어셈블리프로그램설계및실습 보고서

과제 주차: Block Data Tranfer & Stack

학 과: 컴퓨터정보공학부

담당교수: 이형근 교수님

실습분반: 화 6,7 목 5

학 번: 2019202021

성 명: 정 성 엽

제 출 일: 2022.10.31(월)

1. Problem Statement

LDM과 STM 명령어를 사용하여 레지스터와 메모리 사이의 블록 단위 데이터의 load와 store을 이해하고 스택 저장 방식과 명령어를 익혀본다. 또한 B 명령어를 통한 함수의 제작 그리고 사용법을 익히고 직접 설계하여 본다.

2. Design

A. LDM, STM

i. LDM Rn, { }

- 특정 메모리 주소 Rn에 저장되어 있는 값을 addressingmode에 따라 뒤의 operand 에 해당하는 위치에 load 한다

ii. STM Rn,{ }

- 특정 메모리 주소 Rn에 addressingmode에 따라 뒤의 operand에 해당하는 위치의 값을 저장한다. 만약 주소 값을 계속 변경하려고 한다면 앞의 Rn에 ! 를 추가하여 바뀐 주소가 저장된다.

iii. Addressingmode란

- IA – 빈공간 없이 해당 주소부터 차례로 접근한다. 오름차순
- IB – 한칸을 비우고 다음 주소부터 차례로 접근한다. 오름차순
- DA – 빈공간 없이 해당 주소부터 역순으로 접근한다. 내림차순
- DB – 한칸을 비우고 다음 주소부터 역순으로 접근한다. 내림차순

B. Stack

R13(sp)에 stack 자료구조를 어셈블리에서 구현할 수 있다. LDM와 STM 명령어를 통해 stack을 push 및 pop을 할 수 있다. 이를 이용하여 레지스터 내부에서 변화를 하지 않고 메모리단에 push했다가 pop을 하여 위치를 변경할 수도 있다.

C. B 명령어와 label

이전 B 명령어에서는 label의 주소로만 이동하는 것을 익혔다면 이번에는 새롭게 레지스터가 저장하고 있는 주소로 이동할 수 있다.

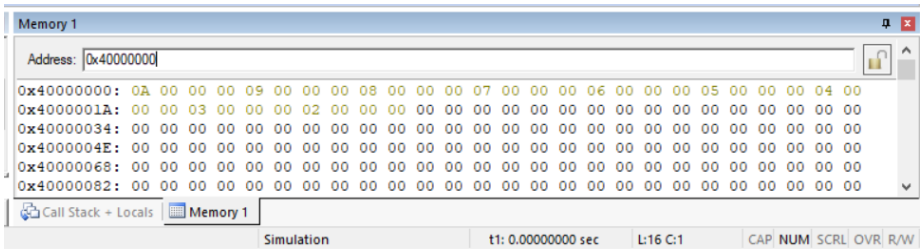
- B 명령어: PC레지스터를 해당 라벨의 주소로 이동시킨다.
- BL 명령어: B와 같지만 LR 레지스터에 이동 전 주소를 저장한다
- BX 명령어: 레지스터가 저장하고 있는 주소로 이동시켜준다.
- BLX 명령어 – BX 명령어와 같되 LR 레지스터에 이동 전 주소를 저장한다.

D. 두 수의 최대공약수

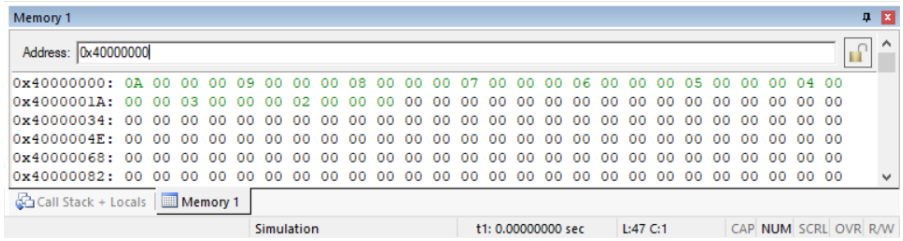
- 두 수의 최대 공약수를 구하는 알고리즘은 두개의 수 A와 B 중 큰 수에서 작은 수를 빼고 그 수를 A, B 중 큰 수에 저장하고 이를 반복하여 A와 B가 같아질 때 이 수는 이 A와 B 두수의 최대 공약수이다.

3. Conclusion

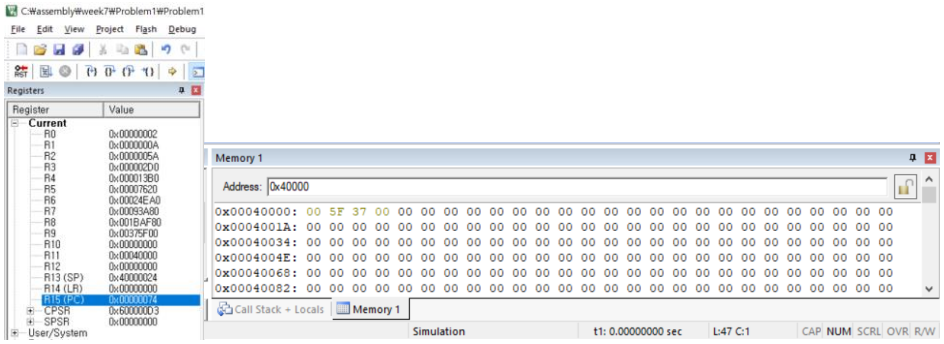
i. Problem1



재귀함수를 통해 10부터 2까지의 값을 메모리에 push해 두었다. 1을 저장 안한 이유는 곱했을 때 결과가 같기 때문이다.

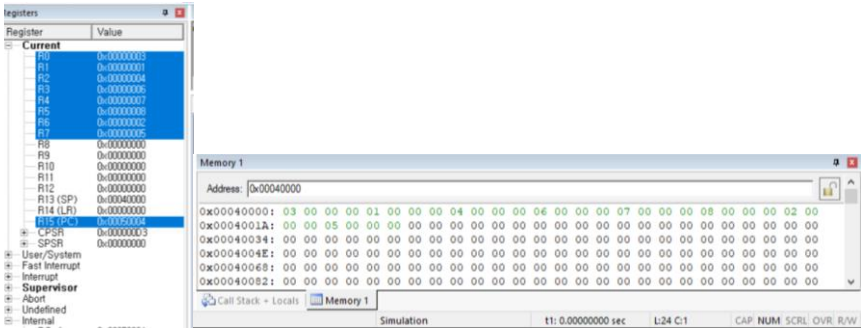


push된 값이 모두 pop 되었고



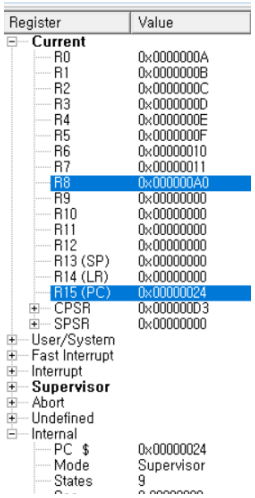
하나하나 pop 하며 곱한 값은 register에 저장되었으며, 40000번대 메모리에 10!의 값이 저장되어있다.

ii. Problem2

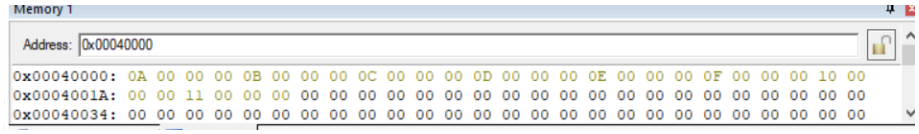


R0 = 1, R1= 2, R2= 3, R3=4, R4 = 5, R5 = 6, R6 =7, R7 = 8 이었으나 push와 pop 이후에 R0값이 R1으로 R1값이 R6로 R2값이 R0로 R3값이 R2로 R4값이 R7으로 R5값이 R3로 R6값이 R4로 R7값이 R5로 이동함을 확인할 수 있다.

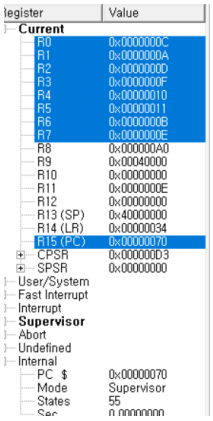
iii. Problem3



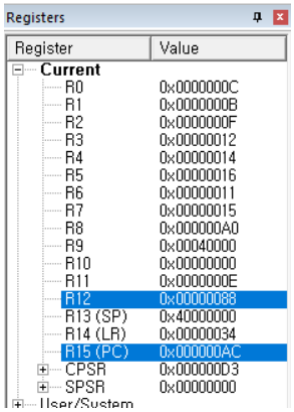
R0~R7까지 10~17의 값이 입력되었다.



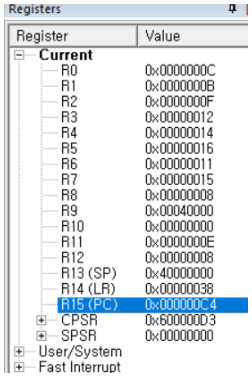
임의의 위치에 이 값들을 모두 push 해 놓았으며 이는 초기 값이다.



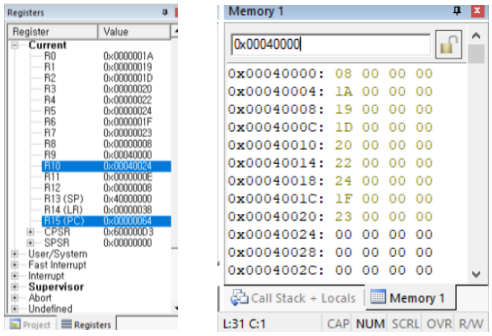
doRegister label에서 problem 2와 같이 자리를 옮기고



각 자리에 Register index만큼 더하고 R12에 모두 더한 값을 저장하였다.



doGCD를 진행했을 때 160을 저장해둔 R8과 모든 레지스터값을 더한 R12가 같을 때 반복을 탈출했다.



GCD의 값을 40000번대 메모리에 저장하고 레지스터값을 이전 R4의 값을 더하고 그 값을 이어서 40000번대 메모리에 push 하였다.

4. Consideration

이번 과제는 이전 C언어 또는 C++에서 했던 재귀함수를 어셈블리로 구현하는 것으로 그리 큰 어려움은 없었지만 새로운 명령어 STM/ LDM을 이용하여 push와 pop을 진행하는 것은 놀라울 정도로 간편하고 한번에 처리할 수 있다는 점이 매력이 있었다.

이때 SP 레지스터를 조정하여 STM/LDM 명령어를 사용하였으며 메모리 주소를 변경시켜 push나 pop 되는 위치를 다르게 하였다.

B 명령어 또한 이전에 사용해 보았기에 큰 어려움은 없었으나 새롭게 BX 명령어를 사용

해 보면서 lr에 저장되었던 위치로 다시 이동함을 실습해보았다.

이번 과제에서 가장 힘들고 어려웠던 것은 3번 문제의 해석이었다. doRegister을 하기 전의 R4를 주소에 더하라는 것인지 레지스터에 더하라는 것인지 말뜻이 조금 애매모호하였지만 아무래도 레지스터 각 값에 더하는 것을 요구하는 것이라 판단하여 진행하였다.

5. Reference

이형근 교수님/어셈블리프로그램설계및실습/광운대학교 컴퓨터정보공학부/ 2022