어셈블리 프로그램 설계 및 실습

프로젝트 제안서

Term project

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

학 번: 2019202052

성 명: 김호성

1. 과제 제목&과제 목표

A. 과제 제목

Convolution and Max pooling

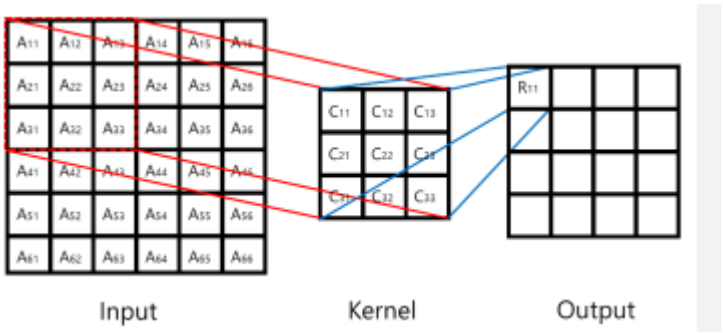
B. 과제 목표

임의의 정수 데이터로 이루어진 NxN(10 <= N <= 20) 정방행렬에 대하여 Convolution 연산과 Max pooling 연산을 수행하는 어셈블리 코드를 작성하자.

2. 각 function 별 알고리즘

A. Convolution

Stride 값만큼 평행이동 시키면서 곱셈과 덧셈을 수행하는 경우를 모두 Convolution 이라고 한다. Project의 예시를 활용하여 값을 도출한다면 다음과 같다.



R11 = A11*C11 + A12*C12 + A13*C13 + … + A31*C31 + A32*C32 + A33*C33

R12 = A12*C11 + A13*C12 + A14*C13 + … + A32*C32 + A33*C32 + A33*C33

…

R43 = A43*C11 + A44*C12 + A45*C13 + … + A63*C31 + A64*C32 + A65*C33

R44 = A44*C11 + A45*C12 + A46*C13 + … + A64*C31 + A65*C32 + A66*C33

이러한 연산을 통해 [input-kernel+1]x[input-kernel+1]사이즈의 Output 값을 도출할 수 있다. (stride의 값이 1일 때 기준이며, stride 값이 2이상 일 경우에는 stride-1 값을 뺀 만큼의 size가 나온다.

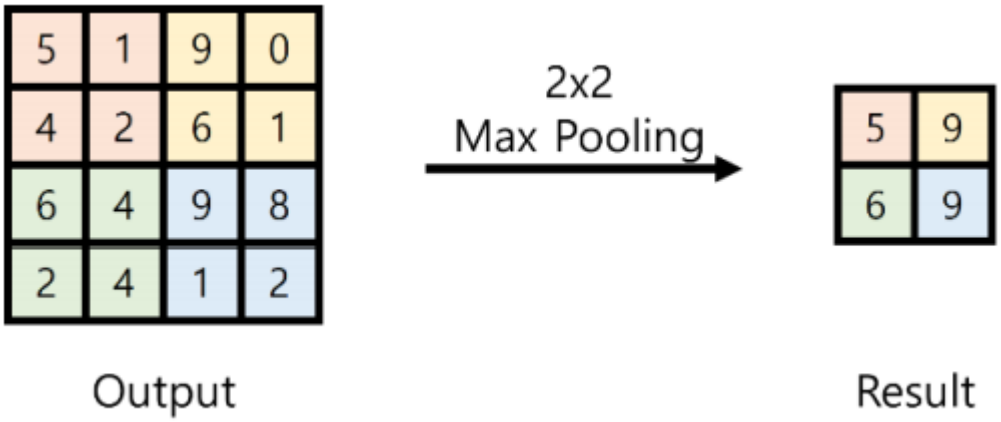
즉 R값을 일반화 한다면 다음과 같은 식이 완성된다.

Ryx = Ayx*C11 + Ay(1+1)*C1(1+1) + … + A(y+i)(x+j)*C(1+i)(1+j)

- x와 y는 행렬의 가로축과 세로축임.
- i와 j는 반복문의 count임.
- i와 j는 각각 Kernel 행렬의 각 축의 길이 – 1한 값까지 반복됨.

B. Max pooling

Max Pooling은 Activation map을 MxM의 크기로 잘라낸 후, 그 안에서 가장 큰 값을 뽑아내는 방법이다. 아래 그림은 프로젝트 제안서의 예시로 나온 그림이다.



이처럼 붉은 배경을 가진 숫자들 중 가장 큰 값인 5를 도출.
노란 배경을 가진 숫자들 중 가장 큰 값인 9를 도출.
반복하여 6과 9를 도출하여 Result 행렬을 완성한다.

3. 일정

해당 project를 진행할 계획을 표 또는 그림으로 작성

	11	12	13
제안서			
코드작성 및 검증			
결과보고서			

4. 예상되는 문제점

A. Convolution

행렬 값이기 때문에 구조 상 2차원 배열과 동일하다고 할 수 있다.
그렇기 때문에 반복문을 중첩하여 구하는 방법이 있고, 2차원 배열을 1차원 배열로 풀어낸 후 Input의 행의 size의 값에 따라 범위를 조정하여 Convolution연산을 수행하는 방법이 있다. 어떤 방법을 사용했을 때 State의 값이 적게 나오는지 확인하기 위하여 두 가지 방법으로 코드를 구현해야 한다고 생각한다.

- Convolution 중 발생하는 곱하기 연산.

MUL 명령어를 사용할 수 없으므로 다른 곱셈방법을 사용해야 할 것이다. 아마도 Radix 2 방법과 Radix 4 방법이 존재하는 Booth Multiplication을 사용해야 하는데, Booth Multiplication에서 Radix 4가 Radix 2보다 2배 정도 빠른 속도를 가지고 있다. 그러나 Radix 4의 경우 Radix 2보다 다소 어렵다는 단점이 있기 때문에 우선 코드 구현을 목표로 하기 위해 Radix 2를 사용하여 코드를 구현한 이후 여유가 된다면 Radix 4로 구현할 예정이다.

B. Max pooling

Max Pooling의 연산의 경우 Convolution연산으로 나온 Output값을 Max pooling으로 나눌 때 딱 떨어지게 나눠지지 않을 수도 있다. (ex. output = 5X5, Max pooling = 2X2) 이 경우 Result 값을 3X3으로 해야 하는지, 2X2로 한 후 범위에 들어오지 않은 값들을 전부 버릴지 생각해줘야 한다.

5. 검증전략

A. Convolution

Convolution의 경우 우선 6x6일 때를 기준으로 코드를 구현하고자 한다. 또 test case의 값을 이론상 제안서에 나온 예제의 값과 동일하게 나올 수 있도록 맞춘 후 같은 값이 나오는지 확인할 예정이다.

5	1	9	0
4	2	6	1
6	4	9	8
2	4	1	2

B. Max pooling

Convolution연산을 끝낸 값(위의 사진과 동일)을 input으로 한 다음 Result Max pooling을 따로 줄 것이다. 현재 프로젝트는 총 파트가 두 개 이므로, 우선 Max pooling과 Convolution을 각기 다른 파일에서 진행한 후 합치는 작업을 거칠 것이다.

6. 구체적인 프로젝트 진행의 흐름

A. Max pooling 알고리즘 C로 구현

B. Convolution 알고리즘 C로 구현

C. A와 B를 합침. (Code style C)

D. C를 어셈블리어로 변환

E. 결과보고서 작성

총 순서를 5가지로 나눈 이유는 다음과 같다.

- Max pooling 알고리즘이 Convolution 보다 구현하기 쉽기 때문에.
- 단계를 나누어 문제를 작게 쪼갠 이후 풀기 위해.
- 어셈블리어의 경우 메모리와 레지스터를 서로 참조하면서 연산을 진행해야 하는 어려움이 존재해 자칫 코드 구현에 있어 어려움이 존재할 것이라고 판단하여, 알고리즘 부분에서만 C로 구현한 이후, C style 언어가 정상적으로 동작하는 것을 확인한 후, 어셈블리어를 구현하면, 보다 쉽게 어디서 문제가 발생했는지 파악하기 쉬워서.