마이크로프로세서 HW3

2013104371 김상민

MOV5를 완성하고, JZ를 이용한 While구문으로 5x5 행렬 곱 연산 및 성능 향상을 이루어내는 과제였습니다.

편리하게 while구문을 만들기 위해 JZ의 조건을 변경한 JZ1, JZ2, JZ3을 추가하였습니다.

편리한 MOV2사용을 위해 MOV2의 조건을 변경한 MOV2\_를 추가하였습니다.

MOV2를 참고하여 MOV5를 만들었습니다.

성능 향상 전

우선 A행렬의 첫 행과 B행렬을 곱하여 C행렬의 첫 행을 완성시키고 모두 메모리에 저장

A행렬의 나머지 행을 메모리 저장하면서 B행렬과 곱하여 C행렬의 나머지를 완성 및 연산 결과값을 메모리에 저장.

JZ가 빈번히 사용될수록, 메모리 저장이 잦을수록 또는 중복되면 clock 수가 늘어날 것으로 판단했습니다. 때문에 행렬곱셈 연산 이외에는 JZ를 최소화 시켜 보았습니다. 또한, 메모리와 레지스터를 왕래하며 값을 옮기면 성능이 저하될 것으로 생각하여 행렬에 대한 정보만 메모리에 저장만 시킨 후 연산 시에는 레지스터 내에서 수행했습니다.

성능 향상 후

A, B행렬을 MOV2와 ADD를 반복 수행하여 모두 저장 시킨 이후 JZ를 통해 A, B행렬의 곱셈 결과값만을 C의 위치의 메모리에 저장하였습니다.

file4.bin

0011000000000001 int r0=1; (MOV3 R0 #1)

0011000100000001 int r1=1; (MOV3 R1 #1)

0011001000000010 int r2=2; (MOV3 R2 #2)

0011010100001010 int r5=10; (MOV3 R5 #10)

0101010100100000 r5 = r5-r2; (SUB R5 R2)

0111010100100000 r5 = r5\*r2 (MUL R5 R2) <- r5--를 구현하기 위함

while( r5 ){

1000011000000000 int r6=r0+r1; (MOV4 R6 R0)

0100011000010000 (ADD R6 R1)

1000000100000000 r1=r0; (MOV4 R1 R0)

1000000001100000 r0=r6; (MOV4 R0 R6)

0101010100100000 r5--; (SUB R5 R2) <- R5를 2배 해서 2 감소니까 --와 동일

0110010111111010 } (JZ R5 -6)

file3.bin

0011000000000001 int r0=1; (MOV3 R0 #1)

0011000100000001 int r1=1; (MOV3 R1 #1)

0011001000000010 int r2=2; (MOV3 R2 #2)

0011010100001010 int r5=10; (MOV3 R5 #10)

0101010100100000 r5 = r5-r2; (SUB R5 R2)

0111010100100000 r5 = r5\*r2 (MUL R5 R2) <- r5--를 구현하기 위함

while( r5 )

0011011000000000 int r6=r0+r1; (MOV3 R6 #0)

0100011000000000 (ADD R6 R0)

0100011000010000 (ADD R6 R1)

0001000000000000 r1=r0; (MOV1 R0 M0)

0000000100000000 (MOV0 R1 M0)

0001011000000000 r0=r6; (MOV1 R6 M0)

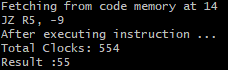
0000000000000000 (MOV0 R1 M0

0101010100100000 r5--; (SUB R5 R2) <- R5를 2배 해서 2 감소니까 --와 동일

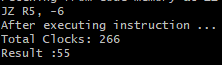
0110010111110111 } (JZ R5 -6)

아래는 실행 결과의 캡쳐로 Total Clocks와 Result를 비교하겠습니다.

<NO MOV4>



<USING MOV4>

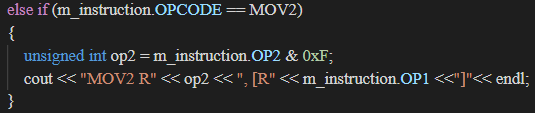


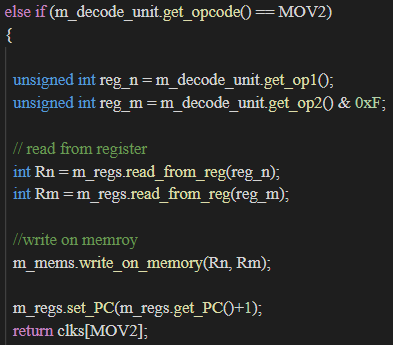
실행 결과값과 같이, MOV4를 사용하면 단순히 작성할 때 쓰이는 instruction 개수뿐만 아니라 while 구문 내에서 반복 실행되는 instruction이 3개나 줄어들기 때문에 비약적인 performance 개선이 이루어질 수 있었습니다. Result는 당연히 동일합니다.

아래는 MOV2와 MOV4 instruction 등 추가/변경된 code입니다.

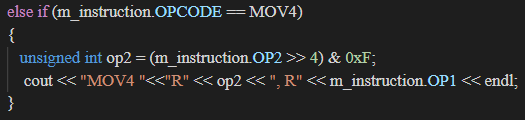
(ppt 자료에 있는 내용들은 첨부하지 않겠습니다.)

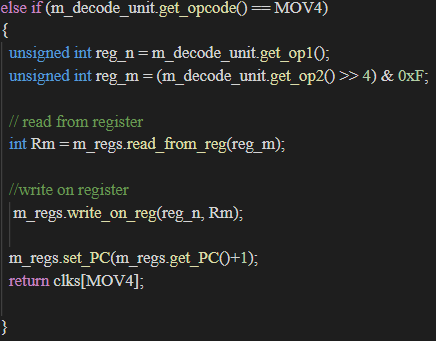
MOV2 – cdecode.cpp, cexecute.cpp



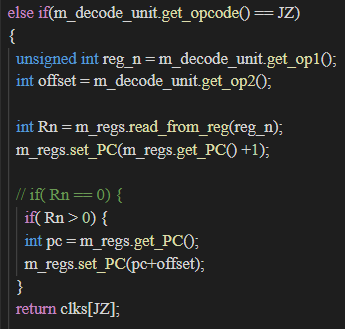


MOV4 – cdecode.cpp, cexecute.cpp

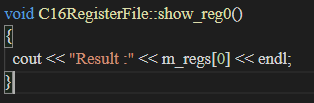




JZ – cexecute.cpp



show\_reg0(); 추가



main.cpp 에 추가한 show\_reg0(); 는 생략하겠습니다.