1. 배경지식

(1) LDM/STM Addressing mode [i]

LDM, STM 명령어는 각각 Load multiple registers와 Store multiple registers를 의미하고, 여러 개의 레지스터에 메모리의 정보를 불러오거나 반대로 메모리에 여러 개의 정보를 저장할 수 있다. 이 명령어를 이용할 때 Addressing mode를 표기하여 메모리가 저장되는 방식을 지정할 수 있다. Addressing mode는 메모리에 정보를 저장하는 방향에 따라 2가지로 나눌 수 있고, 각각을 다시 메모리를 저장하는 것과 주소값을 옮기는 과정의 순서에 따라 2가지로 나누어 총 4가지의 모드로 나누어 사용할 수 있다. 먼저 인자로 전달된 Rn을 기준으로 increment 방향으로 정보를 저장하는 것은 'I'를 이용한다. 반대로 decrement 방향으로 정보를 저장하는 모드는 'D'를 축약어로 갖는다. 정보를 저장한 이후에 주소값을 옮기는 방법은 after의 약자인 'A'를 이용하고, 반대로 주소값을 옮긴 이후 정보를 저장하는 방식은 'B'를 이용하여 before 방식임을 명시한다.

또한 이와 같은 분류는 stack mode에서도 유효하다. 이 경우, 정보를 저장하는 방향이 increment에 해당하는 모드는 Ascending mode이다. 반대로 decrement 방향은 그대로 descending mode라고 한다. stack mode의 경우 addressing mode와는 다르게 정보의 저장이 종료된 후, Rn의 주소값이 가르키게되는 정보가 유효한 정보일 경우 Full을, 유효하지 않은 정보인 경우 Empty를 이용한다. 위에서 살펴본 addressing mode와 비교하면, 주소값을 먼저 옮기고 이후에 정보를 저장하면(Before), 옮겨진 주소값이 가르키는 곳의 정보는 새로 저장된 정보이므로, 유효한 정보를 가르키는 Full이 된다. 반대로 값을 저장한 뒤 주소값을 옮기게 되면(After), 옮겨진 주소값이 가르키는 곳의 정보는 유효한 정보가 아닌 Empty이다.

(2) 서브루틴[ii]

서브루틴은 프로그램에서 필요에따라 어떤 기능을 독립적으로 정의하여 사용하는 것이다. 함수와 서브루틴이 구분이 되지 않는 경우도 있다. 주로 프로그램 내에서 여러 번 사용되는 내용이나 미리 프로그램 되어있는 부분을 이용하는 경우에 서브루틴을 이용하여 구현한다.

(3) 유클리드 호제법[iii]

유클리드 호제법은 두 정수의 최대공약수(Greatest Common Divisor)를 구할 수 있는 알고리즘이다. 고대 그리스의 수학자 유클리드에 의해 고안되었다. 유클리드 알고리즘은 어떤 수를 다른 수로 나누어 얻은 나머지를 이용한다. 어떤 수를 나누는데 사용한 수를 다시 나머지로 나누는 과정을 결국 나머지가 0이 될 때까지 반복하고,

마지막으로 나누는데 사용한 수가 구하려던 두 수의 최대공약수가 된다.

2. 코드내용

(1) Problem1

r0부터 r7에 0부터 7까지의 수를 저장한다. STMFA명령어를 이용해서 변경할 값을 순서대로 stack에 삽입한다. 이후 LDMFA명령어를 이용하여 stack의 값을 r0부터 r7까지 순서대로 저장한다.

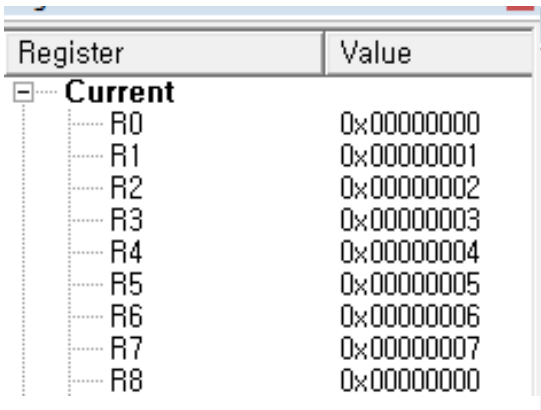
(2) problem2

r0부터 r7에 10부터 17까지의 수를 저장한다. STMIA 명령어를 이용해서 모든 값을 메모리에 저장한다. doRegister 함수에서 r0부터 r7에 ADD명령어를 이용해 index를 더한다. 또한 r0부터 r7의 값을 모두 더하여 r9에 저장하는 계산 역시 doRegister함수에서 수행한다. 이후 doGCD함수에서 r9에 저장된 값과 160과의 최대공약수를 구한다. doGCD함수는 최대공약수를 구하는 알고리즘인 유클리드 호제법을 이용하였으며, 나머지를 구하는 과정을 loop label과 B를 이용하여 구현하였다. 다음으로 doADD함수에서 메모리에 저장해 두었던 10에서 17까지의 수를 하나씩 꺼내어 r0부터 r7에 각각 더해준다. 이 계산이 종료되면 문제에서 제시한 값을 모두 구한 것이므로, 다시 r8에 TEMPADDR1 주소를 지정한 뒤, STMIA명령어를 이용하여 GCD값과 r0부터 r7의 값을 메모리에 저장한다.

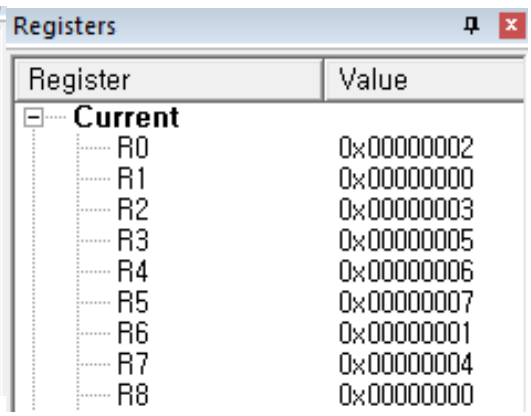
3. 결과

(1) problem1

LDM 실행 전

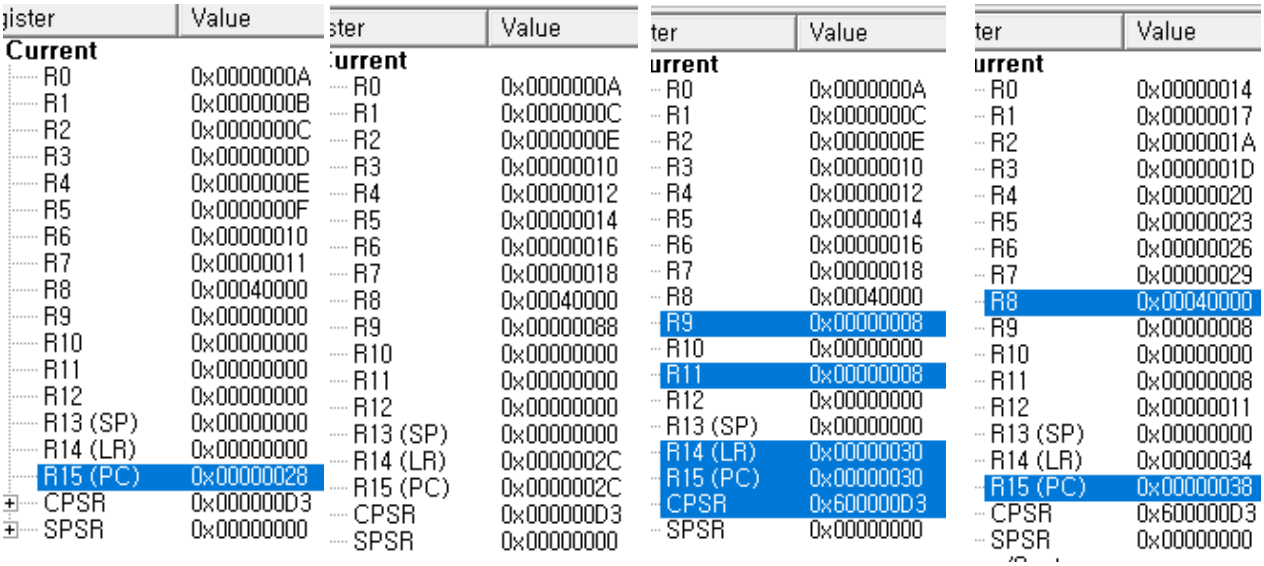


실행 후

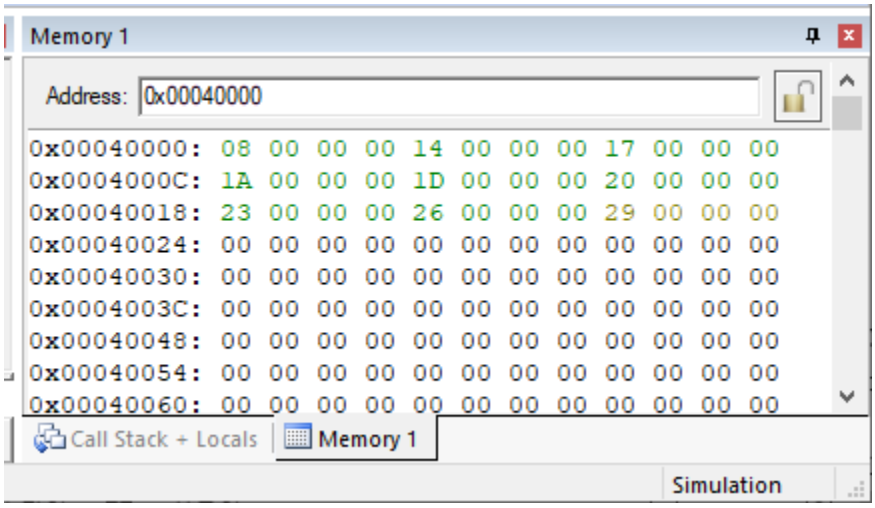


(2) problem2

값 저장 후 doRegister 결과 doGCD결과 doADD 결과



프로그램 실행 완료



4. 고찰

(1) problem1

레지스터에 여러 개의 데이터를 한 번에 저장하고 가져오는 LDM, STM명령어를 응용하는 문제였다. LDR/STM 명령어의 뒤에 붙는 addressing mode에 대한 이해가 어려웠다. 이 부분은 레포트 1번 항목의 배경지식 작성을 위해서, 인터넷을 통해 자료

를 검색하여 얻은 정보와 몇 가지 testcase를 직접 작성하여 비교하는 것을 통해 이해하는 시간을 가졌다.

(2) problem2

직접 함수를 정의하여 서브루틴을 실행하는 연습을 할 수 있는 문제였다. BL명령어를 이용하여 프로그램의 위치를 기록하는 것으로 구현하였다. 이 문제의 경우 r0부터 r7까지 반복적인 작업을 하는 부분이 많았는데 이것을 하나씩 나열하는 것이 아닌 반복을 통해 해결하려고 했으나 레지스터의 번호를 적절히 바꾸는 것을 실패하여 구현하지 못한 점이 아쉽다.

---

[i] ARM Developer, Implementing stacks with LDM and STM https://developer.arm.com/documentation/dui0068/b/writing-arm-and-thumb-assembly-language/load-and-store-multiple-register-instructions/implementing-stacks-with-ldm-and-stm?lang=en

Embedded Recipes, Stack의 정체와 자세히 보기, http://recipes.egloos.com/5059742

한컴MDS, LDM and STM, http://trace32.com/wiki/index.php/LDM_and_STM

[ii] 정보통신용어사전, 서브루틴, http://word.tta.or.kr/dictionary/dictionaryView.do?subject=%EC%84%9C%EB%B8%8C%EB%A3%A8%ED%8B%B4

[iii] 정보통신기술용어해설, 유클리드알고리즘, http://www.ktword.co.kr/word/abbr_view.php?m_temp1=285&m_search=%EC%9C%A0%ED%81%B4%EB%A6%AC%EB%93%9C%ED%98%B8%EC%A0%9C%EB%B2%95

Britannica, Euclidean algorithm, https://www.britannica.com/science/Euclidean-algorithm