어셈블리프로그램 설계및실습 보고서

5차 실습과제

학 과: 컴퓨터정보공학부

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2019202050

성 명: 이강현

제 출 일: 2022.10.31(월)

1. Problem Statement

레지스터와 메모리간에 블록단위로 데이터를 이동하는 것에 대한 이해를 기른다. 어셈블리에서 서브루틴의 사용법을 익힌다. 어셈블리에서 스택의 필요성과 스택이 어떻게 사용되고 실행되는지 관련 예제를 통해 이해하고 직접 활용해본다.

2. Design

<problem1>

첫번째 문제는 10!의 값을 스택과 재귀함수를 활용하여 구현하는 것이다.

따라서 곱할 값 r1과 결과값을 저장할 r2의 초깃값을 1로 세팅해두고 sp(stack pointer)에는 40000의 주소값을 저장해두고 시작한다.

그 이후 factorial이라는 함수로 서브루틴을 진행하기위해 이동하고 해당 함수내에서 r1을 1씩 증가시켜가며 r2에 곱해준다. 탈출조건은 r1이 10과 같아질때로 설정해두었기 때문에 함수가 함수내에서 탈출조건을 만족하기 전까지는 계속 함수를 호출한다. 결국 r1이 10과 같아졌을 때는 r2에 10!값이 저장되어 있게 되고 Endline이라는 함수로 이동하게 된다. 이 함수에서는 sp값에 해당하는 메모리에 r2의 값을 저장하고 프로그램을 종료한다.

<problem2>

두번째 문제는 레지스터의 값을 서로 바꾸는데 스택 혹은 블록복사 명령어를 활용하여 구현하는 것이다. 우선 초기세팅은 r0에 0, r1에 1,r2에 2……

r7에 7을 넣어둔 상태로 세팅해두었다. 그 후 r0부터 순차적으로 저장되어야하는 레지스터값을 메모리에 저장하였고 블록단위가 필요하다면 활용하였다. 그 후 메모리에 있는 값들을 블록단위로 한번에 LDMFA명령어를 통해 가져와 r0부터 r7까지의 레지스터에 값들을 넣어주고 프로그램을 종료한다.

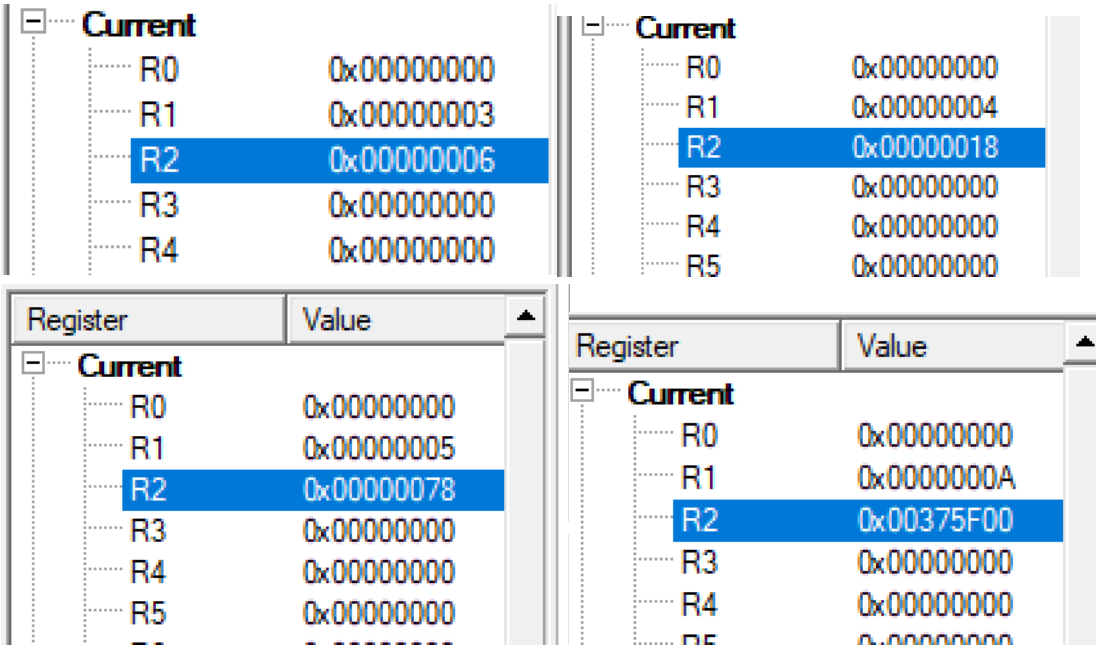
<problem3>

세번째 문제는 r0에는 10, r1에는 11…… r7에는 17이 저장되어 있는 초기세팅을 주고 이를 함수를 호출하여 값들을 변경하고 최종값을 메모리에 저장하는 흐름으로 진행된다. 우선 레지스터에 문제에서 제공된 값들을 저장한 후 doRegister함수를 BL명령어로 실행한다. doRegister함수는 각각의 register 값과 register의 인덱스 값을 더해 다시 레지스터에 저장하고 모든 레지스터의 값들을 더한 후 그 값을 return하는 함수이다. 따라서 ADD명령어를 통해 구현하였고 값을 구한 후

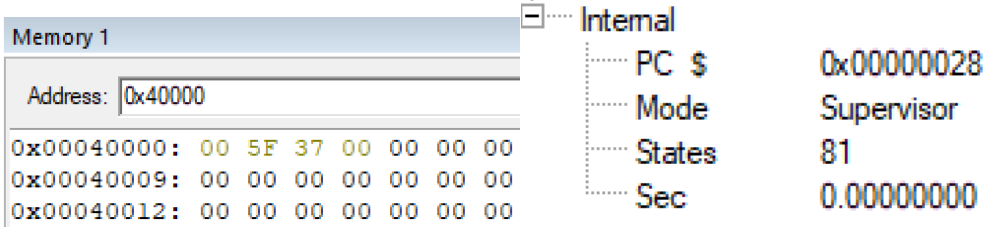
MOV pc,lr 명령을 통해 BL명령어를 사용한 바로 다음줄로 이동한다. 그 후 B 명령어를 통해 doGCD함수를 실행하는데 해당 함수는 doRegister에서 return된 값과 160의 최대공약수를 구하는 함수이다. 최대공약수를 구하는 법은 유클리드 호제법을 활용하였고 구한 최대공약수에 r4레지스터의 초기값을 더해주고 메모리에 저장해야하므로 메모리에 카피해둔 레지스터값들을 스택포인터를 활용한 블록단위명령어를 통해 레지스터에 저장한 후 r4를 최대공약수와 더해준다. 이를 sp값에 저장하고 프로그램을 종료한다.

3. Conclusion

<problem1>

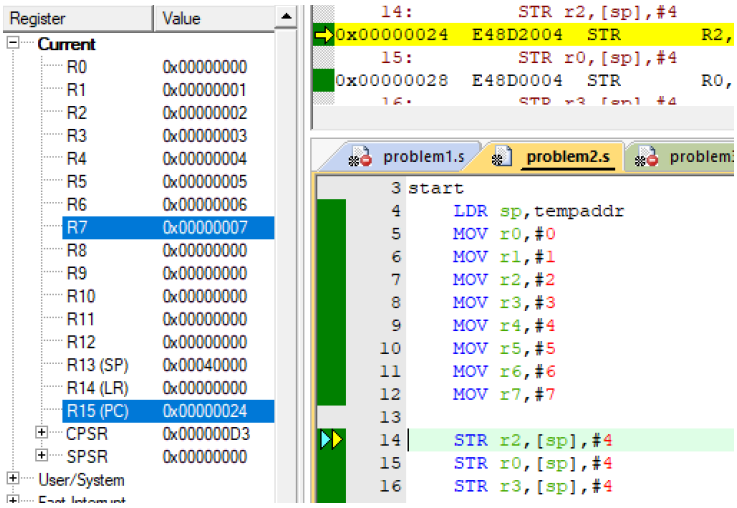


왼쪽 위부터 오른쪽순서로 r1이 1씩 증가해서 r2에 곱해지고 있고 최종적으로 r1이 10이 되었을 때 아래와 같이 메모리에 저장되는 것을 확인할 수 있다.

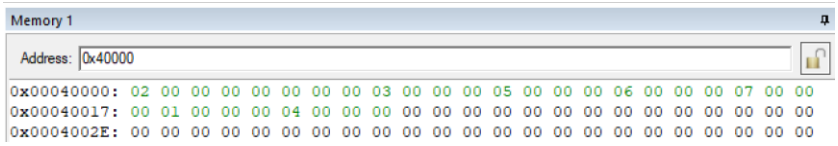


코드의 state 수는 81이다.

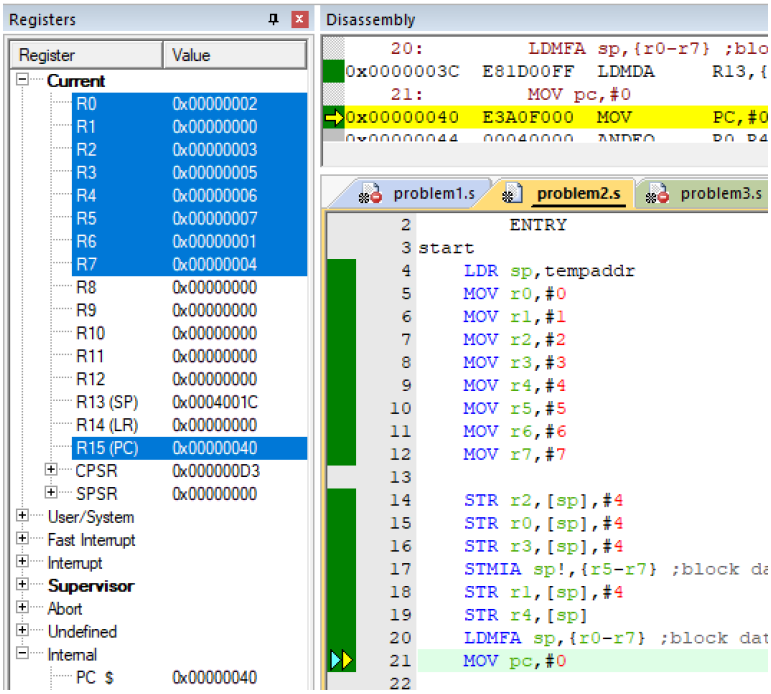
<problem2>



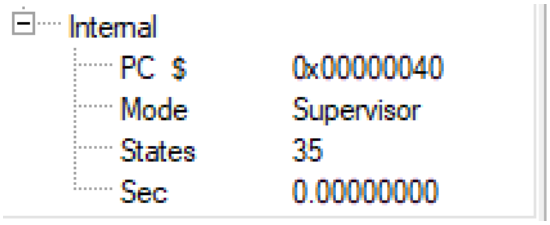
초기세팅



메모리에 문제에서 요구한 조건에 맞게 정렬하여 저장한 모습

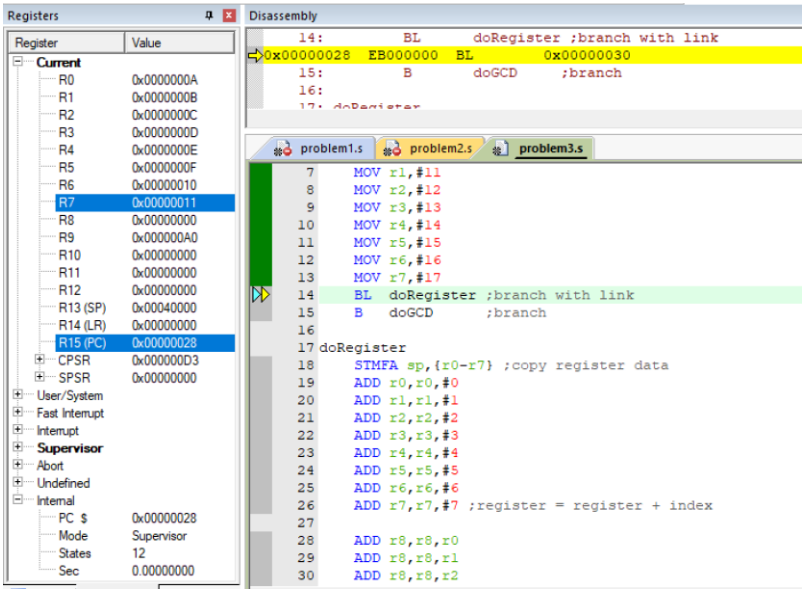


블록단위로 메모리에서 가져와 한번에 레지스터에 값을 저장한 모습

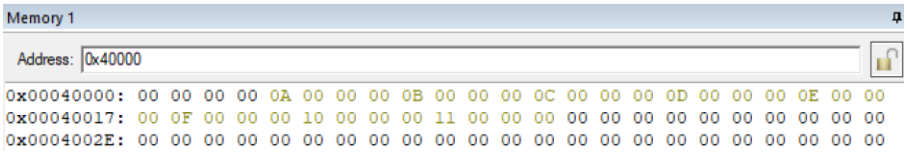


state수는 35이다.

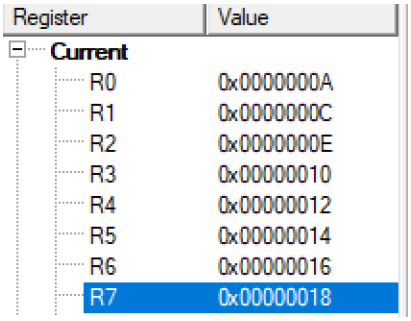
<problem3>



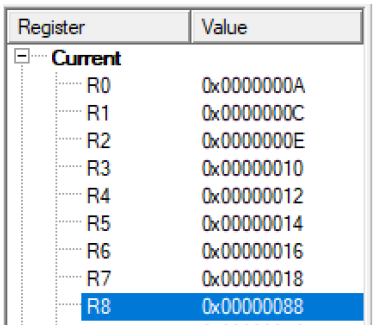
<초기세팅>



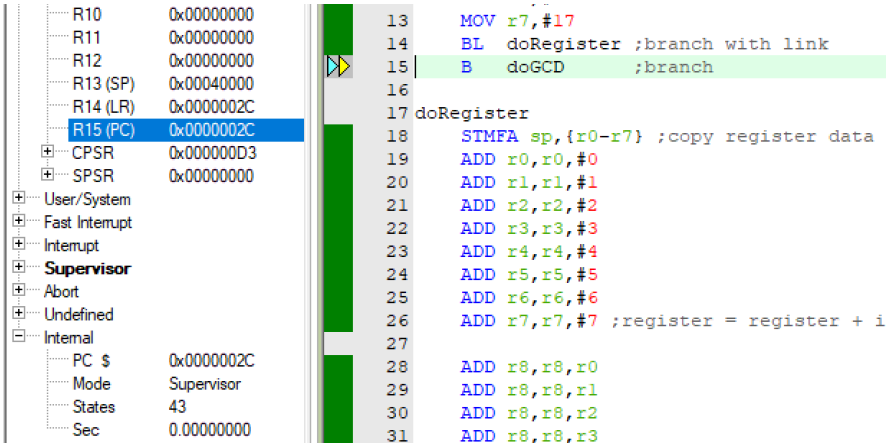
doRegister함수에서 레지스터값을 변경하기 전 미리 메모리에 저장해둠



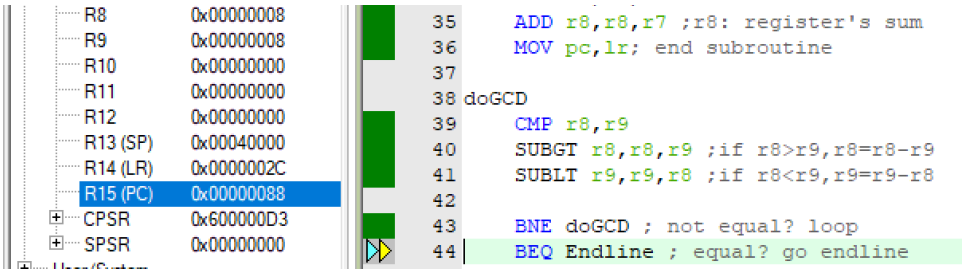
<인덱스 값을 각각 더해준 모습>



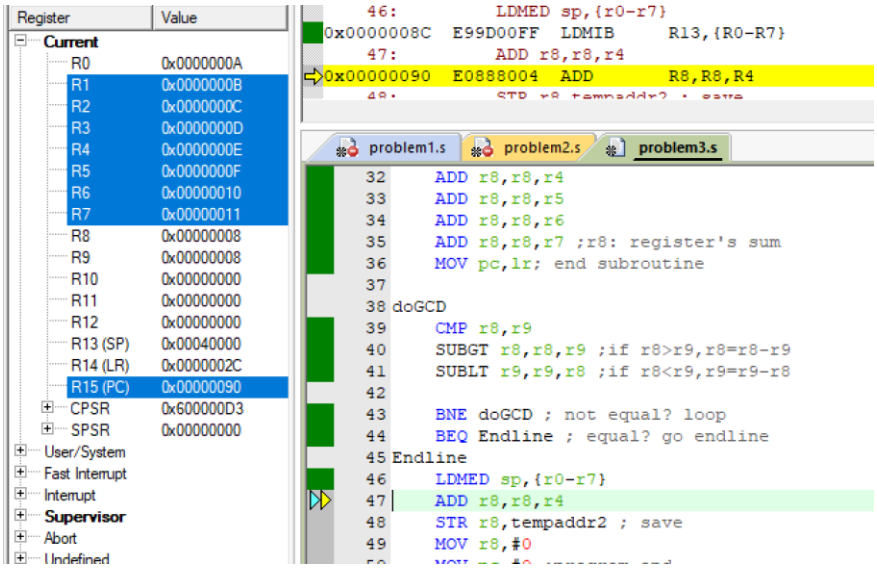
<이후 모든 레지스터의 값들을 더해 r8에 저장>



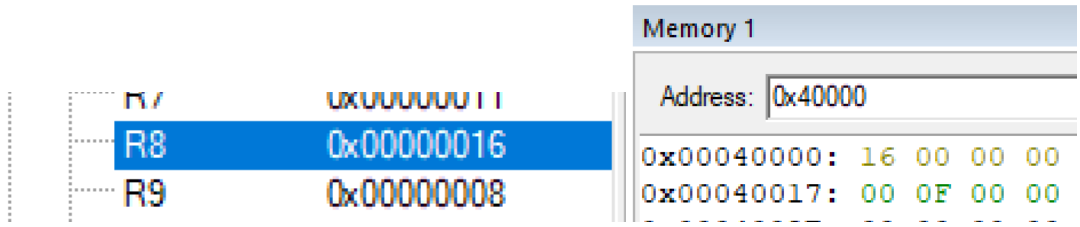
BL명령어로 저장해둔 곳에 doRegister함수를 마치고 돌아온 모습



R8,R9이 서로 값이 같아지자 doGCD함수를 탈출하는 모습

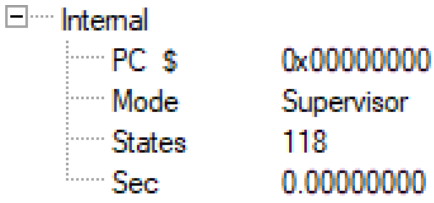


초기 r4값을 연산한 결과와 더해주기 위해 메모리에서 블록단위로 가져오는 모습



R8에 저장하고 STMFA를 통해 값을 저장하고 LDMED를 통해 값을 가져왔으므로 40000번지는 비어있고 sp또한 40000번지를 가리키고 있으므로 sp가 가리키는

곳에 0x16을 저장한다.



state수는 118이 나왔다.

4. Consideration

이번 실습에서는 스택과 블록단위 데이터 이동, 재귀함수의 사용등 다양한 개념을 학습하였다. FA,FD,EA,ED,IA,IB,DA,DB가 블록단위명령어 뒤에서 어떻게 작동하는지를 코드를 구현해보며 쉽게 이해할 수 있었다. 또한 서브루틴을 직접 구현해보면서 어셈블리 언어도 다른 high level 언어와 같이 함수를 구현할 수 있다는 것을 깨달았다. 또한 state수를 통해 블록단위 명령어,서브루틴등의 작업이 어느 정도의 state를 요구하는지를 알 수 있었다.

또한 BL명령어를 사용해보면서 일반적으로 사용해본 B명령어와는 다르게 다시 순차적으로 실행하던 부분으로 돌아갈 수 있다는 점이 앞으로 코드를 구현할 때 더 유용하게 사용할 수 있을 것 같다는 생각이 들었다.

5. Reference

이형근/어셈블리프로그램 설계 및 실습/광운대학교(컴퓨터정보공학부)/2022