9.42578	-16.4922	-23.5586
0x60000360: -30.625	-22.6992	-14.7734	-6.84766	1.07813	-46.1914

End page

0x600060F0: -9	-7.01563	-5.03125	-3.04688	-1.0625	43.2031
0x60006108: 87.4688	131.734	176	134.5	93	51.5
0x60006120: 10	6.1875	2.375	-1.4375	-5.25	-3.375
0x60006138: -1.5	0.375	2.25	-3.8125	-9.875	-15.9375
0x60006150: -22	-216.5	-411	-605.5	-800	-599.934
0x60006168: -399.867	-199.801	0.265625	0.265625	0.265625	0.265625
0x60006180: 120	89.8047	59.6094	29.4141	-0.78125	-216.586
0x60006198: -432.391	-648.195	-864	-524	-184	156
0x600061B0: 496	204	-88	-380	-672	-462
0x600061C8: -252	-42	168	126.258	84.5156	42.7734
0x600061E0: 1.03125	0.857422	0.683594	0.509766	0.335938	54.252
0x600061F8: 108.168	162.084	216	162.176	108.352	54.5273
0x60006210: 0.703125	0.425781	0.148438	-0.128906	-0.40625	-2.55469
0x60006228: -4.70313	-6.85156	-9	-7.01563	-5.03125	-3.04688
0x60006240: -1.0625	43.2031	87.4688	131.734	176	134.5
0x60006258: 93	51.5	10	6.1875	2.375	-1.4375
0x60006270: -5.25	-3.375	-1.5	0.375	2.25	-3.8125
0x60006288: -9.875	-15.9375	-22	-216.5	-411	-605.5
0x600062A0: -800	-599.934	-399.867	-199.801	0.265625	0.265625
0x600062B8: 0.265625	0.265625	120	89.8047	59.6094	29.4141
0x600062D0: -0.78125	-216.586	-432.391	-648.195	-864	-524
0x600062E8: -184	156	496	204	-88	-380
0x60006300: -672	-462	-252	-42	168	126.258
0x60006318: 84.5156	42.7734	1.03125	0.857422	0.683594	0.509766
0x60006330: 0.335938	54.252	108.168	162.084	216	162.176
0x60006348: 108.352	54.5273	0.703125	0.425781	0.148438	-0.128906
0x60006360: -0.40625	-2.55469	-4.70313	-6.85156	-9	-7.01563
0x60006378: -5.03125	-3.04688	-1.0625	43.2031	87.4688	131.734
0x60006390: 176	134.5	93	51.5	10	6.1875
0x600063A8: 2.375	-1.4375	-5.25	-3.375	-1.5	0.375
0x600063C0: 2.25	-3.8125	-9.875	-15.9375	-22	-216.5
0x600063D8: -411	-605.5	-800	-599.934	-399.867	-199.801
0x600063F0: 0.265625	0.265625	0.265625	0.265625	0	0

Input6

N=10, K=3

First page

Address: 0x60000000

0x60000000: 160	102	44	-14	-72	-130
0x60000018: -188	-246	-304	-233	-162	-91
0x60000030: -20	51	122	193	264	223
0x60000048: 182	141	100	59	18	-23
0x60000060: -64	-22	20	62	104	146
0x60000078: 188	230	272	237.555	203.109	168.664
0x60000090: 134.219	99.7734	65.3281	30.8828	-3.5625	49.8828
0x600000A8: 103.328	156.773	210.219	263.664	317.109	370.555
0x600000C0: 424	366.75	309.5	252.25	195	137.75
0x600000D8: 80.5	23.25	-34	-30.5156	-27.0313	-23.5469
0x600000F0: -20.0625	-16.5781	-13.0938	-9.60938	-6.125	-5.64063
0x60000108: -5.15625	-4.67188	-4.1875	-3.70313	-3.21875	-2.73438
0x60000120: -2.25	-2.25	-2.25	-2.25	-2.25	-2.25
0x60000138: -2.25	-2.25	140.156	86.7617	33.3672	-20.0273
0x60000150: -73.4219	-126.816	-180.211	-233.605	-287	-215.75
0x60000168: -144.5	-73.25	-2	69.25	140.5	211.75
0x60000180: 283	240.281	197.563	154.844	112.125	69.4063
0x60000198: 26.6875	-16.0313	-58.75	-21.6768	15.3965	52.4697
0x600001B0: 89.543	126.616	163.689	200.763	237.836	200.092
0x600001C8: 162.348	124.604	86.8594	49.1152	11.3711	-26.373
0x600001E0: -64.1172	-9.7207	44.6758	99.0723	153.469	207.865
0x600001F8: 262.262	316.658	371.055	319.767	268.479	217.19
0x60000210: 165.902	114.614	63.3262	12.0381	-39.25	-35.8887
0x60000228: -32.5273	-29.166	-25.8047	-22.4434	-19.082	-15.7207
0x60000240: -12.3594	-10.2949	-8.23047	-6.16602	-4.10156	-2.03711
0x60000258: 0.0273438	2.0918	4.15625	4.15625	4.15625	4.15625
0x60000270: 4.15625	4.15625	4.15625	4.15625	120.313	71.5234
0x60000288: 22.7344	-26.0547	-74.8438	-123.633	-172.422	-221.211
0x600002A0: -270	-198.5	-127	-55.5	16	87.5
0x600002B8: 159	230.5	302	257.563	213.125	168.688
0x600002D0: 124.25	79.8125	35.375	-9.0625	-53.5	-21.3535
0x600002E8: 10.793	42.9395	75.0859	107.232	139.379	171.525
0x60000300: 203.672	162.629	121.586	80.543	39.5	-1.54297
0x60000318: -42.5859	-83.6289	-124.672	-69.3242	-13.9766	41.3711
0x60000330: 96.7188	152.066	207.414	262.762	318.109	272.783
0x60000348: 227.457	182.131	136.805	91.4785	46.1523	0.826172
0x60000360: -44.5	-41.2617	-38.0234	-34.7852	-31.5469	-28.3086

End page

0x60006120: -2.5625	-102.242	-201.922	-301.602	-401.281	-500.961
0x60006138: -600.641	-700.32	-800	-708	-616	-524
0x60006150: -432	-340	-248	-156	-64	-64
0x60006168: -64	-64	-64	-64	-64	-64
0x60006180: -15.5	-12.5625	-9.625	-6.6875	-3.75	-0.8125
0x60006198: 2.125	5.0625	8	0.625	-6.75	-14.125
0x600061B0: -21.5	-28.875	-36.25	-43.625	-51	-43
0x600061C8: -35	-27	-19	-11	-3	5
0x600061E0: 13	10.375	7.75	5.125	2.5	-0.125
0x600061F8: -2.75	-5.375	-8	-58	-108	-158
0x60006210: -208	-258	-308	-358	-408	-435
0x60006228: -462	-489	-516	-543	-570	-597
0x60006240: -624	-546.32	-468.641	-390.961	-313.281	-235.602
0x60006258: -157.922	-80.2422	-2.5625	-102.242	-201.922	-301.602
0x60006270: -401.281	-500.961	-600.641	-700.32	-800	-708
0x60006288: -616	-524	-432	-340	-248	-156
0x600062A0: -64	-64	-64	-64	-64	-64
0x600062B8: -64	-64	-15.5	-12.5625	-9.625	-6.6875
0x600062D0: -3.75	-0.8125	2.125	5.0625	8	0.625
0x600062E8: -6.75	-14.125	-21.5	-28.875	-36.25	-43.625
0x60006300: -51	-43	-35	-27	-19	-11
0x60006318: -3	5	13	10.375	7.75	5.125
0x60006330: 2.5	-0.125	-2.75	-5.375	-8	-58
0x60006348: -108	-158	-208	-258	-308	-358
0x60006360: -408	-435	-462	-489	-516	-543
0x60006378: -570	-597	-624	-546.32	-468.641	-390.961
0x60006390: -313.281	-235.602	-157.922	-80.2422	-2.5625	-102.242
0x600063A8: -201.922	-301.602	-401.281	-500.961	-600.641	-700.32
0x600063C0: -800	-708	-616	-524	-432	-340
0x600063D8: -248	-156	-64	-64	-64	-64
0x600063F0: -64	-64	-64	-64	0	0

위 이미지는 TEST Input으로 주어진 데이터의 결과 값입니다. 특히 Input 5, 6은 데이터의 양이 크기 때문에 첫번째 page와 마지막 page만 올리게 되었습니다. 다음으로는 구현한 프로그램의 Performance입니다.

Code Size

```
Program Size: Code=960 RO-data=0 RW-data=408 ZI-data=0
```

Input1

States 1465

State*state*code = **2060376000**

Input2

States 5774

State*state*code = **32005512960**

Input3

States 8686

State*state*code = **72428732160**

Input4

States 15262

State*state*code = **223611498240**

Input5

States 287401

State*state*code = **79295361408960**

Input6

States 275848

State*state*code = **73048434339840**

5. Conclusion

프로젝트 전까지 해왔던 과제를 기반으로 프로젝트를 구현할 수 있었습니다. 프로젝트와 평소 과제의 차이점으로는 복잡도와 Register의 사용제한 정도가 있었습니다. 특히 프로젝트를 구현하면서 Floating point를 계산하기 때문에 오래 정보를 유지해야 하는 경우 제약이 많았습니다. 이로 인하여 Register하나를 메모리에 접근하여 메모리에 정보를 기억할 수 있도록 구현하게 되었습니다. 문제점은 메모리에 접근하는 명령자체가 state를 많이 잡아먹기 때문에 최대한 반복적으로 수행하는 곳은 Register를 통한 계산과 비교를 할 수 있도록 하였고 계산이 끝나는 padding부분에서는 메모리 접근을 통한 조건(count)를 수행하게 되며 최대한 state를 줄일 수 있도록 하였습니다. 또한 Branch를 사용한 접근과 code size는 커지지만 Unrolling을 통한 구현으로 state의 차이를 확인하게 되었으며, 알고리즘과 k, n의 값에 따라 state가 달라지는 것을 확인하게 되었습니다. 이번 프로젝트를 진행하면서 구현한 프로젝트는 총 3가지이며, 각각 프로그램의 성능을 비교하여 가장 state가 좋은 코드를 선정하게 되었습니다. 가장 먼저 구현한 프로그램은 중점을 이용한 구현이며, 두번째로 구현한 프로그램은 두 점의 거리를 구해서 k값에 따라 거리를 나누어 순차적으로 더하는 프로그램입니다. 세번째는 중점정리를 다시한번 구현하며 최적화한 프로그램입니다. 첫번째 프로그램과 두번째 프로그램을 작성하며 문제가 있었는데 어느 한 개의 프로그램이 모든 case에서 좋은 퍼포먼스를 나타내는 것이 아니라 k, n의 값에 따라 퍼포먼스가 달랐습니다. 이유는 두번째 프로그램인 거리를 구하는 프로그램은 k의 값이 적고 n의 값이 크면 두 점의 거리를 구하는 계산기 한번 더 쓰이기 때문에 state가 크지만 k의 값이 크면 state가 좋게 나왔습니다. 그래서 두 점의 거리를 구하는 계산이 한번 더 들어가기에 한계가 있을 거라 생각되어 중점을 이용한 프로그램을 토대로 state를 줄이며 3번째 프로그램을 구현하게 되었습니다. 해당 프로젝트를 진행하며 Register가 부족한 경우 대처할 수 있는 방법과 state를 줄이는 방법을 배우고, 화면을 확대할 때 해당 프로그램을 응용할 수 있다는 것을 배우게 되었습니다. 프로젝트를 진행 후 기대되는 학습 효과로는 최종적인 목적을 달성하는 code뿐만 아니라 작성한 프로그램의 성능까지 고려하는 생각을 할 수 있다는 점입니다. 지금 까지는 단순히 원하는 동작만 나오면 만족하며, 프로그램의 속도나 성능은 생각하지 않았지만 해당 프로젝트를 진행하며 code size와 state를 많이 신경 쓰며 진행하였기 효율적인 code작성이 보다 쉽게 가능하다고 생각됩니다.