어셈블리프로그램설계및실습 보고서

과제 주차: 6주차

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2021202058

성 명: 송채영

제 출 일: 2022.10.15(금)

1. Problem Statement

Multiplication operation과 Second operand에 대해 알아본다. Multiplication operation과 Second operand를 사용해 구현해보고 성능의 차이를 비교한다. Operand의 순서에 따른 성능의 차이에 대해 알아본다.

2. Design

Second operand는 add연산과 shift 연산 만을 이용해 Multiplication 연산을 수행하는 것을 말한다. Multiplication operation은 MUL instruction(명령어)을 이용해 Multiplication 연산을 수행하는 것을 말한다. 이번 과제에서 사용하는 명령어에 대해 자세히 살펴보면, 먼저 MUL은 MUL{cond} Rd, Rm, Rs의 형식으로 사용하는데 Rm에 저장된 값과 Rs값을 곱하여 Rd에 저장한다. LSL은 LSL #number의 형식으로 사용하는데 Number만큼 왼쪽으로 bit stream을 shift 한다.

-problem 1

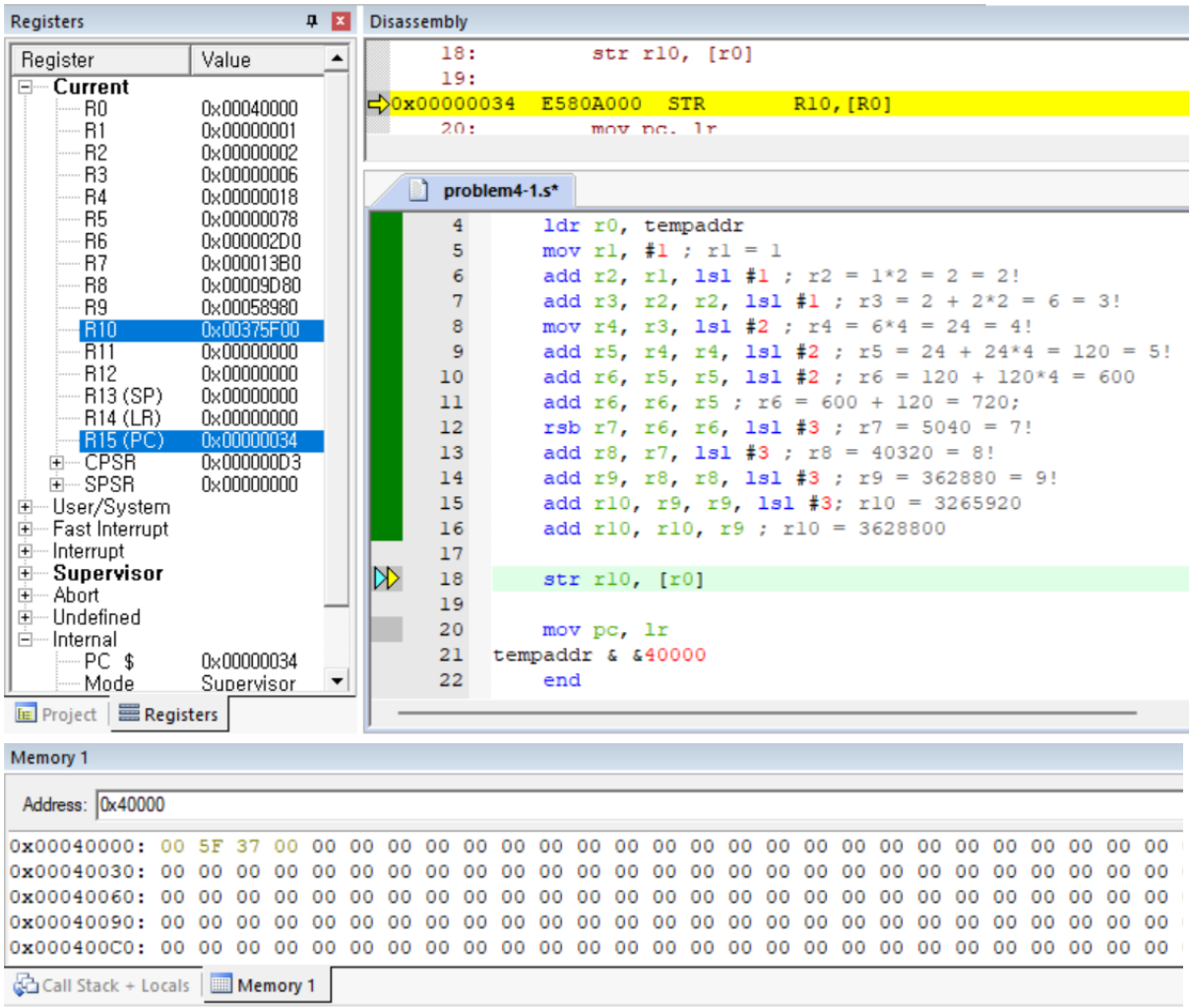
레지스터 r0에 40000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> 레지스터 r1에 1을 저장한다. -> r2에 r1*2의 값을 저장한다. -> r3에 r2*2 +2의 값을 저장한다. -> r4에 r3*2값을 저장한다. -> r5에 r4*4 + r4의 값을 저장한다. -> r6에 r5*4 + r5의 값을 저장한 다시 r6에 r6+r5한 값을 저장한다. -> r7에 r6*8 – r6의 값을 저장한다. -> r8에 r7*8의 값을 저장한다. -> r9에 r8*3 + r8의 값을 저장한다. -> r10에 r9*8+r9의 값을 저장한 후 다시 r10에 r10+r9의 값을 저장한다. -> r10의 메모리 값을 r0에 저장한다. -> 프로그램을 종료한다.

-problem 2

레지스터 r0에 4000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> r1에 1을 저장한다. -> r10에 2를 저장한다. -> MUL 명령어를 이용해 r2에 r1*r10의 값을 저장한 후 r10의 값을 1 증가한다. -> MUL 명령어를 이용해 r3에 r2*r10의 값을 저장한 후 r10의 값을 1 증가한다. -> r4에 r3*r10의 값을 저장한 후 r10의 값을 1 증가한다. -> r5에 r4*r10의 값을 저장한 후 r10의 값을 1 증가한다. -> r6에 r5*r10의 값을 저장한 후 r10의 값을 1 증가한다. -> r7에 r6*r10의 값을 저장한 후 r10의 값을 1증가한다. -> r8에 r7*r10의 값을 저장한 후 1을 증가한다. -> r9에 r8*r10의 값을 저장한 후 r10의 값을 1을 증가한다. -> r11에 r9*r10의 값을 저장한다. -> r11의 메모리 값을 r0에 저장한다. -> 프로그램을 종료한다.

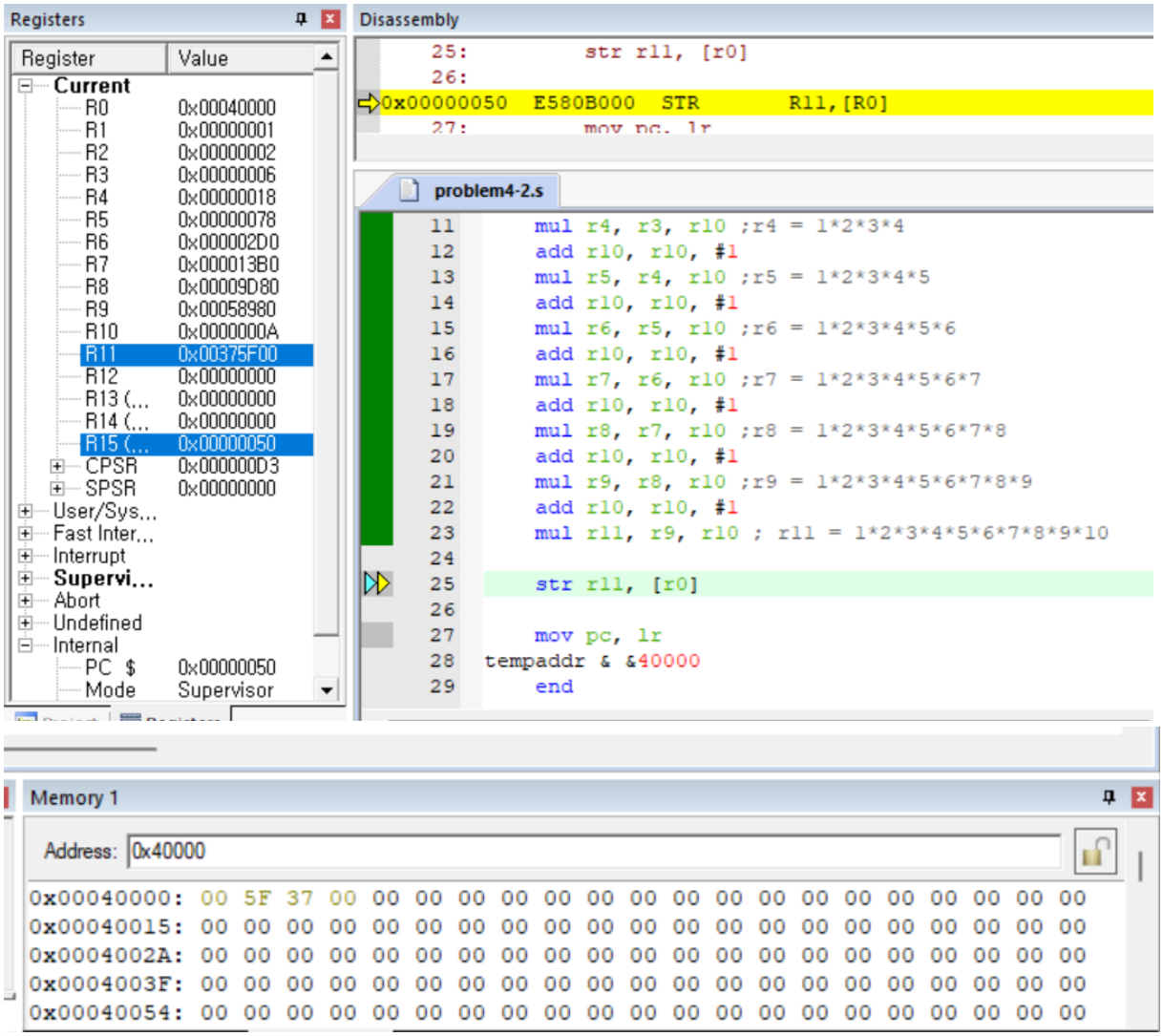
3. Conclusion

- problem 1



r1부터 r10까지 Second Operand을 통해, 즉 ADD와 LSL 명령어로 연산을 진행하는 것을 볼 수 있다. 또한 r10 즉 10!의 값인, 375f00이 메모리에 저장된 것을 볼 수 있다.

- problem 4-2



r1부터 r10까지 Multiplication Operation을 통해, 즉 MUL 명령어로 연산을 진행하는 것을 볼 수 있다. 또한 r11 즉 10!의 값인, 375f00이 메모리에 저장된 것을 볼 수 있다. 또한 Second Operand를 통해 multiplication 연산을 수행한 것과 결과가 같은 것을 알 수 있다.(차이는 consideration에서 설명함)

4. Consideration

Second Operand를 이용해 multiplication 연산을 수행하면서 값을 하나하나 계산해서 넣어야 하는 점이 복잡했다. R1에 1을 넣고 시작했지만, 10을 넣고 시작해서 10!, 9! 이런 식으로 내려오는 방법이 덜 복잡했을 것 같다. MUL 명령어를 사용하여 multiplication 연산을 수행하는 것은 10분도 안 걸려서 코드를 다 짰던 것 같다. 이번 과제를 수행하면서, Second Operand를 사용해 연산을 했을 경우, 시간이 오래 걸리고 계산하는 부분이 복잡했지만, MUL 명령어를 사용했을 때 보다 code size와 state가 작고 코드 길이도 짧았다. MUL 명령어를 사용해 연산을 했을 경우가 코드도 더 짧고 성능이 좋을 것이라 예상했는

데 그 반대였다.

5. Reference

이형근 교수님/어셈블리프로그램설계및실습/광운대학교(컴퓨터정보공학부)/2022