

어셈블리 프로그램 설계 및 실습

프로젝트 보고서

Matrix Convolution

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

강의 시간: 화 0, 1, 2, 목 2

학 번: 2018202046

성 명: 이준휘

1. Introduction

A. Title

Matrix convolution

B. Object

해당 프로젝트를 통해 현재까지 배운 어셈블리 언어를 활용하여 주어진 과제와 관련한 코드를 작성할 수 있다. 해당 Matrix의 Padding, Convolution, Sub-Sampling의 원리를 알 수 있고 이를 코드로 구현할 수 있다. 주어진 조건을 고려하여 state를 최적화하는 방향으로 코드를 작성함으로써 state를 줄일 수 있는 다양한 방법을 연구한다.

C. Schedule

	11/21~11/22	11/23~11/24	11/25~11/26	11/27~11/28
제안서 작성				
프로젝트 수행				
보고서 작성				

2. Algorithm

A. FloatingAdd

해당 함수는 양 또는 0끼리의 덧셈을 진행하는 branch다. 이 때 두 피연산자의 값은 Float형으로 표기된다. 해당 함수에서는 r9, r10에 값을 넣으며 r8에 결과를 출력한다. 또한 r6 ~ r12의 범위의 레지스터를 사용한다.

피연산자의 값이 0일 경우는 0x80000000 또는 0x00000000이다. 이를 판단하기 위해 두 피연산자의 sign bit를 LSL 1을 통해 제거하여 해당 값이 0인지를 비교한다. 만약 0으로 판별될 경우 해당 덧셈의 결과는 0이 아닌 피연산자의 값으로 반환되기 때문에 0이 아닌 피연산자를 반환하고 프로그램을 종료한다.

0이 아님을 확인한 후에는 LSL과 LSR을 통해 두 피연산자의 exponent랑 Mantissa를 분리한다. 이 때 Mantissa는 값 손실이 일어났을 경우 이전 비트에서 반올림을 해주어야 하기 때문에 1:24 자리에 위치시켜준다. 그 후 mantissa 맨 앞에 생략되었던 1을 붙여주고 두 exponent를 비교한다. exponent가 작은 쪽의 Mantissa를 두 exponent의 차만큼 LSR을 통해 위치시켜 두 값을 연산할 수 있도록 도와준다. Mantissa는 단순히 더하기만 진행해도 되는데 이유는 두 값이 조건에서 양수라고 주어졌기 때문이다. 두 덧셈의 연산

결과가 26비트 즉, 이전 자리를 기억하기 위한 비트까지 포함한 25비트를 초과한 경우 LSL을 통해 25비트로 만들어주면서 동시에 큰 쪽의 exponent의 값을 1 늘려준다.

이후에는 Mantissa의 반올림 과정과 float 값으로 재결합 과정이다. 0번째 자리의 비트를 따로 떼어낸 후 Mantissa를 우측으로 1번 shift 해준다. 그 후 해당 수를 0번째 자리를 떼어낸 비트와 덧셈 연산을 통해 반올림을 진행해준다. 반올림이 끝나면 해당 값과 exponent, sign(0)을 결합하고 해당 함수를 종료한다.

B. FloatingMul

해당 함수는 양의 자연수와 양의 정수끼리의 곱셈을 진행하는 함수다. 이 때 두 피연산자의 값은 Float형으로 표기된다. 해당 함수에서는 r9, r10에 값을 넣는다. 이 때 r9의 값에는 자연수를, r10 자리에는 정수를 넣어주어야 한다. 그 후 연산 결과는 r8에 출력한다. 또한 r6 ~ r12의 범위의 레지스터를 사용한다.

해당 함수에서는 FloatingAdd와 마찬가지로 0을 확인하는 과정이 있다. 만약 한 쪽의 피연산자가 0일 경우 해당 결과를 0으로 출력하고 함수를 종료한다.

이후 덧셈과 동일하게 Mantissa와 exponent를 분리시켜주는데 Mantissa의 경우에는 자연수는 상위 8비트의 Mantissa만 남겨둔다. 이유는 이후의 비트가 0이기 때문이다. 이후에 Mantissa에 맨 앞에 1을 붙여주는 작업을 수행한다.

Exponent끼리 연산은 두 값의 덧셈으로 이루어지는데 이 때 두 값에서 127을 뺀 값을 더한 후 다시 127을 더한다. 기본적으로 exponent가 지수에 127을 더한 값이기 때문에 이러한 과정이 추가되었다.

해당 과정을 마친 후 exponent가 8비트인 Mantissa에 이전 비트의 패턴인 0을 뒤에 덧붙여준다. 그 후 result를 저장할 r8을 0값으로 초기화 시켜준 후 MulLoop로 넘어간다.

MulLoop에서는 실질적으로 4-radix Booth algorithm이 구현되는 부분이다. 해당 부분에서는 Multiplier(9비트)의 마지막 3개의 비트의 패턴을 확인한다. 그리고 이를 0, 1~2, 3, 4, 5~6, 7의 경우에 따라 Multiplicand를 더해주거나 빼주는 것을 달리한다. 그 후 MulShift 과정으로 넘어간다.

MulShift에서는 result와 multiplier의 비트 조작과 탈출 조건이 명시된 부분이다. 우선

Multiplier를 LSR 2를 진행한다. 만약 해당 결과가 0일 경우 result는 shift를 진행시키지 않고 아닐 경우에는 result를 ASR 2를 진행하고 MulLoop로 돌아가서 다시 연산을 수행한다.

Mantissa 결과가 나온 경우 이를 맞추어주고 반올림하는 과정이 필요하다. 우선 연산 결과로 나온 Mantissa가 26비트보다 큰지 확인한다. 만약 작을 경우 이전에 추가했던 맨 앞자리 1을 제거하고 마지막 비트의 패턴을 따로 저장해둔다.

그 후 비트를 shfit하여 비트 수를 1개 줄인다. 이후 만약에 아까 비교에서 크거나 같은 결과였을 경우 마지막 비트의 패턴을 지금 저장해두고, shift를 한 번 더 진행한다.

해당 값과 이전 비트의 패턴을 더하여 반올림을 진행하고 exponent와 Sign(0) 값과 함께 결합시킨 후 함수를 종료한다.

C. Matrix function

해당 함수의 경우 Padding, Convolution, Sub-Sampling을 동시에 진행한다. 우선 행의 row를 r2, col을 r3로 설정한 후 First_Line_First_Convolution으로 넘어간다.

D. First_Line_First_Convolution

Padding된 Matrix에서 일어나는 Convolution이 다른 Convolution과 다르다. 특히 첫번째 Convolution은 중간에서 일어나는 Convolution과 다르기(Padding) 때문에 따로 제작된 Convolution이다. 해당 부분에서는 곱을 진행하는 Matrix는 다음과 같다.

A	B	C
D	E	F
G	H	I

Matrix_second

a	a	b
a	a	b
c	c	d

Matrix_first

해당 연산의 결과는 $a * A + a * B + a * D + a * E + b * C + b * F + c * G + c * H + d * I$ 이며 이는 $a(A + B + D + E) + b(C + F) + c(G + H) + d * I$ 로 묶어서 연산할 수 있다.

해당 함수는 이 묶은 연산에 해당되며 r4에 각 단계의 중간 연산을 누적한다. 연산이 끝나면 저장할 위치 sp에 이를 저장하고, Matrix_first의 읽는 위치를 4byte만큼 옮긴다. 그 후 col을 2 늘린다. 위치를 4byte, col을 2늘리는 이유는 Sub_samplig을 한다면 다음 위

치가 아니라 그 다음의 위치에서 연산을 수행하기 때문이다. 이후에는 First_Line_Middle_Convolution을 진행한다.

E. First_Line_Middle_Convolution

해당 행 또한 일반적인 cell과는 다른 연산이 진행되기 때문에(Padding) 따로 분리하여 연산을 시켜주었다.

해당 연산이 진행되는 Matrix는 다음과 같다.(채색된 cell은 현재 위치한 주소를 의미한다.)

A	B	C
D	E	F
G	H	I

Matrix_second

a	b	c
a	b	c
d	e	f

Matrix_first

해당 연산의 결과는 $a * A + a * D + b * B + b * E + c * C + c * F + d * G + e * H + f * I$ 이며 이는 $a(A + D) + b(B + E) + c(C + F) + d * G + e * H + f * I$ 로 묶어서 연산할 수 있다.

해당 함수 또한 위의 함수처럼 연산을 수행하고 이를 sp에 저장한다. 그 후 col을 62와 비교하여 작을 경우에는(해당 줄의 Convolution이 전부 진행되지 않은 경우) col을 2, r0를 8byte 다음으로 위치시킨 후 다시 First_Line_Middle_Convolution을 진행한다. 만약 같은 경우(해당 줄에서 연산이 끝난 경우) 다다음줄의 첫 번째 위치로 이동해야하기 때문에 col값은 0으로 초기화시켜주고 row 값은 2 늘려준 후 r0가 읽는 주소를 $4 * 64 + 4 + 4$, 즉 0x108 위치로 이동시켜준다. 그 후 Middle_Line_First_Convolution을 진행한다.

F. Middle_Line_First_Convolution

모든 행의 첫 번째 Convolution은 Padding되어있기 때문에 일반적인 상황과는 다른 연산이 진행된다. 때문에 이를 따로 분리하여 연산을 시켜주었다.

해당 연산이 진행되는 Matrix는 다음과 같다.(채색된 cell은 현재 위치한 주소를 의미한다.)

A	B	C
D	E	F
G	H	I

Matrix_second

a	a	b
c	c	d
e	e	f

Matrix_first

해당 연산의 결과는 $a * A + a * B + b * C + c * D + c * E + d * F + e * G + e * H + f * I$ 이며 이는 $a(A + B) + c(D + E) + e(G + H) + b * C + d * F + f * I$ 로 묶어서 연산할 수 있다.

해당 연산을 수행한 결과는 sp에 저장되며 First_Line_First_Convolution과 마찬가지로 col과 r0를 조작시킨 후 Middle_Line_Middle_Convolution을 진행시켜준다.

G. Middle_Line_First_Convolution

해당 연산이 진행되는 Matrix는 다음과 같다.(채색된 cell은 현재 위치한 주소를 의미한다.)

A	B	C
D	E	F
G	H	I

Matrix_second

a	b	c
d	e	f
g	h	i

Matrix_first

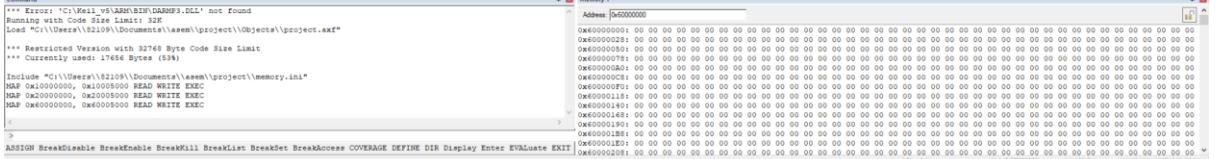
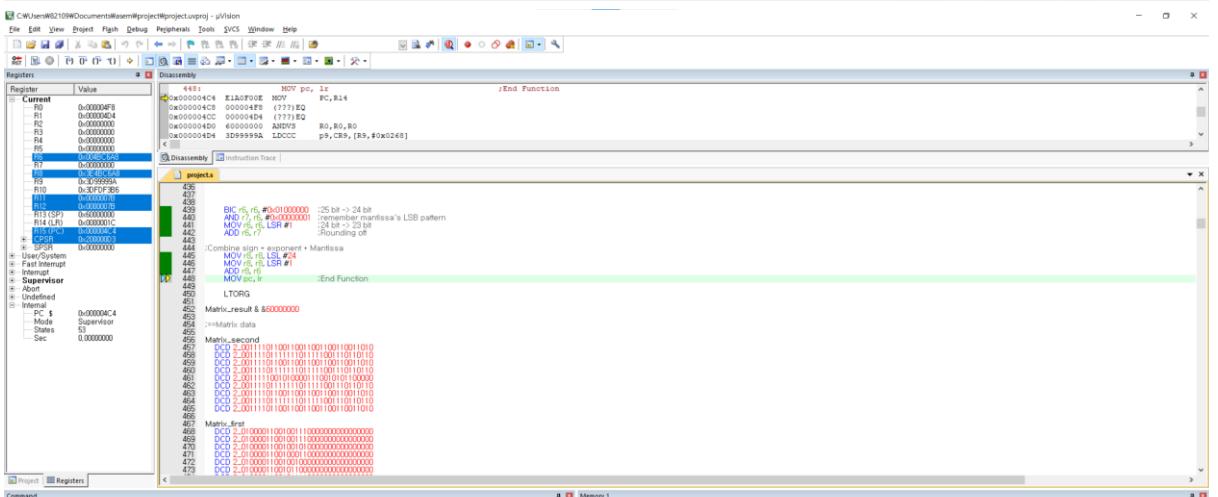
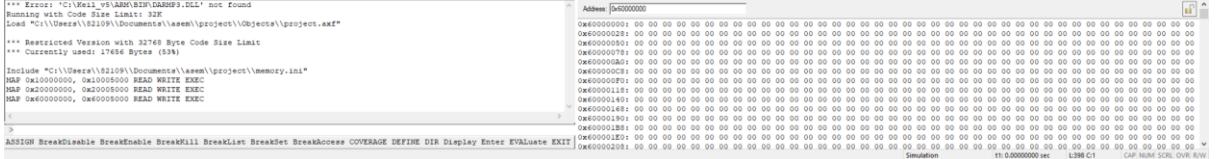
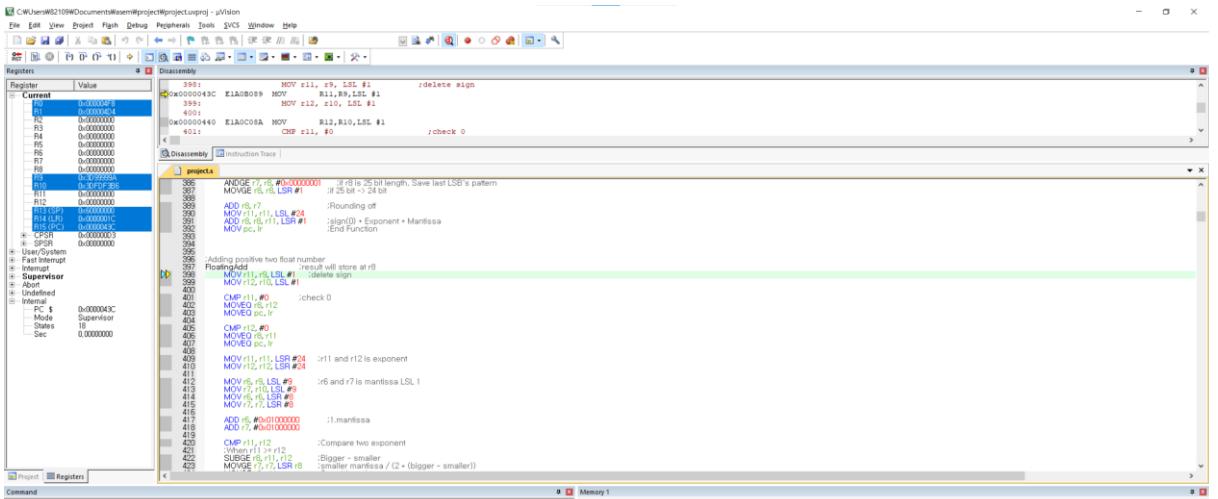
해당 연산의 결과는 $a * A + b * B + c * C + d * D + e * E + f * F + g * G + h * H + i * I$ 이며 묶을 수 없다.

해당 연산을 수행한 결과는 sp에 저장되며 First_Line_Middle_Convolution과 마찬가지로 col이 62보다 작은지 확인하고 작을 경우 col을 늘리고 주소를 8 옮기는 과정을 수행한다. 만약 col이 62일 경우 해당 row가 62보다 작은지 확인한다. 만약 작을 경우 다다음 줄의 첫번째 위치로 이동할 수 있도록 row는 2 늘리고, col은 0으로 초기화, 그리고 주소는 0x108만큼 이동시켜준 후 Middle_Line_First_Convolution으로 이동한다. 만약 row 또한 62

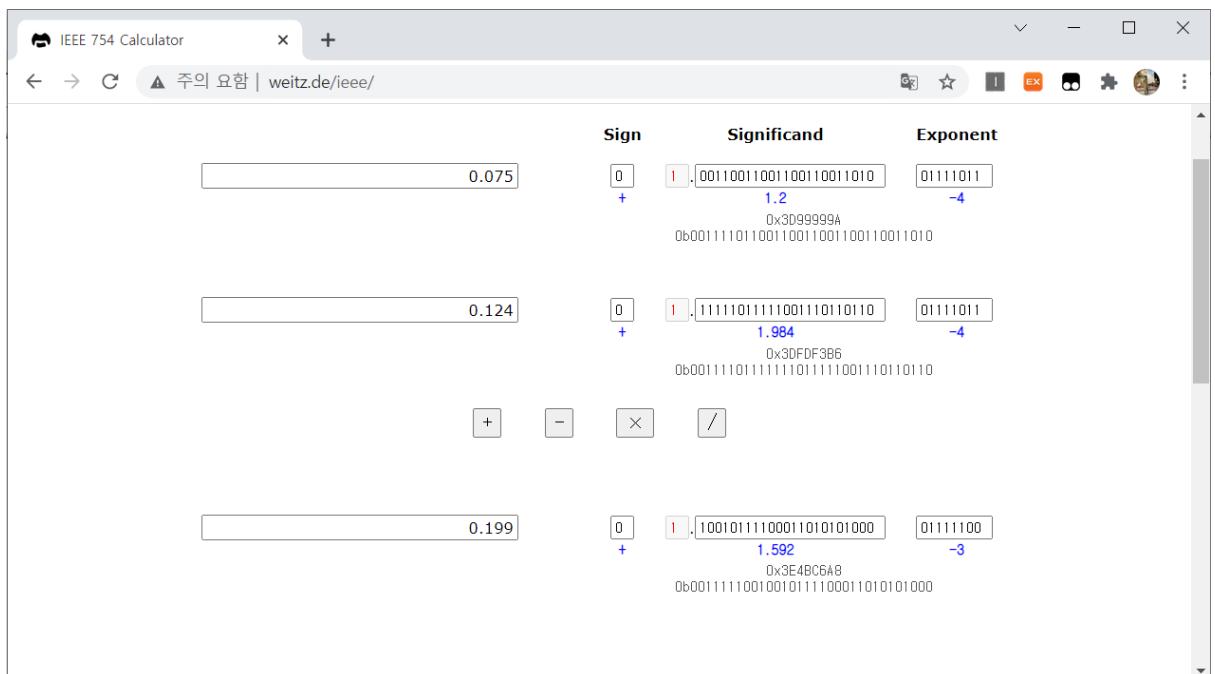
일 경우 프로그램은 종료된다.

3. Performance & Result

A. FloatingAdd

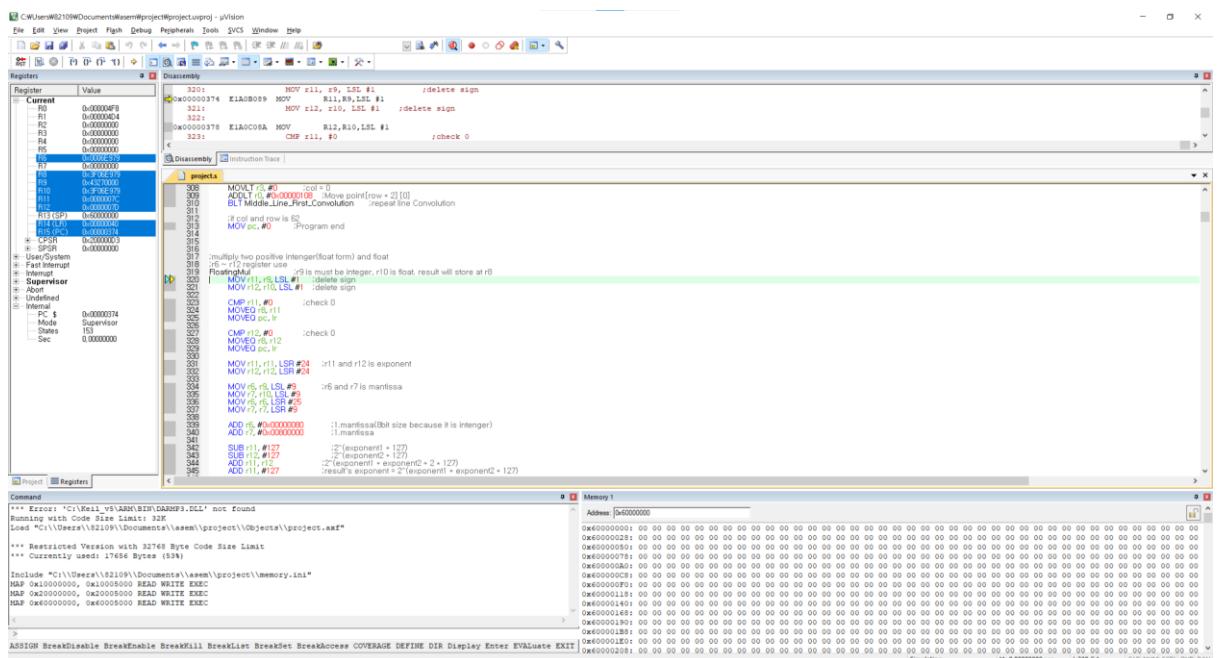


해당 사진은 덧셈이 제대로 되는지 확인하는 사진이다 0x3D99999A와 0x3DFDF3B8의 덧셈 결과로 0x3E4BC6A8이 나오는 모습을 보인다.



덧셈의 결과가 정확히 나오는 것을 볼 수 있다.

B. FloatingMul



The screenshot shows the WinDbg debugger interface with the Registers window at the top and the Assembly window below it. The assembly code is as follows:

```
0:00000043E0880A8 ADD R8,R9,R11,LSR #1
    MOV pc, lr      ;End Function

3941
3941
3942 ;Adding positive two float number

Disassembly | Instructions Trace |
```

The assembly code continues with several floating-point operations involving registers R9, R10, R11, R12, and R13, including comparisons (CMP), moves (MOV), and arithmetic operations (ADD). The code is annotated with comments explaining the purpose of each instruction, such as "if r9 is not 25 bit length(25 bit)" and "Save last LSB's pattern". The assembly code ends with a "End Function" label.

해당 결과를 보면 값이 들어가서 r8로 나오는 것을 확인할 수 있다. 해당 값이 맞는지 확인해보면 아래와 같이 동일하게 결과가 나오는 것을 확인할 수 있다.

The screenshot shows the IEEE 754 Calculator interface with three floating-point numbers displayed in binary format:

- 0.527**: Sign = 0, Significand = 1.0001101110100101111001, Exponent = 01111110. Value: 0.527.
- 167.0**: Sign = 0, Significand = 1.01001110000000000000000, Exponent = 10000110. Value: 167.0.
- 88.009**: Sign = 0, Significand = 1.01100000000010010011100, Exponent = 10000101. Value: 88.009.

The calculator also features a toolbar with standard arithmetic operators (+, -, ×, ÷) and a clear button (C). A note at the bottom of the page reads: "See info at bottom of page."

C. First_Line_First_Convolution

해당 결과를 확인해보면 정상적으로 연산 후 저장되는 것을 확인할 수 있다.

Two screenshots of the µVision debugger showing assembly code and memory dump.

Screenshot 1:

- Registers:** Shows registers R0-R7, PC, CPSR, and CPSR.
- Disassembly:** Shows assembly code for the `BLT First_Line_Middle_Convolution` function. The code includes instructions for loading H, clearing r3, moving r3, adding r2, incrementing r0, and performing floating-point additions and subtractions.
- Memory Dump:** Shows memory starting at address 0x00000000. Values include 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 19

또한 한 줄이 전부 계산되는 모습을 볼 수 있다.

두 가지 경우의 디버깅 환경을 보여주는 스크린샷입니다.

상단 이미지(Left): **Memory 1** 탭에서 주소 0x00000000부터 0x0000000F까지의 메모리 내용을 표시하는 테이블입니다.

Address	0x00000000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007	0x00000008	0x00000009	0x0000000A	0x0000000B	0x0000000C	0x0000000D	0x0000000E	0x0000000F
0x40000000:0001	166.651	164.726	164.042	168.972	128.103	118.431	119.424	129.253								
0x40000000:01	138.979	141.075	142.801	143.726	143.527	142.801	141.801	142.651								
0x40000000:02	162.328	140.159	140.253	143.103	130.057	134.541	142.021	142.378								
0x40000000:03	154.488	150.279	139.991	139.253	132.451	134.398	132.322									
0x40000000:04	165.199	0	0	0	0	0	0	0								
0x40000000:05	0	0	0	0	0	0	0	0								
0x40000000:06	0	0	0	0	0	0	0	0								
0x40000000:07	0	0	0	0	0	0	0	0								
0x40000000:08	0	0	0	0	0	0	0	0								
0x40000000:09	0	0	0	0	0	0	0	0								
0x40000000:0A	0	0	0	0	0	0	0	0								
0x40000000:0B	0	0	0	0	0	0	0	0								
0x40000000:0C	0	0	0	0	0	0	0	0								
0x40000000:0D	0	0	0	0	0	0	0	0								
0x40000000:0E	0	0	0	0	0	0	0	0								
0x40000000:0F	0	0	0	0	0	0	0	0								

하단 이미지(Right): **Memory 1** 탭에서 주소 0x00000000부터 0x0000000F까지의 메모리 내용을 표시하는 테이블입니다.

Address	0x00000000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007	0x00000008	0x00000009	0x0000000A	0x0000000B	0x0000000C	0x0000000D	0x0000000E	0x0000000F
0x40000000:0001	178	157	122	111	117	118	119	122								
0x40000000:01	130	136	140	141	141	141	143	144								
0x40000000:02	144	143	144	144	143	142	142	142								
0x40000000:03	143	143	143	141	141	140	140	140								
0x40000000:04	134	126	146	141	142	143	142	142								
0x40000000:05	133	131	131	131	131	134	134	135								
0x40000000:06	133	139	166	165	164	163	165	172								
0x40000000:07	133	139	166	165	164	163	165	172								
0x40000000:08	134	139	141	141	140	140	143	144								
0x40000000:09	134	143	143	143	143	142	142	142								
0x40000000:0A	134	143	143	143	143	142	142	142								
0x40000000:0B	134	142	142	141	140	139	139	139								
0x40000000:0C	134	142	142	141	140	139	139	139								
0x40000000:0D	134	142	142	141	140	139	139	139								
0x40000000:0E	134	142	142	141	140	139	139	139								
0x40000000:0F	134	142	142	141	140	139	139	139								

해당 결과를 확인하면 다음 줄의 첫번째 Convolution이 정상적으로 된 모습을 볼 수 있다.

F. Middle_Line_Middle_Convolution

다음 인자가 정상적으로 계산된 것을 확인할 수 있다

The screenshot shows the Immunity Debugger interface with several windows open:

- Registers**: Shows CPU register values.
- Disassembly**: Displays assembly code with annotations. The current instruction is `CMPSB EDX, ECX`. Annotations include:
 - Row col is 62 and row is little than 62
 - ADSLT r2, #2 :row ++ 2
 - ADSLT r2, #2 :row == 2
 - MOVLT r3, #1 :rowl = 0
- Registers**: Shows CPU register values.
- Disassembly**: Shows assembly code with annotations. The current instruction is `STR r5, [sp], #4`. Annotations include:
 - :Save SUM
 - CMPSB EDX, ECX :if col is little than 62
 - ADSLT r2, #2 :row == 2
 - ADSLT r2, #2 :row ++ 2
 - ADSLT r2, #2 :row == 2 (repeat Cell Convolution)
 - BLT Middle_Line.Middle_Convolution :repeat Cell Convolution
- Registers**: Shows CPU register values.
- Memory**: Shows memory dump at address 0x40000000.

해당 값은 해당 줄이 전부 계산된 결과다. 정상적으로 출력되는 것을 확인할 수 있다.

The screenshot shows the Immunity Debugger interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help. The Registers pane on the left shows CPU registers like R0-R15, CPSR, and PSR, with their current values. The Registers pane also includes sections for User-System, Fast Interrupt, Supervisor, and Undefined. The bottom Command pane displays assembly code and memory dump information.

해당 결과를 보면 기존의 모든 데이터를 읽었고, $32 \times 32 \times 4$ 개의 크기와 맞게 데이터가 저장되고 프로그램이 종료되는 모습을 볼 수 있다.

4. Conclusion

해당 프로젝트를 수행하면서 기본적으로 코드를 주어진 조건에 따라 최적화 시키기 위해 여러가지 방법을 사용했다. 우선 Padding, Convolution, Sub_Sampling 과정을 통합함으로써 불필요한 Convolution을 줄이고 메모리에 결과로는 쓰이지 않은 값을 저장하는 것을 줄였다. 또한 Convolution 과정에서 식을 묶어서 연산을 진행함으로써 불필요한 곱셈이나 덧셈의 연산 횟수가 많아지는 것을 방지하였다. 또한 Branch를 이동해야 할 상황을

되도록 피하도록 프로그램을 작성하고 비교를 할 경우 3개 이하로 instruction이 들어가도록 고려하였다. 마지막으로 곱셈의 연산 과정에서 현재 주어진 조건이 자연수와 정수의 곱셈임을 고려하여 자연수의 패턴을 파악하고 자연수의 Mantissa를 줄여 Loop를 통한 반복을 최대한 줄이는 것을 고려하였다.

위와 같은 과정을 진행하며 코드를 줄일 수 있는 다양한 방법을 고려하는 코드 작성에 대해 생각할 수 있었고, 알고리즘의 최적화에 대한 여러가지 방안 또한 생각하는 프로젝트였다. 또한 Matrix를 조작하는 여러가지 방법을 알 수 있는 시간이었다.