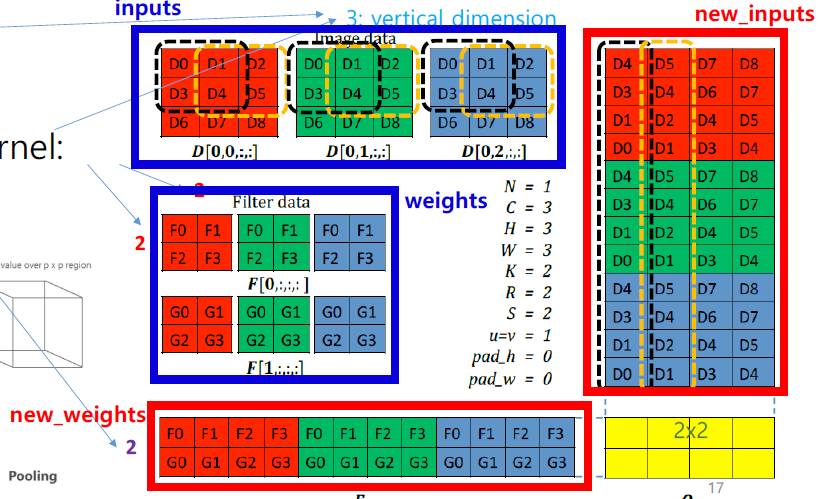
**Hardware System Design**

**LAB07**

**이다운 2016-13919**

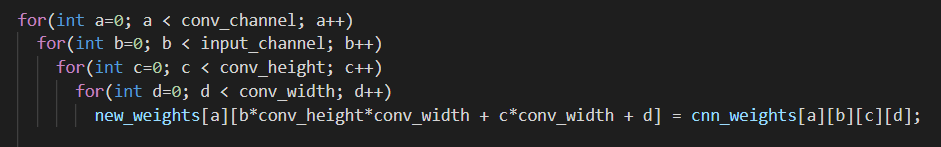
이번 LAB에서의 핵심목표는 Matrix multiply 연산과정을 보다 효율적으로 하기위해 convolution이 가능한 matrix form으로 바꿔주는 것으로 이 과정을 Convolution Lowering이라 한다.



위 그림과 같이 weight matrix (Filter)와 inputs matrix를 각각 convolution 연산을 위한 하나의 matrix로 변환하는 것이 목표이다.

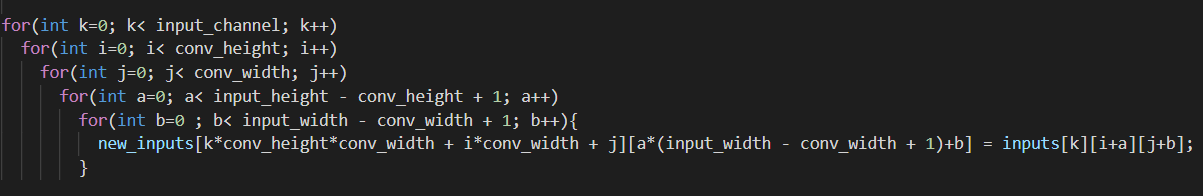
**1. Code**

**1.1 new\_weights**



첫번째 페이지의 그림의 참조하면 new\_weight matrix의 행들은 weights의 channel로 나눠지고 각 행은 input\_channel 순서로 weight 값이 순서대로 나온다. 즉 행의 개수는 conv\_channel (weight channel) 이고 열의 개수는 input\_channel \* (한 filter의 원수 개수) = input\_channel \* conv\_height \* conv\_width 이다.

그러므로 위 스크린샷처럼 conv\_channel, input\_channel, conv\_height, conv\_width 순서대로 for문을 돌면서 new\_weight 원소 값을 채워준다.

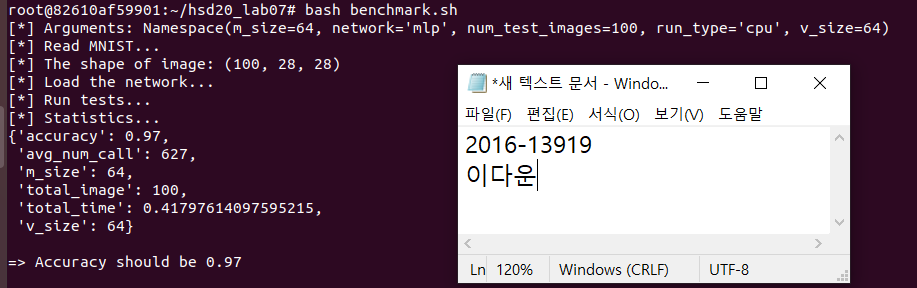
**1.2 new\_inputs**

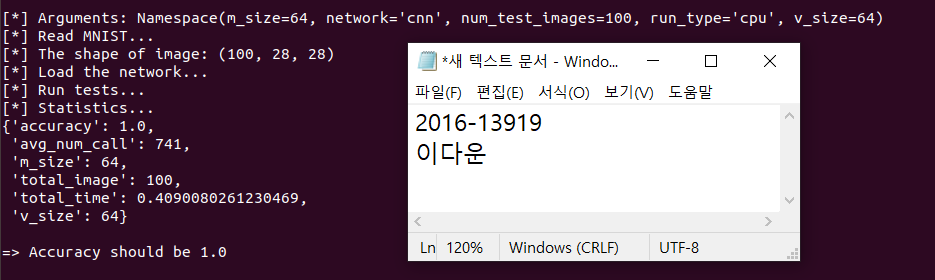
new\_inputs도 new\_weigth와 비슷하다. 마찬가지로 new\_inputs 그림을 참조하면 행은 크게 input\_channel로 나눠지고 각 나눠진 부분은 weights의 원소 개수만큼 행을 가진다. new\_inputs의 열은 실질적인 곱의 횟수를 의미하므로 한 Filter가 한 input에 Filtering하는 횟수가 된다.

한 Filter가 한 input에 Filtering하는 횟수는 input과 weigths의 넓이 및 높이의 차이에서 1씩더한값의 곱이므로 1.1 new\_weight와 마찬가지로 input\_channel, conv\_heigth, conv\_width, input\_height – conv\_height +1, input\_width – conv\_width +1 만큼을 순서대로 for문을 돌면서 new\_input의 행 -> 열 순서로 채워간다.

new\_inputs의 행-> 열 순서대로 원소를 보면 input이 filtering 하는 각 연산에서의 곱 순서라는 것을 알 수 있다. 그 순서에 맞게 inputs 원소가 new\_inputs에 초기화 되도록 for문을 구현하였다.

**2. Result**





benchmark를 실행한 결과 mlp 모드에서는 정확도가 0.97, cnn모드에서는 1.0이 나오는 것을 나왔다. run\_tpye = ‘fpga’ 인 경우 에는 에러가 나오기 때문에 사진을 첨부하지 않았다.

**3. Discussion**

첫 페이지의 Convolution Lowering 그림을 보면 new\_inputs의 원소가 역순으로 되어있는 것을 확인할 수 있다. 하지만 실제로의 구현에서는 역순으로 하면 안되고 실제로 곱의 정방향 순서대로 new\_inputs matrix를 채워야 한다.

new\_inputs, new\_weights 모두 다중 for문을 사용하여 연산 속도가 느릴 것 같지만 실제로 연산 횟수는 convolution matrix의 원소 개수만큼만 하므로 속도 저하에 영향을 미치지 않는다고 본다.