어셈블리프로그램설계및실습 보고서

과제 주차: 5주차

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2021202058

성 명: 송채영

제 출 일: 2022.09.30(금)

1. Problem Statement

Memory로부터 값을 받아 이를 이용해 원하는 연산을 수행한다. Assembly code를 작성할 때 performance를 생각하며 작성한다. Branch 명령어와 conditional execution의 차이를 알아본다. Memory를 할당해 data를 저장해본다.

2. Design

Branch와 conditional execution의 차이점과 성능 차이에 대해 말해보면 Branch는 실행 순서를 바꾸는 것을 말하며, conditional execution은 if문 같은 조건문으로 설명할 수 있다. 조건이 3개 이하일 때는 conditional execution을 사용하는 것이 효율적이며 조건이 4개일 때부터 branch를 쓰는 것이 성능 면에서 더 좋다.

-problem 1

레지스터 r0에 4000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> 레지스터 r1, r2에 각 문자열의 주소를 load한다. -> MOV 명령어를 통해 r3, r4에 각각 0x0A(10)을 0x0B(11)을 저장한다. -> r0, r1에 저장된 문자열을 한 글자 단위로 r5, r6에 불러와 두 글자가 다르다면 different로 11을 저장한다. -> 두 글자가 같다면 equal로 이동해 문자열의 끝인 0이 나올 때 까지 비교한 후 일치한다면 10을 저장한다. ->프로그램을 종료한다.

-problem 2

레지스터 r0에 4000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> 레지스터 r1에 10부터 1까지 나열된 주소를 불러온다. -> 레지스터 r2에 뒤에서부터 접근하도록 불러온다. -> 레지스터 r1에 저장된 메모리를 뒤에서부터 불러온 후 레지스터 r0에 저장한다. -> 배열 처음의 값까지 저장해준 후 프로그램을 종료한다.

-problem 3-1

레지스터 r0에 4000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> 레지스터 r1에 1을 저장한 후 LSL과 ADD명령어를 통해 11을 만들어 r2에 저장한다. -> LSL과 ADD 명령어를 통해 r3에 29를 저장한다. -> r4에 11부터 29까지의 값을 더한다. -> 11부터 2씩 더해서 증가시키면서 r2의 값이 29가 되면 값을 저장한다. -> 프로그램을 종료한다.

-problem 3-2

레지스터 r0에 4000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> 레지스터

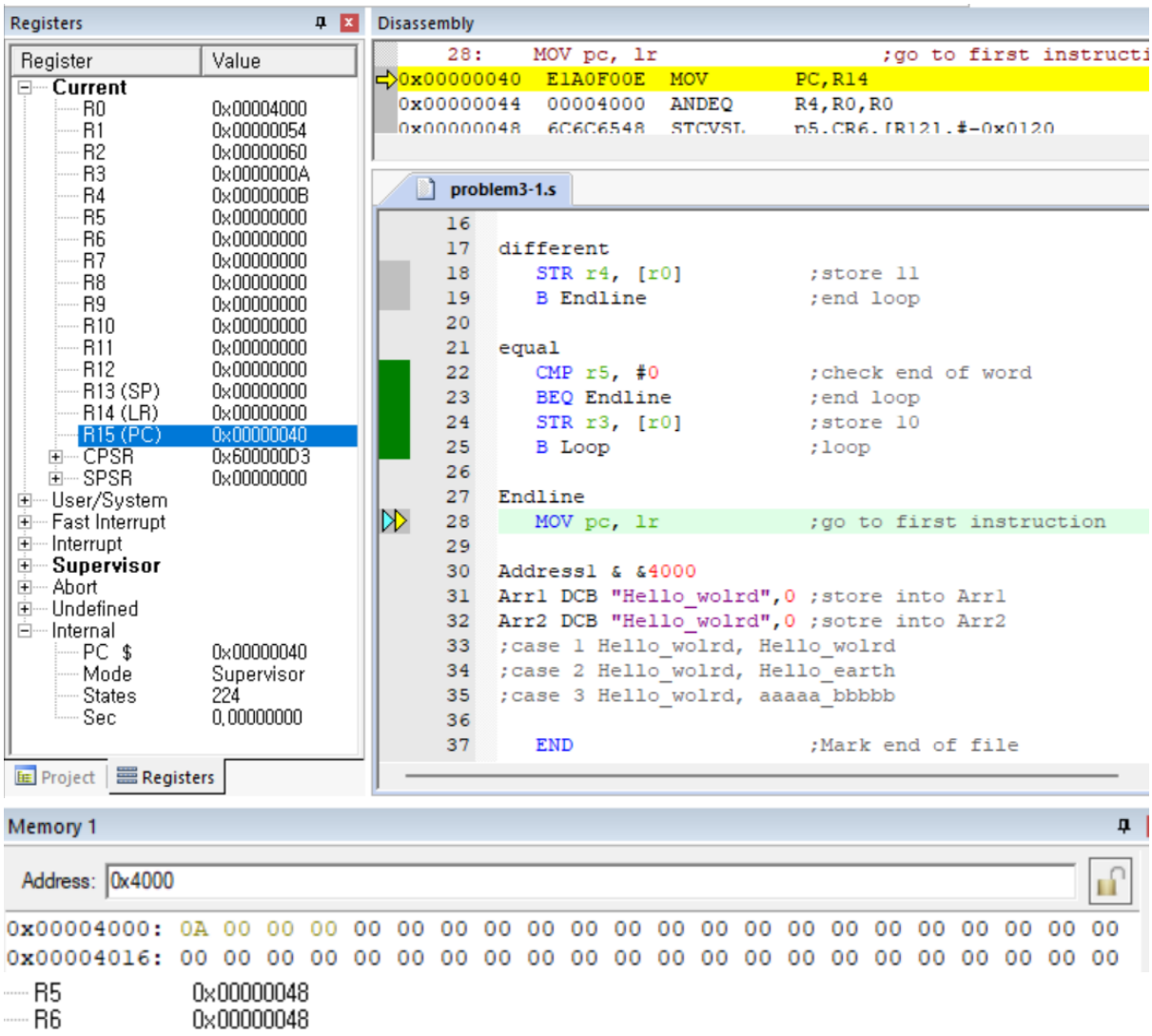
r1에 1을 저장한 후 LSL과 ADD명령어를 통해 n값 10을 만든다. -> r3에 (n+10)의 값 20을 만들어 저장한다. -> r4에 MUL명령어를 통해 계산한 값을 저장한다. -> r4에 저장된 값을 r0에 저장한다. -> 프로그램을 종료한다.

-problem 3-3

레지스터 r0에 4000번지 메모리를 불러와 메모리에 접근할 수 있도록 한다. -> 레지스터 r1에 11, 13, 15, 17, 19, 21, 23, 25, 27, 29의 수를 저장한 arr1을 불러온다. -> post index addressing을 통해 배열에 접근해 r2에 각 배열 요소들을 더해 나가는 과정을 반복한다. -> r2에 저장된 값을 r0에 저장한다. -> 프로그램을 종료한다.

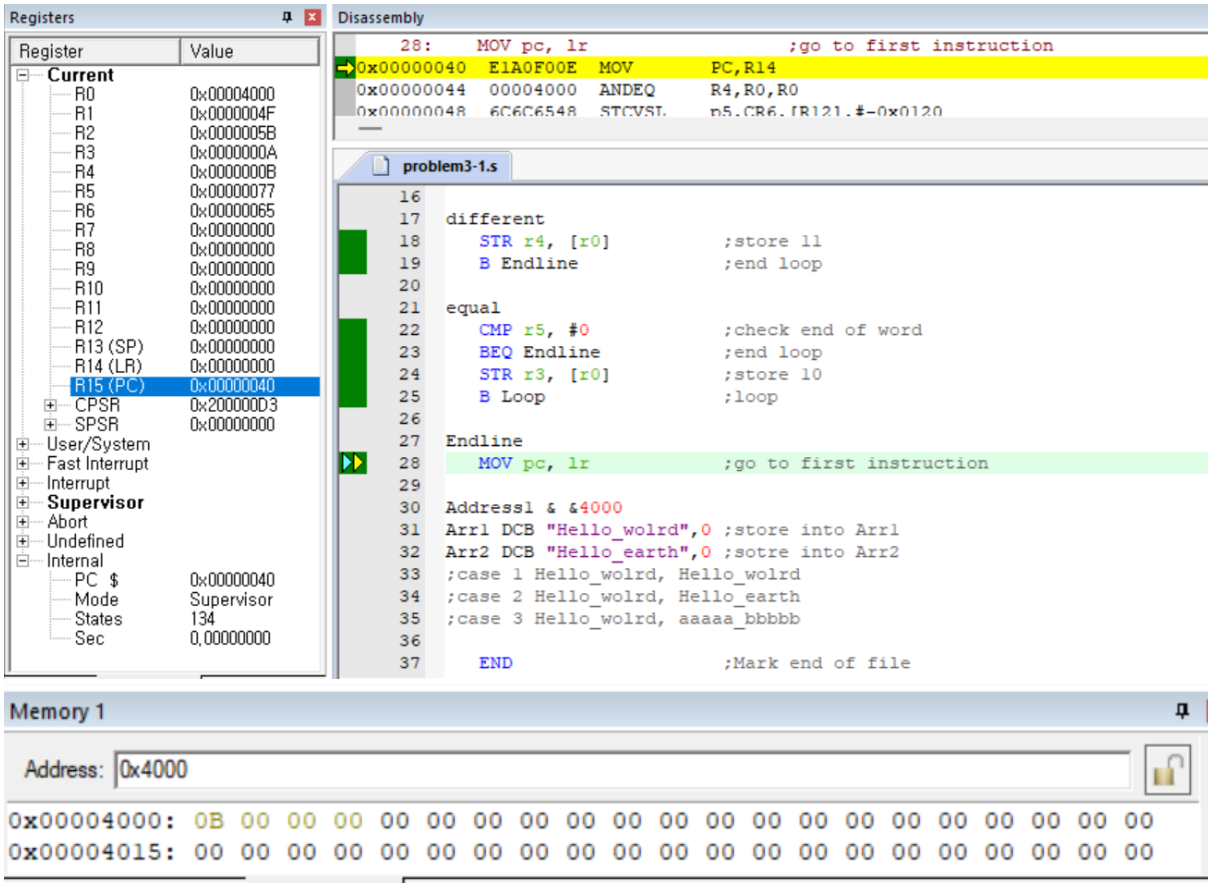
3. Conclusion

- problem 1

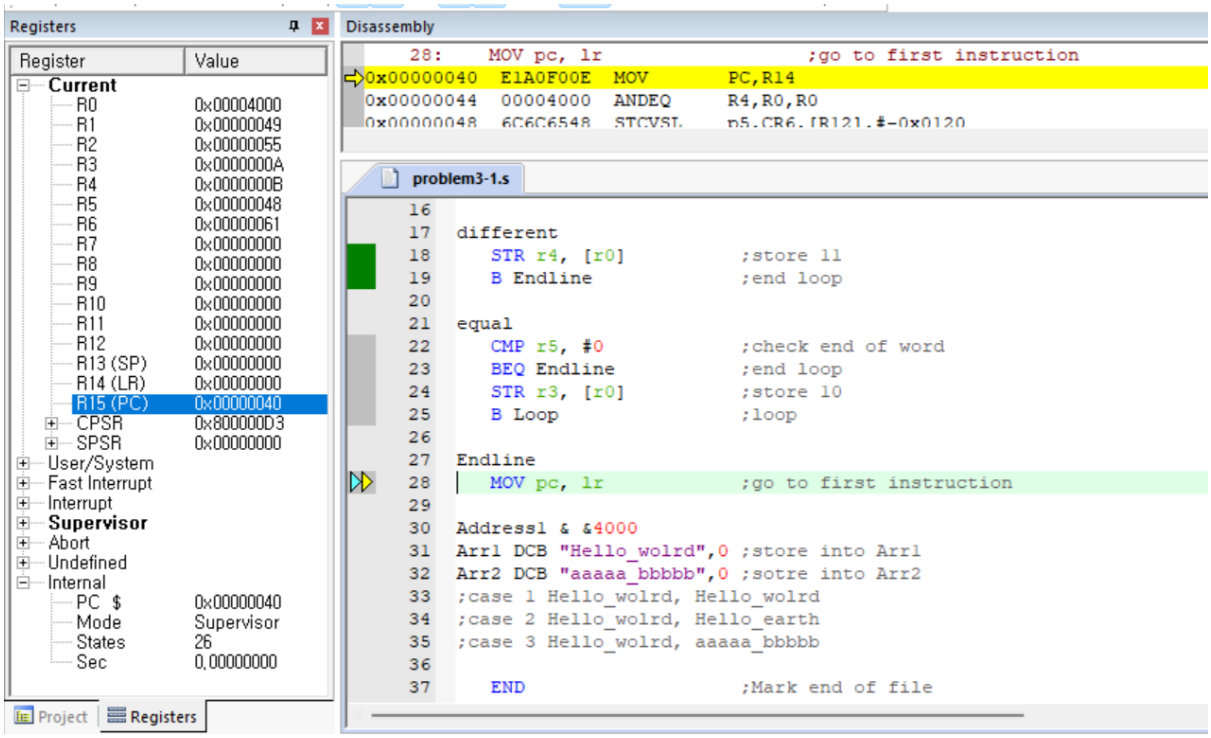


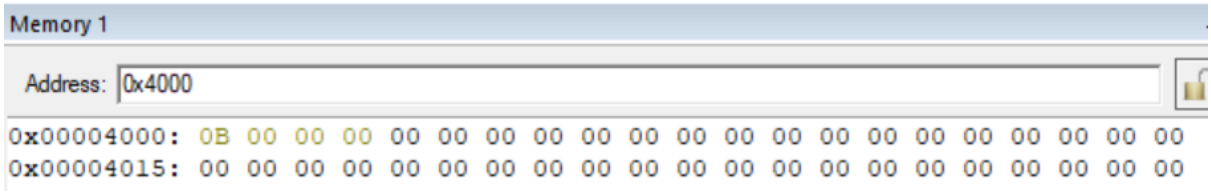
ARR1과 ARR2에 같은 문자열이 저장되어 있는 경우, 4000번지의 메모리에 A가 저장되어

있고, R5와 R6의 value 값이 일치하는 것을 확인할 수 있다. 또한 문자열의 끝을 의미하는 0을 만날 때 까지 비교한 후 10을 저장한 것을 알 수 있다.



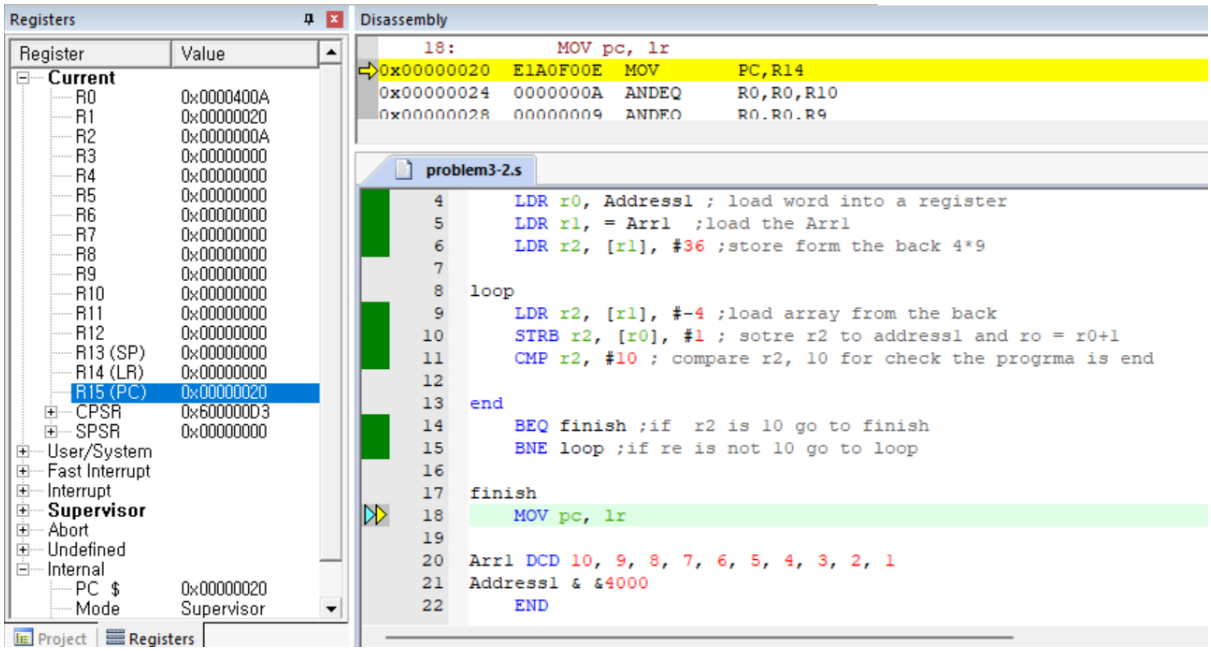
ARR1과 ARR2에 다른 문자열이 저장되어 있는 경우, 4000번지의 메모리에 B가 저장되어 있고, R5와 R6의 value 값이 다른 것을 확인할 수 있다.

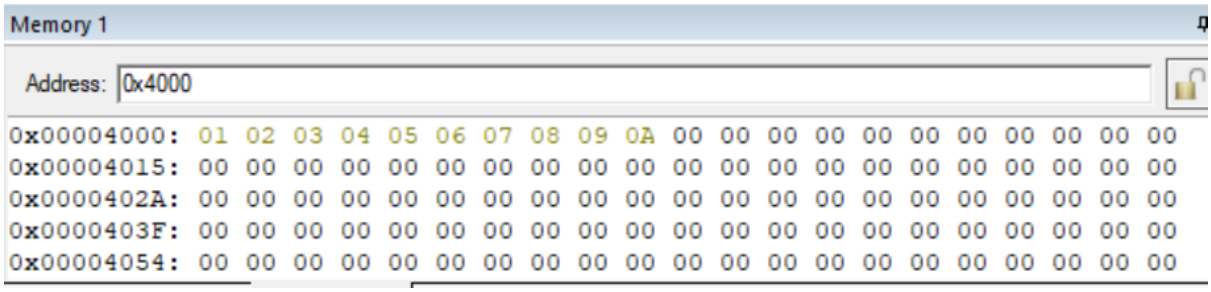




ARR1과 ARR2에 다른 문자열이 저장되어 있는 경우, 4000번지의 메모리에 B가 저장되어 있고, R5와 R6의 value 값이 다른 것을 확인할 수 있다.

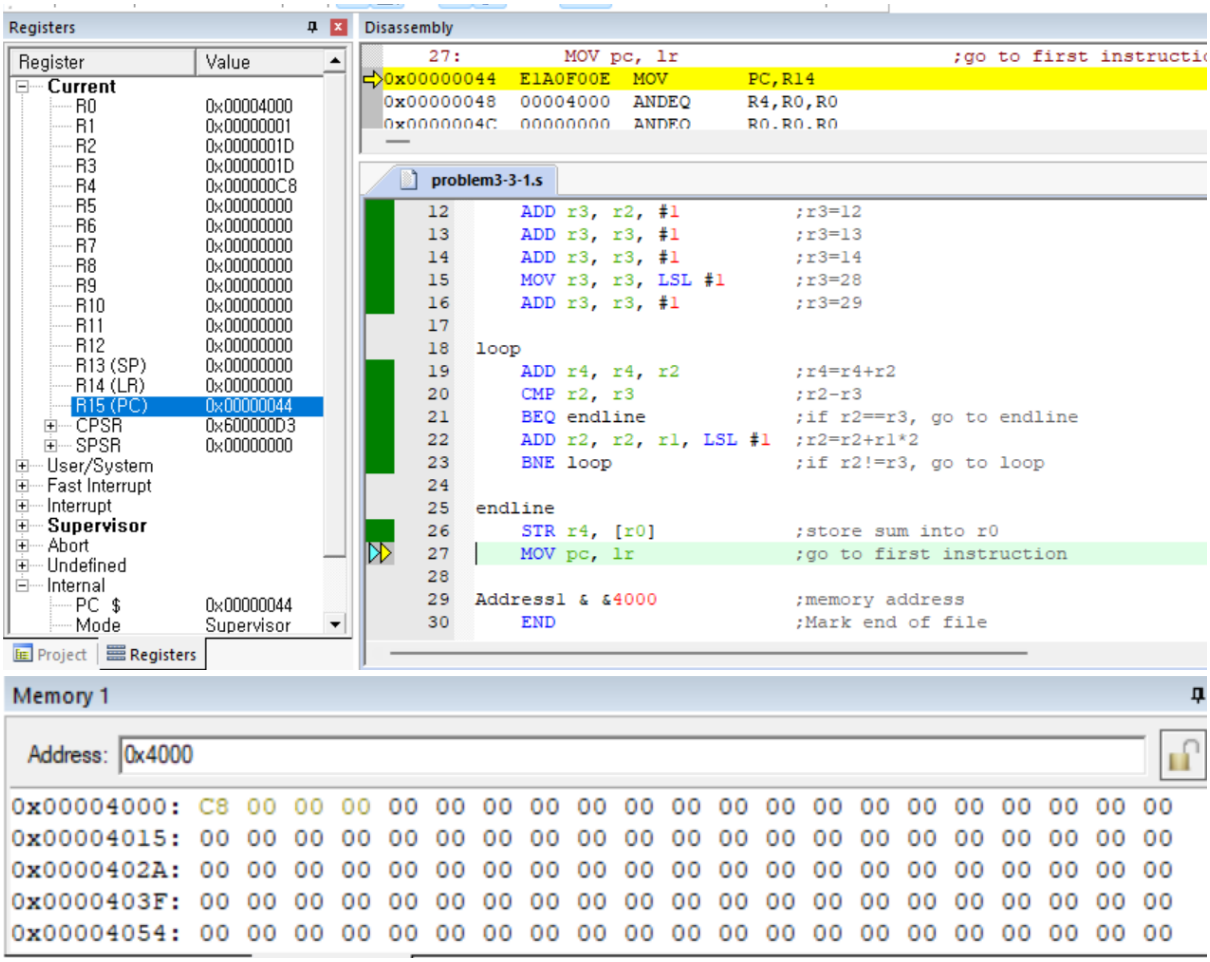
- problem 2





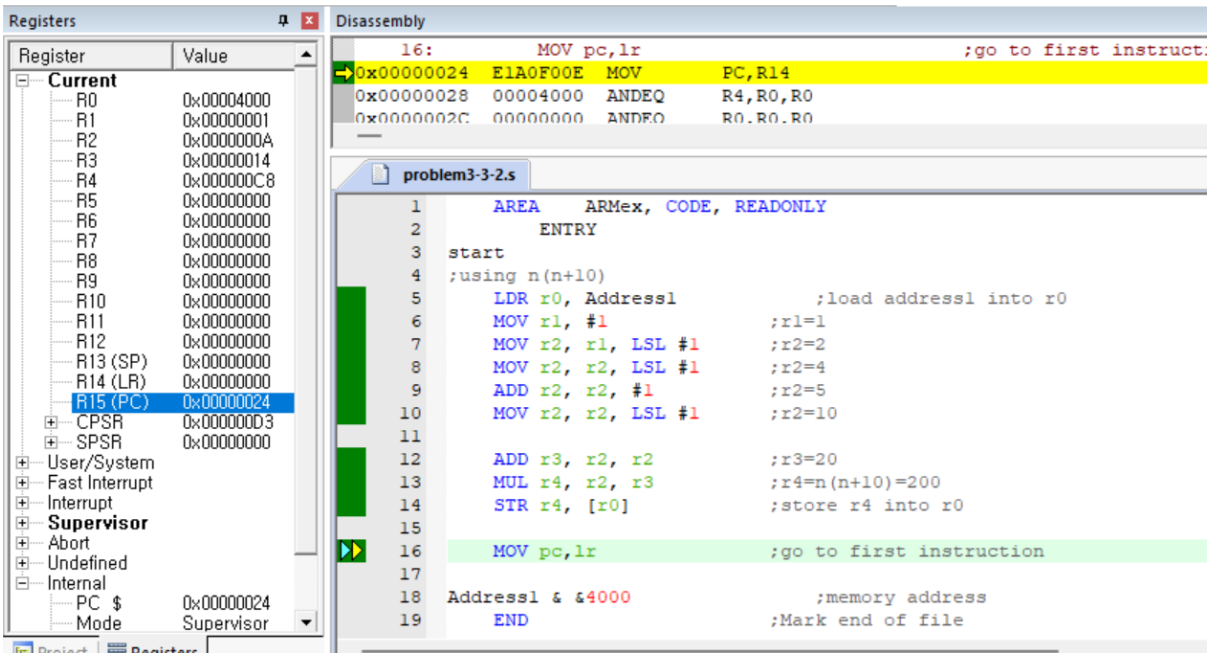
다음과 같이 Array의 마지막에 저장된 수부터 4000번지 메모리에 차례대로 정렬되어 저장 된 것을 볼 수 있다.

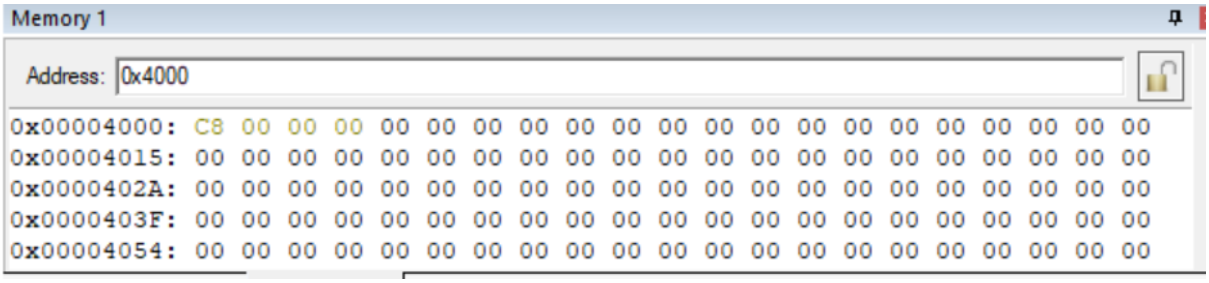
- problem 3-1



4000번지에 16진수로 C8, 즉 10진수로 200이 저장된 것을 볼 수 있다.

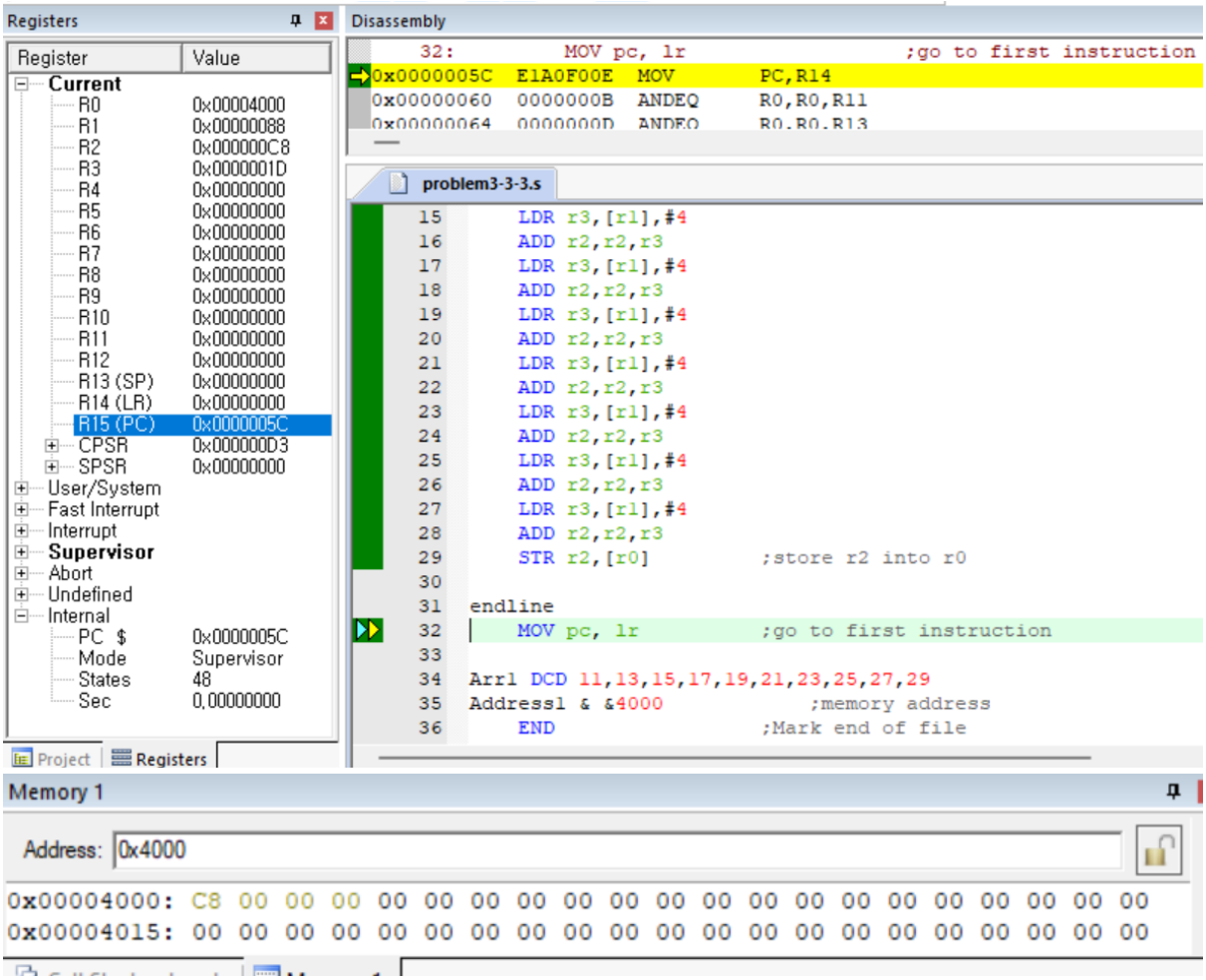
- problem3-2





4000번지에 16진수로 C8, 즉 10진수로 200이 저장된 것을 볼 수 있다.

- problem 3-3



4000번지에 16진수로 C8, 즉 10진수로 200이 저장된 것을 볼 수 있다.

4. Consideration

Problem 3-3번에서 unrolling이라는 단어를 처음 들어 보았는데, unrolling은 반복문이 돌아가는 과정을 줄여 성능을 높이는 방식임을 알았다. Loop와 n(n+10) 방식과 비교했을 때 code size가 커서 unrolling의 performance가 가장 크게 나왔다. 3가지 방법 중 일반화된 식, 즉 problem 3-2를 이용하는 것이 performance도 적게 나와 효율적일 것 같다. Problem 3을 통해 2번 Design에서 설명했듯이 조건적으로 수행하는 명령어가 3개 이하일

때에는 branch보다 conditional execution을 사용하는 것이 효율적이므로 conditional execution을 사용하는 것이 code size를 짧게 하며 성능도 좋게 만들 수 있다는 것을 알게 되었다. Branch 명령어, 문자열 저장 등을 처음 배워 처음엔 감을 못 잡았지만, 예시 코드를 수행해보며 이해하고 과제를 수행하니 훨씬 수월했던 것 같다.

5. Reference

이형근 교수님/어셈블리프로그램설계및실습/광운대학교(컴퓨터정보공학부)/2022

Branch와 conditional execution의 차이점/

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=jinlee0007&logNo=40105822384