어셈블리프로그래밍설계 및 실습

Term project 결과보고서

설계제목: matrix convolution

설계일자: 2021년 11월 22~11월 29일

제출일자: 2021년 11월 30일 (화요일)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습분반: 화0,1 목2

학 번: 2018202060

성 명: 이 준 형

1. 제목 및 일정

A. 제목

matrix convolution

B. 일정

기간 / 항목	1주차							2주차
11월	23	24	25	26	27	28	29	30
제안서 제출								
문제정의 및 정보수집								
coding								
coding 검증								
Report 작성								
최종 제출								

2. Project specification

이번에 설계한 프로젝트의 전체적인 흐름은 매트릭스의 값을 불러와서 패딩을 시켜서 저장한 후, 그 매트릭스의 행렬곱을 시키면서 전에 했던 SUM과 점점 합쳐가면서 9번 연산이 완료되면 그 곱이 짝수행 짝수열인지 확인을 하고 짝수행 짝수열이면 저장을 시켰습니다. 곱하면서 sub_sampling을 진행했습니다.

PADDING과정

START

주어진 매트릭스와 패딩이 되는 매트릭스를 활용하기 위해 이 함수에서 로드를 시키고 카운팅을 할 레지스터를 미리 초기화했습니다.

PADDING

먼저 맨 첫째줄의 패딩하는 과정인데 맨 첫째줄에 로드를 하고 SAVE를 2번씩 시키는 과정입니다.

PADDING_LOOP

OFFSET을 4씩 증가시키며 행렬의 값을 61번 STR시켰습니다.

CHECK_PADDING_LOOP

한줄의 행렬의 패딩이 끝났는지 확인을 하는 함수입니다.

INNER2_PADDING_LOOP

두번째줄부터 맨 마지막줄 바로 위에줄의 패딩하는 반복문 함수입니다.

STOP

맨 아랫줄이 되기전까지 패딩이 되었으면 OFFSET을 -256시켜 맨 아랫줄 패딩을 시작합니다.

PADDING_LOOP3

맨 아랫줄 패딩하는 과정인데, 맨 첫째줄 패딩하는 과정이랑 흡사합니다.

PADDING_END

패딩이 끝난경우 convolution에 대비해서 미리 작은 음수의 offset을 이용해서 패딩 매트릭스의 첫번째 원소를 가르키게 했습니다.

MATRIX CONVOLUTION 과정

DOMUL

먼저 booth multiplication을 위해서 sign과 exponent, mantissa를 분리 시키는 과정입니다. 그 후 R2에 32의 절반인 16이 아닌 28의 절반인 14를 주어서 RADIX_4연산을 시켰습니다.

RADIX_4

R2가 0이 될때까지 아래의 연산 과정을 진행시켰습니다.

SHIFT, ADD_1A, ADD_2A, SUB_1A, SUB_2A

MUL_NOMALIZATION

RADIX_4연산이 끝난 후, 2^28인134217728과 비교를 해서 exponent를 더해주었습니다.

MUL_ADDING

sign과 exponent mantissa가 전부 합치는 과정입니다.

MUL_FINISH

곱한것과 전에 곱해서 합쳐진것을 더하는 과정으로 넘어가는 함수입니다.

DOADD

먼저 입력받은 값들이 0인지 아닌지 확인을 합니다. 그 후에 mul을 하기위해서 sign, exponent, mantissa를 추출했습니다.

ADDING_LOOP, ADDING_MANTISSA, ADDING_ADDING

2^24인 16777216을 비교한 후, mantissa를 더할지 안더할지 결정 후, 더해줍니다,

SUB_SAMPLING 과정

CHECK_STATE

앞에서 곱하고 더한과정이 3번째인지 6번째인지 9번째 인지 확인을 하고, 그 번째에 맞는 함수를 call합니다.

THIRD_SIXTH_CAL

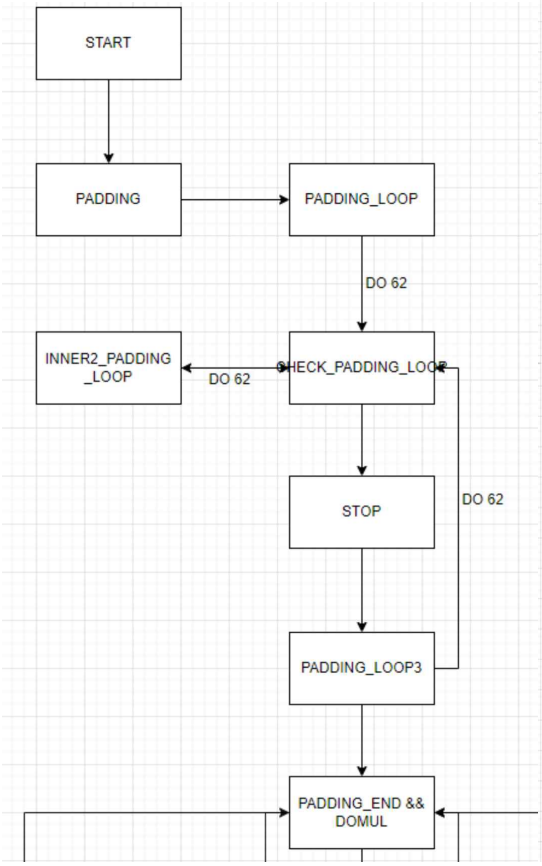
3번째나 6번째인경우 offset을 252만큼 증가시켜서 다시 연산을 진행하게 했습니다.

LAST_CAL

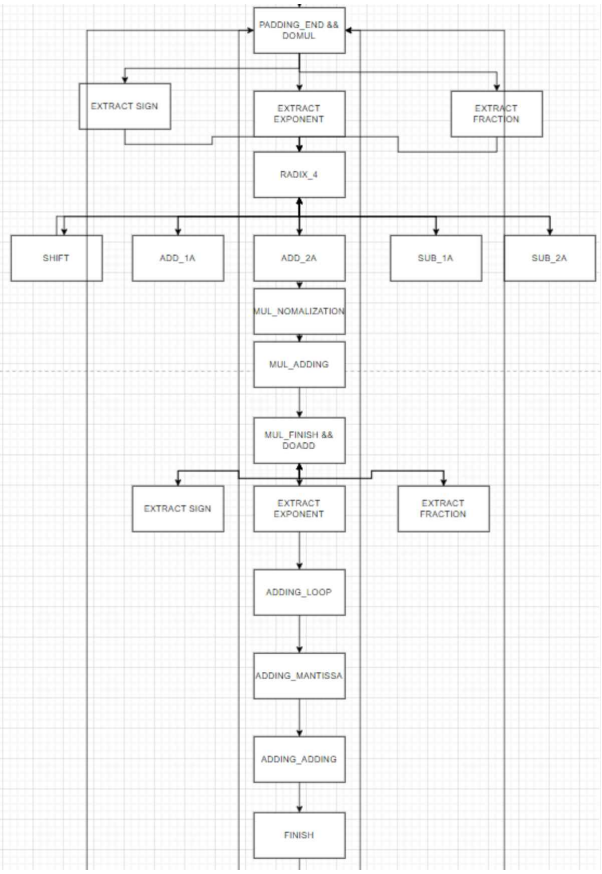
9번째 연산인 경우 sub_sampling을 바로 진행하기위에서 자기 위에위에인 원소를 가르키기 위해서 offset을 -532를 시켜주었습니다.

아래는 각 함수의 flowchart입니다.

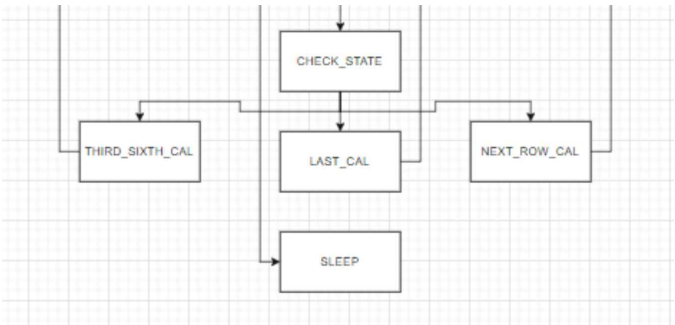
패딩하는 과정입니다.



matrix_convolution과정입니다.



sub_sampling하는 과정입니다.



함수들의 전체 flowchart입니다.

3. Algorithm

padding

패딩이 완료될 66*66의 매트릭스를 생각할때는 그 줄의 맨 끝들이 한번씩 더 호출이 되는것을 확인할 수 있습니다. 그래서 한번씩 더 호출했고, offset을 -256, -252이런식으로 해주면서 값에 LDR하게 했습니다.

booth multiplication of booth algorithm

이번 프로젝트에서 사용한 booth multiplication은

이러한 부동소수점을 곱해주는 알고리즘입니다.

보통의 MUL이라는 명령어를 사용했을때는

value=(-1)^s * (1.mantissa)* 2^(exponent-127)을 이용해서 구하지만, MUL이라는 명령어를 없이 구현할 때 쓰는 알고리즘입니다.

수업시간에 배웠던 radix-2 방식으로 이용했을경우에는 32bits면 32번의 연산이 나야하지만 radix-4인 경우는 16번만 연산을하면 가능합니다. 하지만 경우의수가 2배가 더 많아집니다. radix-4의 경우는 총 경우의수가 8가지인데, 제가 코딩한 경우의수는 총 5가지입니다. 기존이랑 중복되는게 있어서 이와 같은 5가지를 이용했 습니다. shift, add_2a, add_1a, sub_1a, sub_2a입니다.

이 알고리즘은 승수에 0을 더 추가하여 그 0부터 MSB가 될때까지 피승수와의 상호작용을 통해서 곱해지는 알고리즘입니다. 저는 LSL, LSR을 통해서 왔다갔다하면서 Y와 X+1, X, X-1을 바꿔주고 shift나 결과에 2a나 a값을 빼주고 shift 해주는 과정을 14번을 진행한 후, 14번의 2배인 28을 2의 지수로 올려서 비교를 한 후, sign, exponent, fraction을 전부 더하는 식으로 했습니다.

곱해진 수와 기존에 더해졌던 수는 똑같이 sign, exponent, mantissa를 추출해서 exponent를 비교해서 더하는 식으로 했습니다.

위 과정을 진행하면서 1에 가까운 수를 더하는 경우가 있는데 그러때 저는 cr을 0에 가까운 0x800000을 equ해서 define시켜주었습니다.

sub_sampling

제가 구현한 sub_sampling은 레지스터 두개를 카운팅 하게 해서 짝수행이 진행될때 한 레지스터를 진행시켜주고, 짝수열이 진행될때 한 레지스터를 진행시켜주었습니다. 그렇게 하면서 두개의 카운트가 모두 짝수일때 save를 시켜주었습니다.

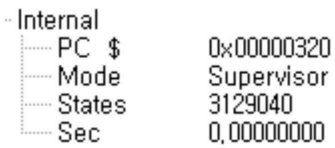
4. Performance & result

A. Performance

Code size

```
Restricted Version with 32768 Byte
Currently used: 17240 Bytes (52%)
```

Code states



조교님께서 말씀해주신 score(states*states*code size)는

1.68794966384384e+17가 나옵니다.

이것은 168794963511869440입니다.

B. Result

Padding

처음에 float로 봄으로써 제대로 값을 받았는지 확인을 했습니다.

Memory 1

Address: 0x10000000

0x10000000:	167	167	167	165	163	164	172	175	157	122	111	117	118
0x10000034:	119	122	130	136	140	141	141	141	143	144	144	143	144
0x10000068:	144	143	142	142	142	143	143	143	141	140	140	140	138
0x1000009C:	134	126	126	146	161	162	161	162	162	161	162	163	172
0x100000D0:	203	205	150	124	129	131	131	133	134	134	135	133	139
0x10000104:	139	167	167	167	165	163	164	172	175	157	122	111	117
0x10000138:	118	119	122	130	136	140	141	141	141	143	144	144	143
0x1000016C:	144	144	143	142	142	142	143	143	143	141	140	140	140
0x100001A0:	138	134	126	126	146	161	162	161	162	162	161	162	163
0x100001D4:	172	203	205	150	124	129	131	131	133	134	134	135	133
0x10000208:	139	139	166	166	165	164	163	165	172	174	155	120	110
0x1000023C:	116	118	118	122	129	135	139	141	141	142	143	144	144
0x10000270:	143	143	143	142	142	141	141	143	143	142	141	140	139
0x100002A4:	139	138	135	127	124	140	159	163	163	164	163	162	162
0x100002D8:	162	167	194	208	166	127	128	130	131	133	133	135	136
0x1000030C:	128	108	108	165	165	165	165	164	168	172	171	152	120
0x10000340:	110	116	118	117	121	129	135	139	141	142	142	142	143
0x10000374:	144	143	143	143	142	142	141	141	143	143	142	141	140
0x100003A8:	139	139	139	136	129	122	131	153	165	167	167	166	164
0x100003DC:	162	161	161	176	206	196	144	126	129	131	132	134	137
0x10000410:	130	98	69	69	165	165	165	166	167	171	171	168	152
0x10000444:	119	110	116	117	117	121	129	135	138	140	141	141	142
0x10000478:	143	143	143	143	142	140	140	140	139	141	142	142	141
0x100004AC:	139	139	139	139	137	131	123	127	148	164	168	167	166
0x100004E0:	165	163	161	159	162	189	210	176	132	128	131	132	136
0x10000514:	130	98	67	61	61	166	166	167	169	171	172	169	167
0x10000548:	152	119	109	116	117	117	121	129	134	137	139	139	140
0x1000057C:	141	142	142	142	143	141	138	137	137	137	138	140	141
0x100005B0:	140	139	139	138	139	137	132	125	127	145	161	166	166
0x100005E4:	165	164	164	163	161	158	170	203	205	156	127	130	133
0x10000618:	127	96	67	62	63	63	168	168	168	171	171	169	168
0x1000064C:	167	152	118	108	115	116	116	121	129	133	136	138	138

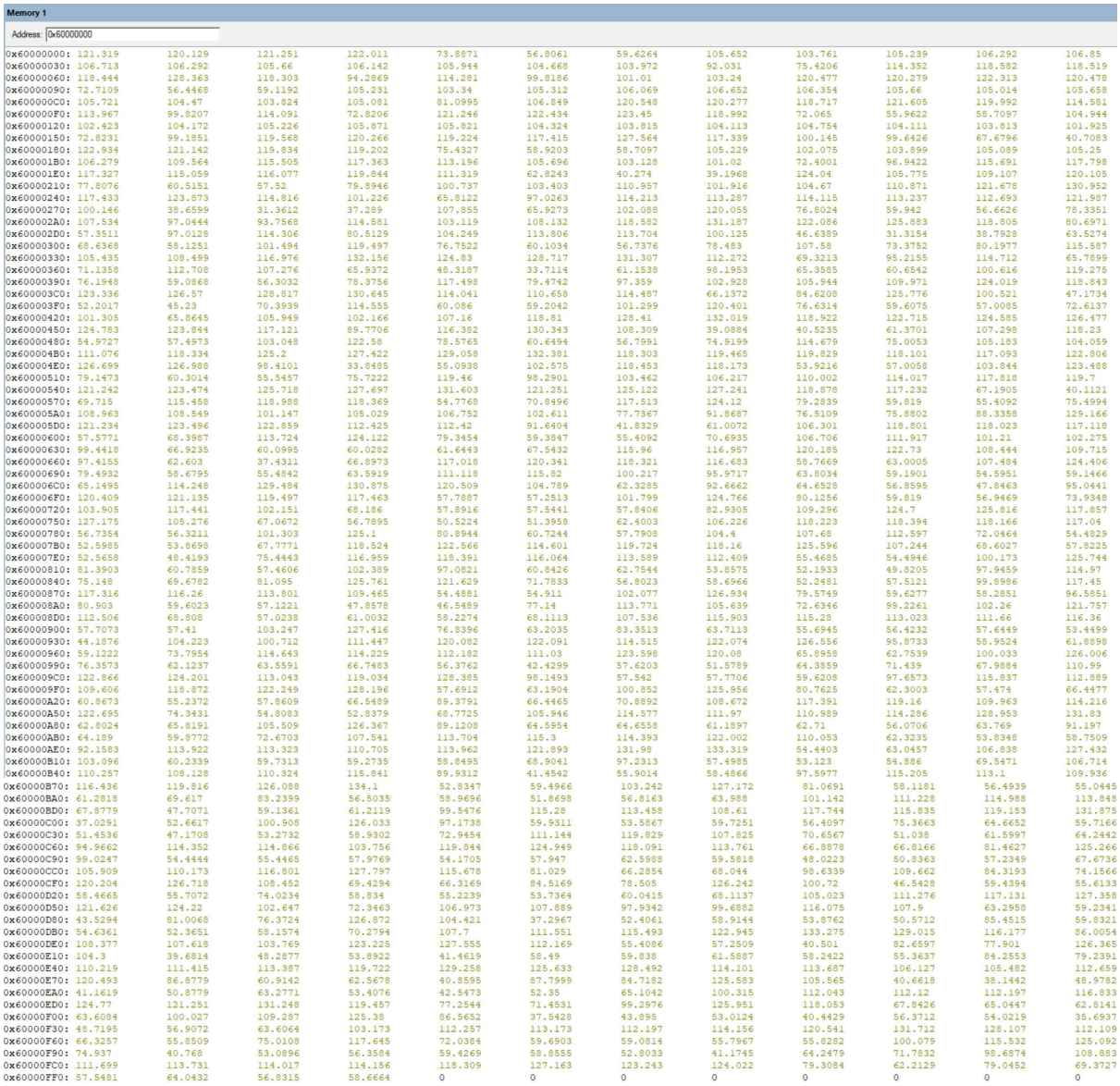
처음에 입력받은 값이 제대로 save되었고 padding또한 제대로 된것을 확인했습니다.

그리고 22개씩 unsigned int로 볼때

0x100041E0:	42C20000	42B60000	42B80000	42DC0000	42D60000	42A60000	428E0000	428E0000	427C0000	427C0000	42980000	430D0000	43360000	43180000	43250000	43370000	43270000	42CA0000	427C0
0x10004230:	428A0000	42900000	42980000	429C0000	42AE0000	42B80000	42A60000	42980000	42A00000	42AC0000	429A0000	428A0000	42B60000	42E80000	43010000	43080000	430C0000	430E0000	43100
0x10004280:	43150000	43170000	43180000	431B0000	431C0000	431D0000	431D0000	431D0000	431E0000	43210000	43260000	432D0000	43350000	433F0000	43490000	43500000	433E0000	42F20000	42D00
0x100042D0:	42FC0000	43000000	42FC0000	42F80000	42DC0000	42C60000	42BA0000	42B60000	42D80000	42E80000	42B80000	428E0000	42940000	42940000	427C0000	427C0000	42980000	430D0000	43360
0x10004320:	43250000	43370000	43270000	42CA0000	427C0000	427C0000	428A0000	42900000	42980000	429C0000	42AE0000	42B80000	42A60000	42980000	42A00000	42AC0000	429A0000	428A0000	42B60
0x10004370:	43010000	43080000	430C0000	430E0000	43100000	43130000	43150000	43170000	43180000	431B0000	431C0000	431D0000	431D0000	431D0000	431E0000	43210000	43260000	432D0000	43350
0x100043C0:	43490000	43500000	433E0000	42F20000	42D00000	42E80000	42FC0000	43000000	42FC0000	42F80000	42DC0000	42C60000	42BA0000	42B60000	42D80000	42E80000	42B80000	428E0000	42940
0x10004410:	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000

66*66이 4356이니까 198줄이 나오는것을 확인할 수 있었습니다.

matrix convolution && sub_sampling



총 32*32인 1024개가 나오는것을 확인 할 수 있지만, 값을 정확히 나오지 않은것을 확인할 수 있습니다.

값이 제대로 나오지 않아서 직접 손으로 계산하면서 해봤는데, 행렬곱을 하는 과정중 radix과정중에서 제대로 값이 추출된것이 있지만 값이 제대로 추출되지 않은것들이 있었습니다. 하지만 행렬곱을 더한 과정은 값이 제대로 나온것을 확인할 수 있었습니다.

5. Conclusion

이번 프로젝트는 뿌듯함이 있었지만 아쉬움이 좀 더 많았던 프로젝트였던것 같습니다. 먼저 padding을 하는 과정에서 3가지의 생각을 했습니다. 전체 패딩이 된 매트릭스를 저장할지, sub_sampling을 할 때 필요한 원소만 저장할 것인가, 아니면 처음부터 sub_sampling 할 때 필요한 padding만 저장할 것인가를 생각했습니다. 세번째 방법으로 진행하면서 size를 줄이려고 했는데 size를 줄이는 대신에 너무나 생각할께 많아서 시간이 부족한 저는 첫번째 과정인 가장 심플한 부분으로 진행하게 되어서 조금 아쉽습니다. 만약에 두번째 경우나 세번째 경우로 진행했을 경우, states수가 반이상이 줄어들 것 같습니다.

그 후 matrix_convolution과 sub_sampling을 같이 진행하기 위해서 짝수행과 짝수열을 카운팅 할 레지스터를 써야하는데, 레지스터가 모자랄때 sp를 이용해서 불렀다가 저장하라는 동기의 조언이 있었지만, 실습시간과 수업시간에 과제외에는 제대로 사용해본적이 없어서 r13(lr)과 r14(sp)를 그냥 카운팅하는 레지스터로 썼다는점도 매우 아쉽습니다. sub_sampling을 위해서 조교님께 여쭤본 stride라는 파라미터라는 제대로 사용해보고 싶었지만 조금 더 공부해야할 것 같습니다. multiplication이 100% 되지않아서 상당히 아쉬웠습니다. offset을 많이 건드려서 정확히 진행했다고 생각했는데 14번의 연산을 통해 진행을 하면서 덜 계산이 되었던 것 같습니다.

하지만 high level에서 느껴보지 못했던 감성을 느낄 수 있었던 좋은 과목과 프로젝트 였던것 같습니다. 어떻게 high level과 다른지, 어떻게 assembly가 되는지도 배움으로써 highlevel언어로 구현하면서 assembly관점에서는 어떻게 진행될것인지 생각해보며 컴퓨터 전공에 한걸음 더 다가가는것 같았습니다. 좋은 강의 해주셨던 조교님과 교수님들께 감사하다는 말씀 전하며 레포트 마치겠습니다.