어셈블리프로그램 설계 및 실습

Term Project – rev1 (241129)

과제 문의

GitHub Issue를 이용하여 질문

github.com/metr0jw/2024-KWU-Assembly-Programming-Term-Project

담당 교수: 이형근 교수님 (컴퓨터정보공학부)

담당 조교: 이지운, 황정원

"꿈을 가지십시오. 그리고 정열적이고 명예롭게 이루십시오."

Abstract

이번 학기 설계 과제는 MNIST 숫자 데이터에 대한 이미지 확대 연산을 ARM 어셈블리로 구현하는 것이다. 입력 데이터는 20x20 크기의 정방행렬이며, 각 픽셀값은 IEEE 754 Single Precision 형식으로 저장되어 있다. Bilinear Interpolation 알고리즘을 사용하여 입력 이미지를 80x80 크기로 4배 확대하는 어셈블리 코드를 작성해야 한다.

우수 과제 평가는 두 가지 기준으로 이루어진다. 첫째, 결과물의 품질은 peak signal-to-noise ratio (PSNR)로 평가하며, 이 값이 클수록 더 좋은 품질을 나타낸다. 둘째, 연산 성능은 코드 크기 (code size)와 상태 (state)의 곱으로 측정하며, 이 값이 작을수록 우수한 성능을 의미한다.

과제 수행 전 상세 명세서를 철저히 검토하여 모든 요구사항을 정확히 이해하고 구현하도록 한다.

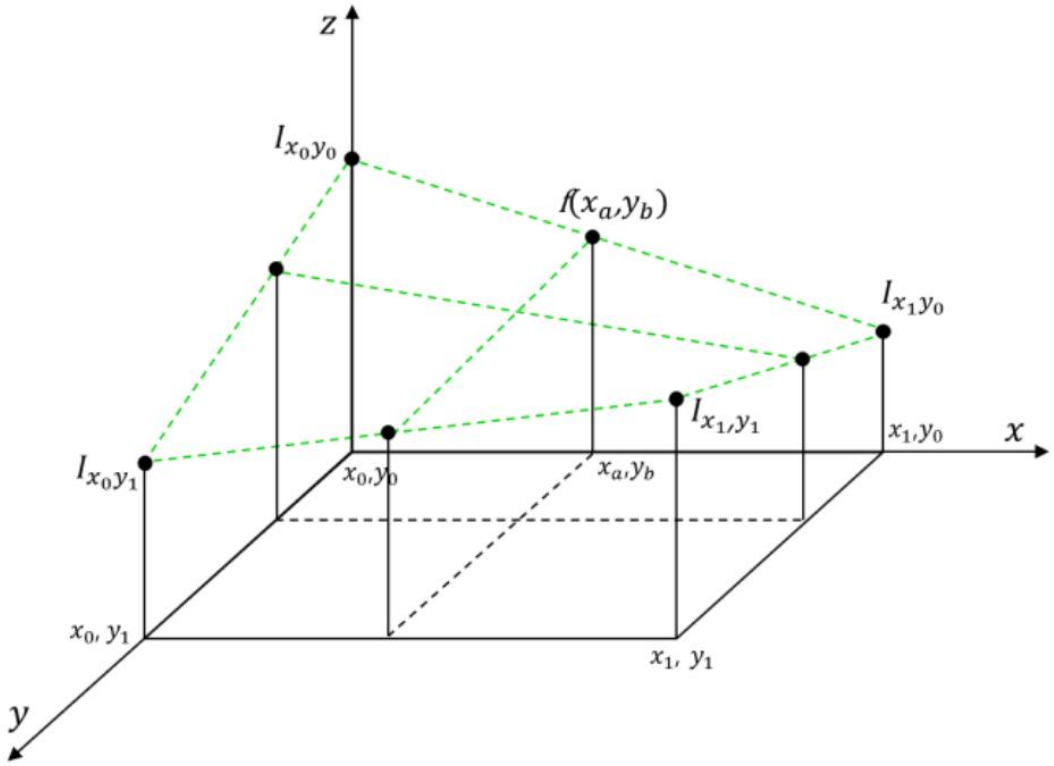

Figure 1. Bilinear interpolation

- **<u>구체적인 설계 사양은 업데이트 될 수 있습니다. 업데이트 될 경우, KLAS 공지사항과 GitHub를 통해 공지될 예정입니다.</u>**

1. Specification

- Scaling up size
  - 행과 열의 각 4배 (20x20 to 80x80)
- Boundary Exception
  - 20x20으로 주어지는 이미지의 마지막 행, 마지막 열의 데이터는 padding을 적용하여 interpolation 수행
  - Padding하여 생성될 데이터와 위치가 가장 가까운 데이터의 값으로 복사

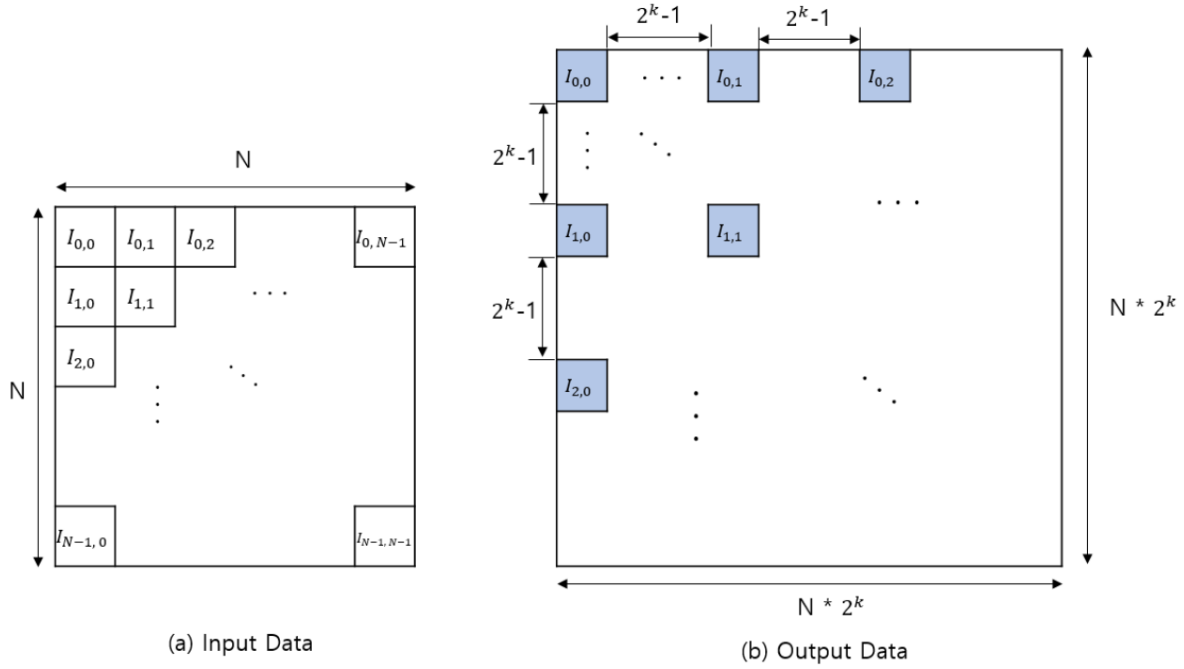

Figure 2. Input and Output. (a) Shape of input data, 20x20. (b) Shape of output data, 80x80 (k=2).

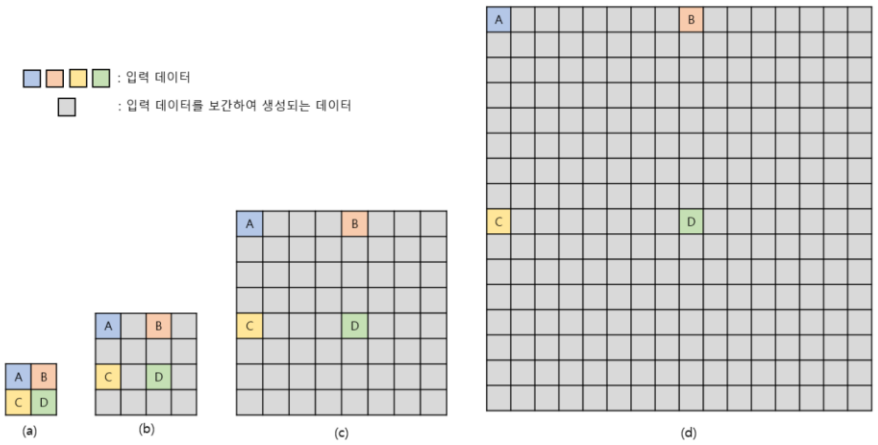

Figure 3. An example of bilinear interpolation. (a) 2x2 input matrix. (b-d) Matrix resulting from interpolation of the rows and columns of (a) at 2, 4, and 8 times the size, respectively.

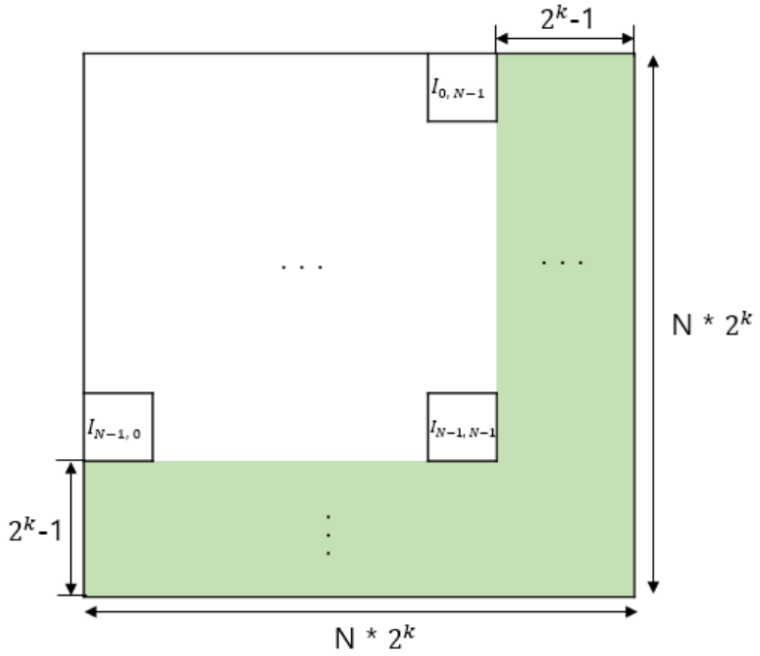

Figure 4. Location of pixels created by padding.

- Skeleton code 관련 안내 (skeleton.s)
  - Skeleton code는 과제 수행시 편의성을 위해 참고 목적으로 제공되는 코드입니다. 아래 내용을 제외한 다른 부분은 모두 수정 가능합니다.
  - **수정 금지 사항**
    - **end_program 서브루틴**
    - **source_data의 label 이름**
      - **단, 0.txt로 제공되는 파일은 숫자 0에 대한 이미지이므로 나머지 1-9.txt에 대해 수행하는 할 때는 1.txt, 2.txt 등과 같이 수정하여 과제를 수행하도록 한다.**
    - **ResultBuffer label 내용 전체**
      - **ResultBuffer는 80x80으로 보간된 이미지가 저장될 위치로, 본 과제에서는 0x10000000 주소가 이용된다. 디버깅 관련 안내는 챕터 2 디버깅 안내를 확인하도록 한다.**
- 프로그램의 종료
  - end_program subroutine을 호출하여 수행한다.
- 데이터 불러오기 및 저장
  - 20x20 이미지는 source_data를 통해 불러온다.
  - Data/downsampled 폴더에 있는 0.txt, 1.txt, …, 9.txt는 각각 숫자 0부터 9까지의 숫자 손글씨 MNIST 데이터셋의 일부이다. 각 텍스트 파일에는 20x20의 손글씨 데이터가 포함되어있다.

- 예: LDR R4, =source_data

○ 80x80 결과 이미지는 ResultBuffer에 1 word 단위로 저장하도록 한다.

- 예: LDR R5, =ResultBuffer; STR R10, [R5, R9]

○ 이미지의 저장 순서는 아래 예시와 같도록 한다.

- 예시
  - 주어진 이미지
    [[0x00 0x01 0x02]
    [0x03 0x04 0x05]
    [0x06 0x07 0x08]]
  - 저장
    00 01 02 03 04 05 06 07 08
  - 예시는 이미지의 row-column 순서의 이해를 위한 것으로, endianess은 무시할 것. STR로 저장되는 그대로 저장하도록 한다.

2. 디버깅 안내 [필독]

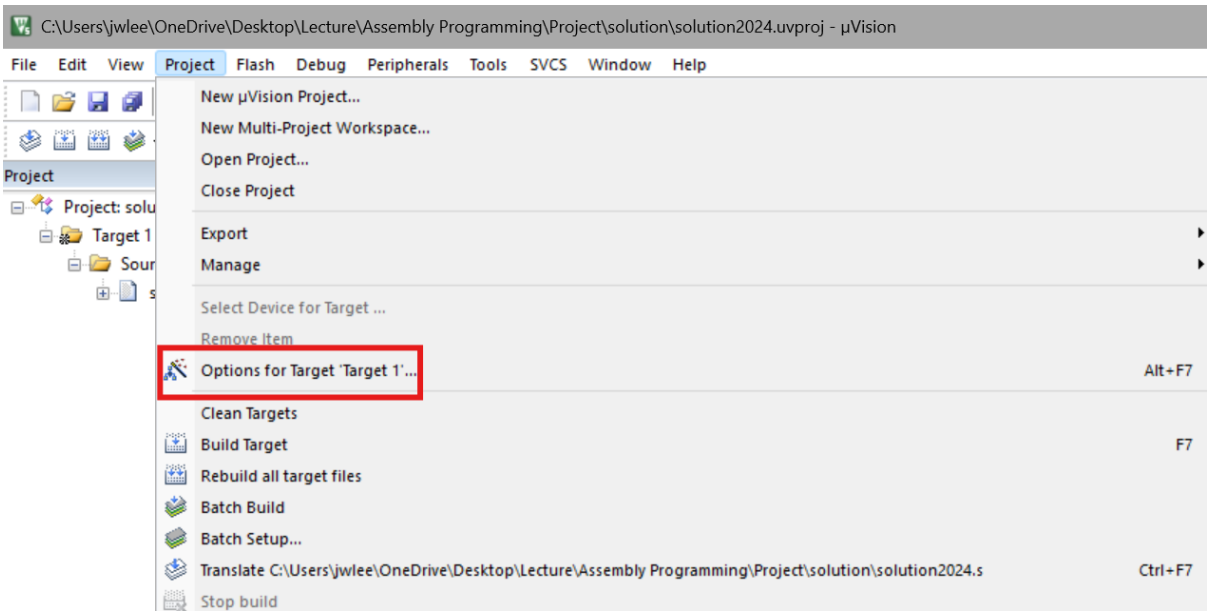

Figure 5. Memory configuration (1/3).

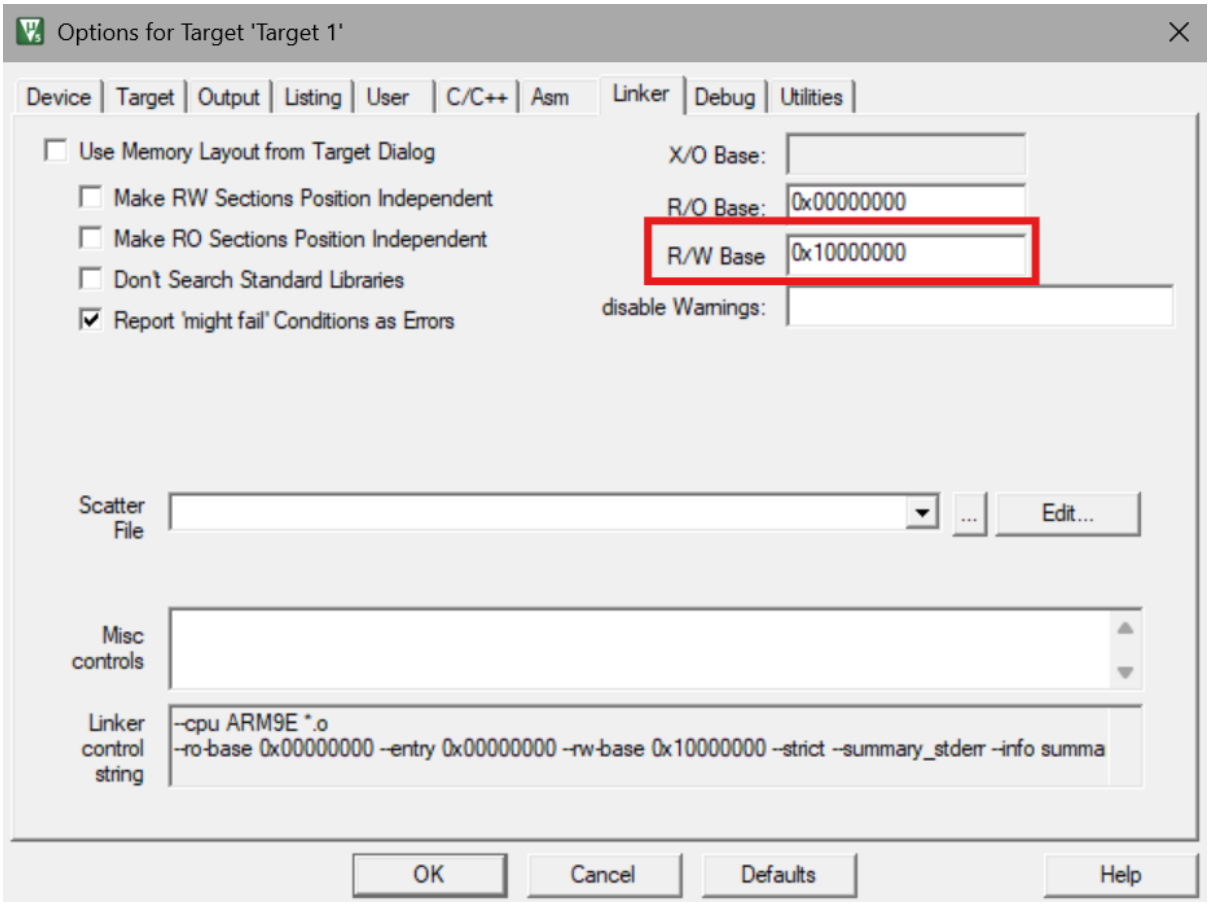

Figure 6. Memory configuration (2/3).

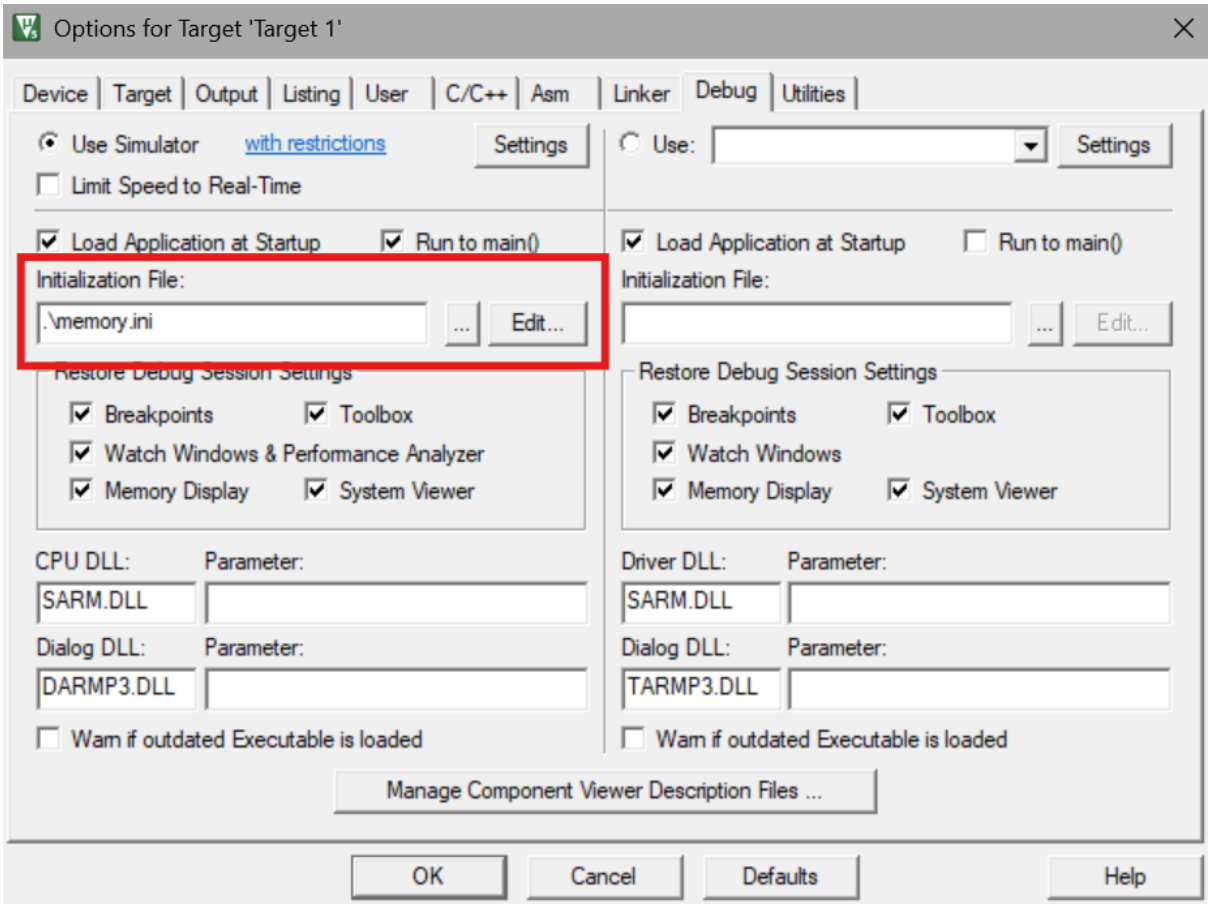

Figure 7. Memory configuration (3/3).

- **<u>Figure 5-8에 따라 메모리 관련 설정 후 과제를 수행하도록 한다.</u>**

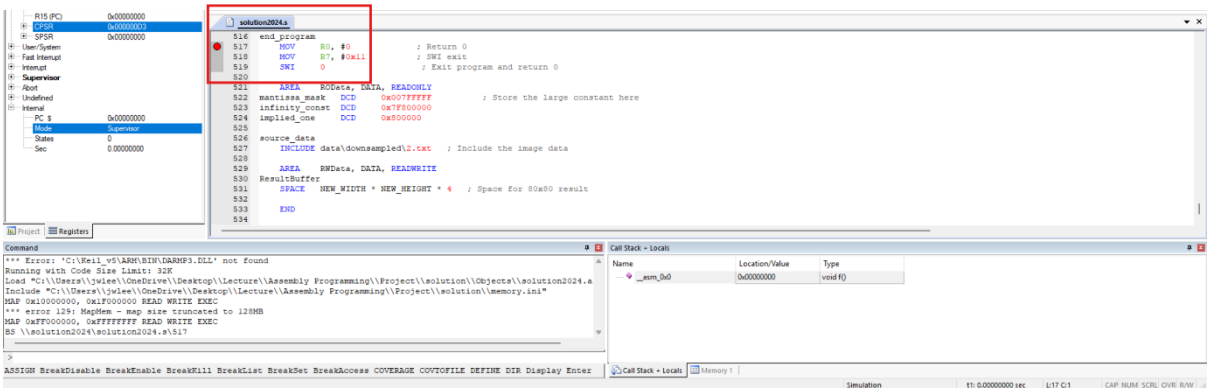

Figure 8. A simple way to check the result.

- Line number의 왼쪽을 클릭하여 breakpoint를 설정할 수 있다.
- **<u>end_program subroutine 부분에 breakpoint를 지정하여 최종 결과를 확인하도록 한다. end_program subroutine부터 값이 바뀌는 것은 고려하지 않는다.</u>**

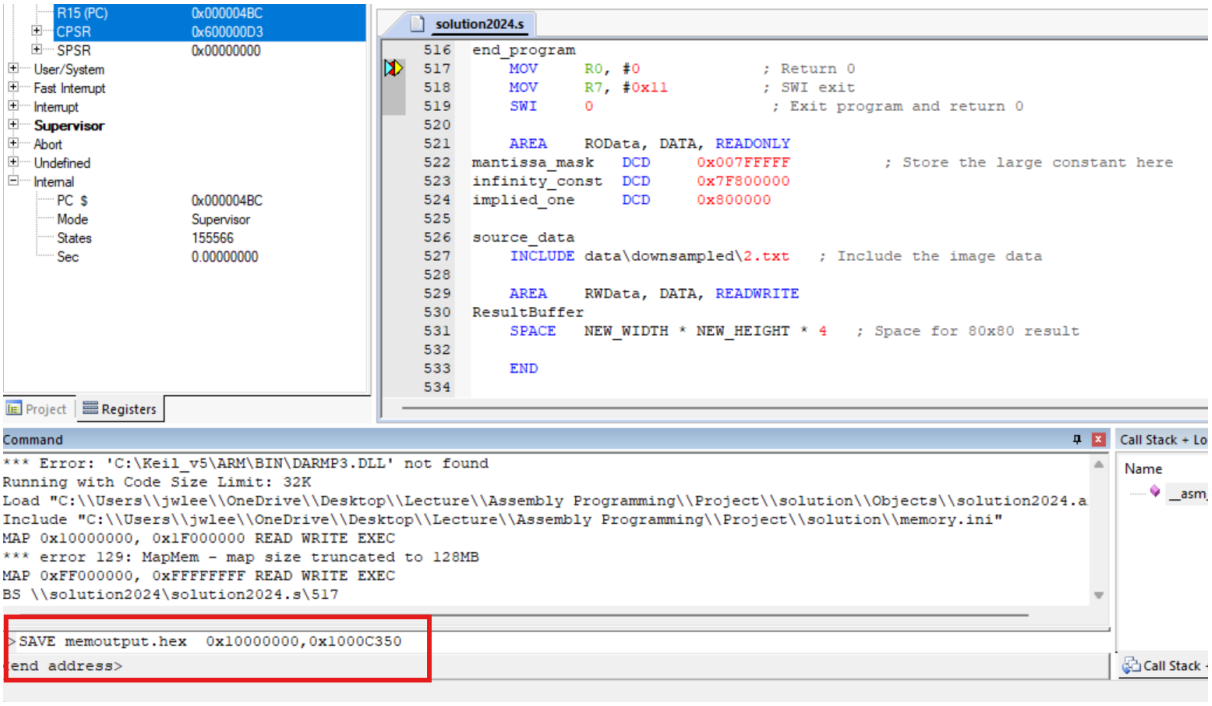

Figure 9. How to check the interpolated result (1/4).

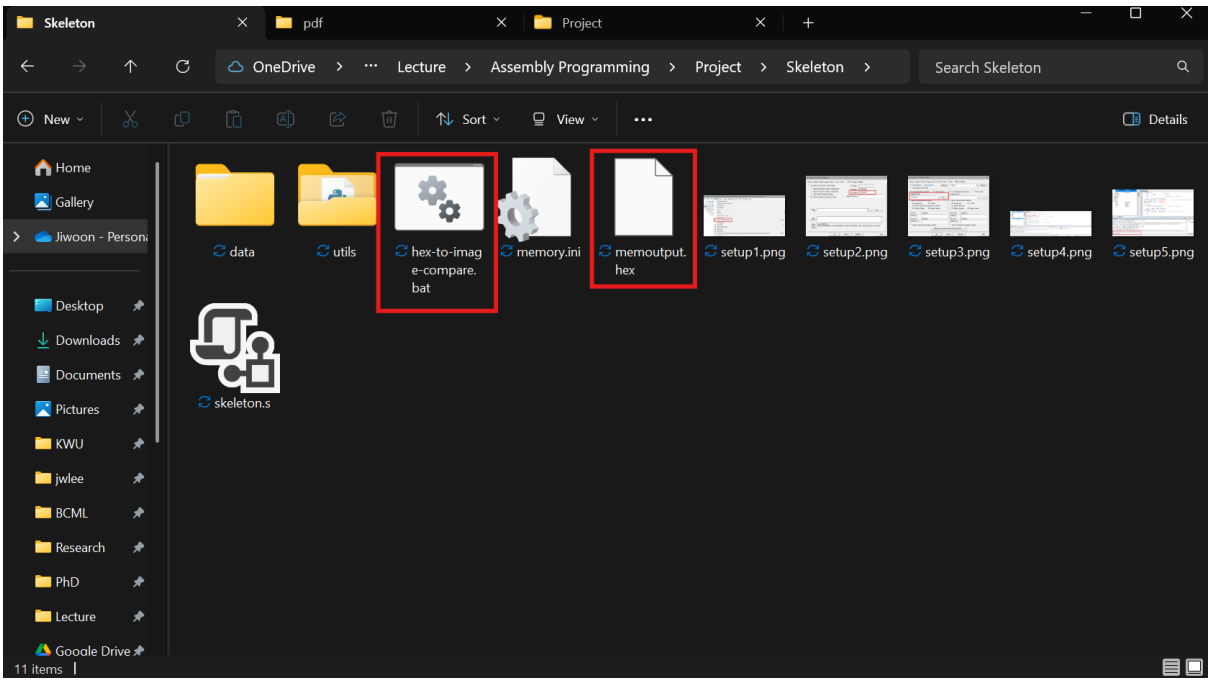

Figure 10. How to check the interpolated result (2/4).

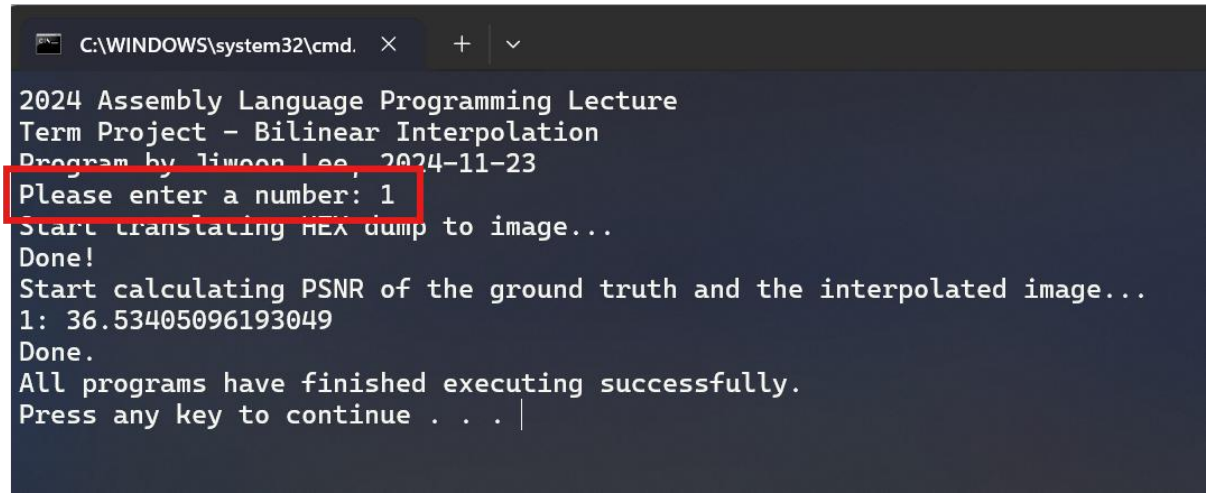

Figure 11. How to check the interpolated result (3/4).

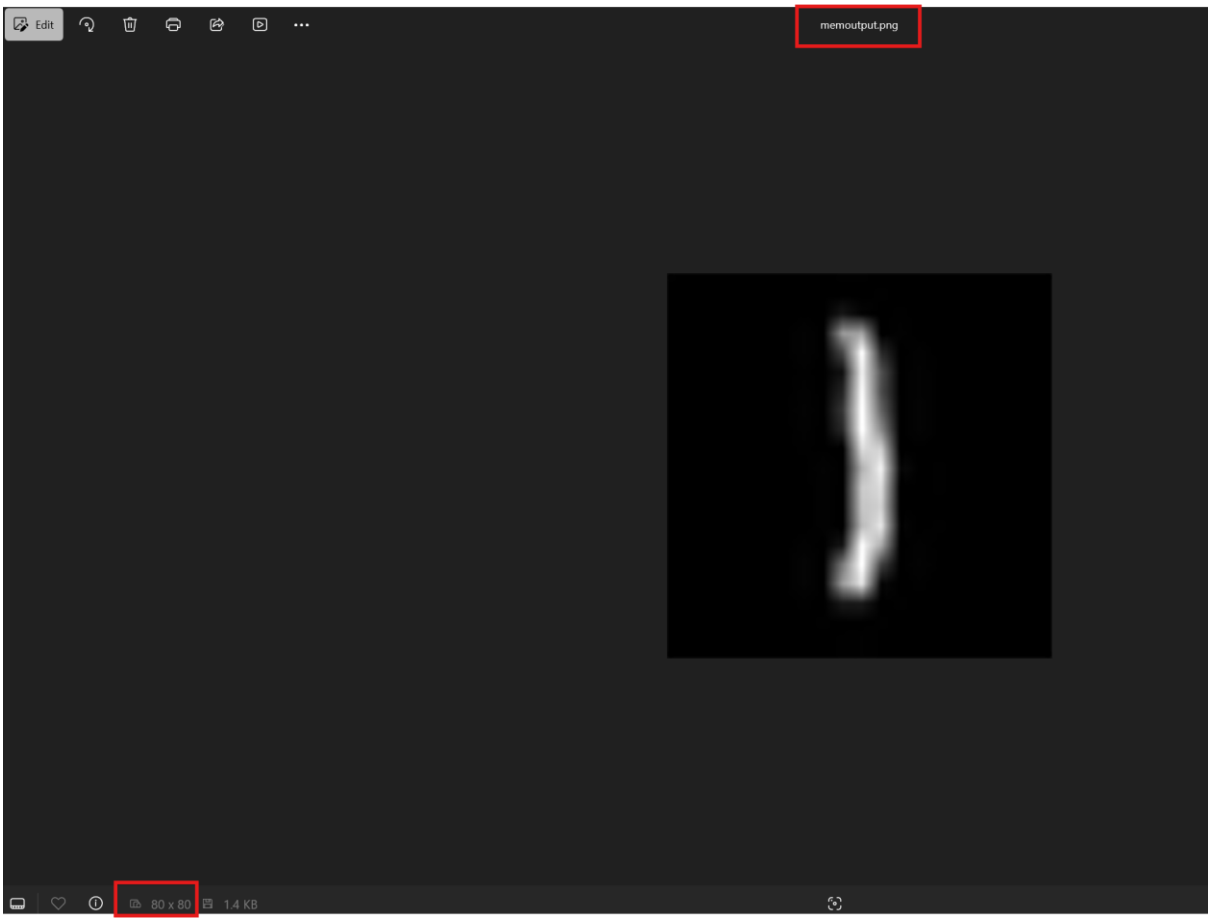

Figure 12. How to check the interpolated result (4/4).

- Keil uVision에서 보간 프로그램을 실행 후 메모리 dump를 이미지로 확인하고 싶을 때, Keil uVision에서 디버그가 실행 중인 상태에서 좌측 하단 command에 아래 커맨드를 입력하여 dump 파일을 생성한 후, 과제 파일에서 제공되는 "hex-to-image-compare.bat"을 이용하여 결과를 확인한다.
- **`SAVE memoutput.hex  0x10000000,0x1000C350`**

- 만일 위 커맨드가 작동하지 않는 경우, 아래 사이트에서 커맨드를 복사하여 붙여넣기 한 뒤 주소만 변경하여 dump 파일을 생성하도록 한다.
  - https://developer.arm.com/documentation/101407/0541/Debug-Commands/SAVE
- 위 예시의 경우 숫자 1에 대한 이미지를 보간한 것이므로, "hex-to-image-compare.bat"을 실행했을 때 1을 입력하여 원본 이미지와의 PSNR을 비교한 것이다.
  - PSNR이란 - https://kr.mathworks.com/help/vision/ref/psnr.html
    - 원본 영상과 복원된 영상의 품질을 측정하기 위해 사용하는 지표로, 단위는 dB를 이용한다. PSNR이 높을수록 복원된 영상의 품질이 높음을 의미한다.

3. 평가 기준

1. 코드 구현 (50%)

a. 실행 가능한 코드 제출 필수

b. 실행 불가 시 0 점 처리

c. Integer 기준 오차 ±1

   i. Rounding 을 별도로 수행하지 않아도 문제 X

d. 지정된 파일명 준수 (project.s)

e. 시스템 콜을 이용한 프로그램 종료

2. 보고서 작성 (50%) – 보고서에 코드 첨부하지 말 것

a. 문제 설명 (Problem Statement)

b. 구현 방법 (Algorithm)

c. 실행 결과 (Result)

d. 토론 및 결론 (Discussion and Conclusion)

e. (선택) 감사의 글 (Acknowledgement)

f. 참고문헌 (References)

g. 지정된 파일명 준수

3. 추가 점수 (반영 비율 미정)

a. 결과물의 품질 (PSNR)
    a. Rounding 처리 필요
b. 연산 성능 (code size, state)
c. 질의응답 참여
    a. GitHub Issue 에 올라온 질문 및 토론 참여도

    github.com/metr0jw/2024-KWU-Assembly-Programming-Term-Project
d. Project 결과 발표
    a. 발표 관련 내용은 추후 공지 예정.

4. 제출 파일

- 아래 파일들을 zip 형태로 압축하여 제출
- 1분반-202420XXXX-김ABC-Project.zip
    - 소스코드 파일 (project.s)
        - Skeleton.s로 제공되는 파일의 이름은 project.s로 수정하여 이용하도록 한다.
    - 보고서
        - 1분반-202420XXXX-김ABC-Project.pdf
- ***DATA, UTILS 폴더 및 보간 결과물은 용량이 큰 관계로 제출 파일에 포함하지 않도록 한다.***

5. 유의사항

- 코드는 독립적으로 실행되지 않아도, 부분 점수를 부여함.
    - 단, 문법의 오류 등으로 인해 실행되지 않는 경우는 점수를 부여하지 않음.
    - 환경의 차이로 인해 실행되지 않는 경우 감점될 수 있음.
- 코드 내 주석 처리 필수
- 특별히 명시된 경우를 제외하고 메모리 주소는 자유롭게 선택 가능
- 보고서 구성 및 소스코드명은 지정된 양식을 준수할 것

- 단, 보고서는 챕터 추가 필요에 따라 추가 가능
- 과제 공지에 첨부되어있는 memory.ini, data 폴더, utils 폴더를 이용하여 과제를 수행할 것

6. 제출 기한

- 2024 년 12 월 9 일 (월) 23 시 59 분 59 초
- 2024 년 12 월 10 일 (화) 11 시 59 분 59 초
    - 지연 제출, 정상 채점 점수의 0.5 배 부여
- KLAS 를 통해 제출할 것

Appendix

1. Pseudo code

a. 과제의 난이도를 고려하여, c로 작성한 Bilinear interpolation의 일부 코드를 제공하며, example 폴더에서 확인 가능합니다.

2. Special cases in IEEE 754

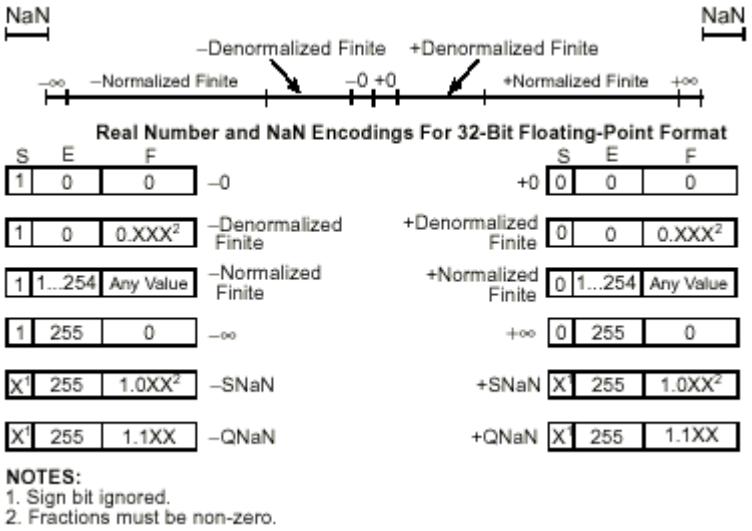

Figure 13. Special cases in IEEE 754. (https://www.randelshofer.ch/fhw/gri/float.html)

a. Zero
   i. 0x00000000 (Positive)
   ii. 0x80000000 (Negative)

b. Infinity
   i. 0x7F800000 (Positive)
   ii. 0xFF800000 (Negative)

c. NaN (Not a Number)
   i. 0x7F800001과 0x7FBFFFFF 사이 또는 0xFF800001과 0xFFBFFFFF 사이 (Signaling NaN, 잘못된 연산을 의미)
   ii. 0x7FC00000과 0x7FFFFFFF 사이 또는 0xFFC00000과 0xFFFFFFFF 사이 (Quiet NaN, 예외를 발생시키지 않고 모든 연산에 NaN을 전파)

d. **하지만 본 과제의 경우, 정상적으로 연산이 이루어졌을 경우 Zero를 제외하고는 발생하지 않습니다. 따라서, Infinity와 NaN에 대해서는 채점이 이루어지지 않습니다.**

3. Dataset

a. 데이터셋은 28x28 크기의 숫자 손글씨 데이터셋인 MNIST를 이용하여 과제를 출제했습니다 (https://yann.lecun.com/exdb/mnist/).

b. 28x28 크기의 이미지를 20x20으로 다운샘플링한 뒤 80x80으로 업스케일링

하였으며, 이를 ground_truth로 설정하였습니다.

c. 80x80으로 업스케일된 이미지를 다시 20x20으로 다운스케일하여, 이를 downsampled로 설정하였습니다.

d. Interpolated에 나와있는 이미지는 조교가 구현한 코드를 통해 보간된 이미지입니다.

4. Subroutine and Registers

a. 다양한 subroutine을 이용하여 연산을 수행하는 과정에서 레지스터가 모자랄 수 있습니다. 이를 위해 일관적인 레지스터 사용이 중요합니다. 예를 들어서, 본 과제에서는 두 피연산자의 multiplication과 addition/subtraction이 주로 수행됩니다. 여기서 R7, R8에 항상 피연산자를 넣어 subroutine으로 넘어가 연산을 수행하고, 그 결과값을 항상 R0를 통해 반환하도록 하면 별도로 stack을 사용하지 않고도 레지스터 사용 및 메모리 액세스를 적게 하여 과제를 구현할 수 있습니다.

5. 전반적인 프로그램의 실행 순서 (241129 추가)

a. Keil uVision에서 Breakpoint 설정 및 디버깅 실행
   i. F5 (Step) 후 Breakpoint 도달 여부 확인
   ii. 만약 Breakpoint에 도달하지 않는다면 프로그램 로직에 문제가 있는 것으로, 코드를 수정할 것

b. SAVE memoutput.hex 0x10000000,0x1000C350 입력 (Figure 9 확인)
   i. 이 단계에서 프로젝트 폴더에 memoutput.hex라는 텍스트 파일 생성됨

c. Hex-to-image-compare.bat 실행
   i. 이 단계에서 memoutput.hex 파일이 memoutput.png 이미지 파일 생성됨

d. PSNR 확인 및 육안으로 이미지 정상 출력되었는지 비교