어셈블리프로그램 설계 및 실습
과제1 보고서

담당교수	이형근 교수님
실습분반	화요일 (2분반)
소속	컴퓨터정보공학부
학번·성명	2021202085 전한아솔
제출일	2024년 11월 12일

1. Problem Statement

과제의 목적은 크게 5가지로,

1. ARM 프로세서의 기본 명령어를 이해하고 활용해본다.
2. 스택과 메모리 관리 능력을 향상한다.
3. 재귀 함수를 이해해보고 구현한다.
4. 레지스터와 메모리를 조작하며 좀 더 능숙하게 다룬다.
5. 프로그램의 성능을 분석하고 최적화 기법을 학습한다.

이렇게 나눌 수 있다. 따라서 이번 과제를 통해 ARM 프로세서의 기본 명령어를 이용해 스택, 메모리에 접근하고 관리하는 능력을 숙달한다. 또 재귀 함수를 ARM 프로세서를 이용하여 구현해보면서 더 잘 이해하도록 하는데 목적을 둔다.

2. Method

Problem 1

Second operand에서 Shift 연산을 이용하여 factorial을 계산하는 문제이다. LSL을 하면 왼쪽으로 한 비트를 shift 하므로 곱하기 2, LSR을 하면 오른쪽으로 한 비트를 shift 하므로 나누기 2라고 볼 수 있다. 따라서 이를 이용하여 3은 1+2, 5= 1+4, 6=2+4, 7=8-1, 9=1+8, 10=2+8로 생각하여 2의 제곱꼴이 아닌 숫자를 2의 제곱수로 구성하여 곱해주면 factorial을 shift 연산을 사용하여 구현할 수 있다.

Problem 2

스택과 재귀 함수를 이용하여 factorial을 계산하는 문제이다. 현재 레지스터의 상태를 push하고 다른 서브루틴을 호출한 후, 복귀하여 pop하면 레지스터의 상태를 복원할 수 있으므로 10!을 계산하기 위해서는 레지스터에 10을 저장, push, 레지스터의 값 감소, 레지스터에 9를 저장, push, 레지스터의 값 감소...의 과정을 1까지 반복한다. 이제 레지스터의 값이 0이 되었다면 0!은 1이므로 branch with link명령어를 이용하여 다른 서브루틴으로 넘어가서 10!이 담길 레지스터를 1(0!을 의미)로 저장한다. 이후 link register를 이용하여 원래의 서브루틴으로 돌아가서 스택의 저장된 값을 pop하여 이전 레지스터의 값을 복원한다. 스택 구조이므로 LIFO의 특성을 가진다. 따라서 10-9-8....2-1의 값을 넣었으므로 pop은 그의 역순인 1-2-....9-10의 값으로 복원 될 것이고 해당 값을 아까 0!을 저장한 레지스터 곱하는 방식으로 factorial을 구현하면 된다.

Problem4

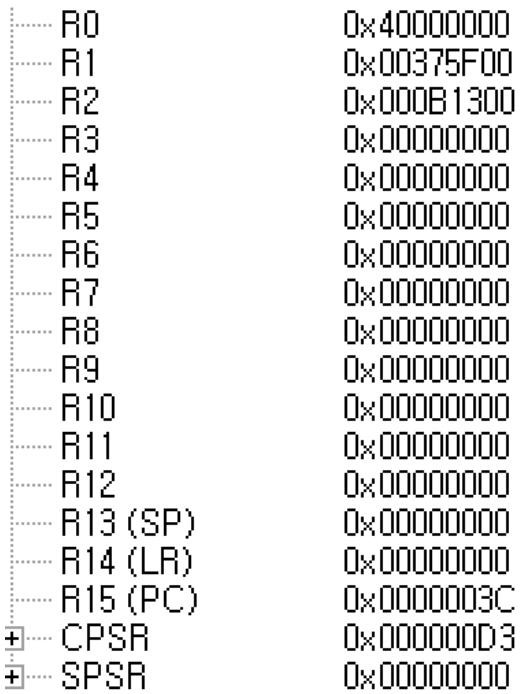
R0~R7의 레지스터에 1~8의 값을 넣고 Stack 또는 Block data transfer 이용하여 레저스터의 값을 교환하는 문제이다.

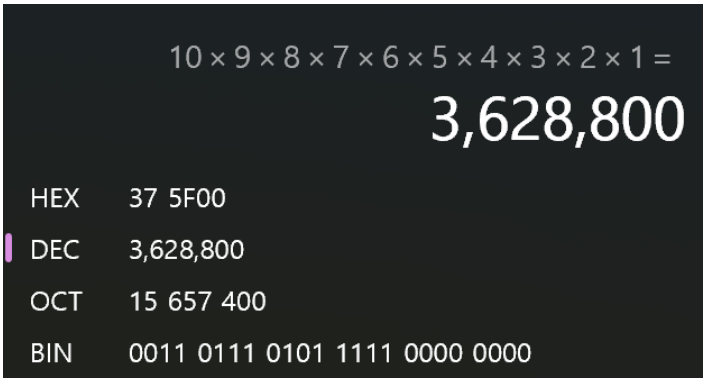
먼저 sp에 40000000의 주소를 LDR을 통해 저장한다. 그리고 R0~R7을 STMEA를 이용하여 후증가의 방식으로 sp가 가리키는 메모리에 모두 저장한다. sp의 값을 ADD를 통하여 증가시키고 LDMFD로 해당 sp에 저장된 값을 가져와서 레지스터의 값이 바뀌도록 구현했다. 이때, LDMFD는 먼저 메모리의 주소를 감소시키고 값을 가져오는 점에 유의하여 sp의 값을 조절했다. 또한 sp의 주소가 감소해야할 때는, !를 붙여서 LDMFD 명령어가 실행되면서 sp의 주소도 같이 감소하도록 구현했다.

3. Result

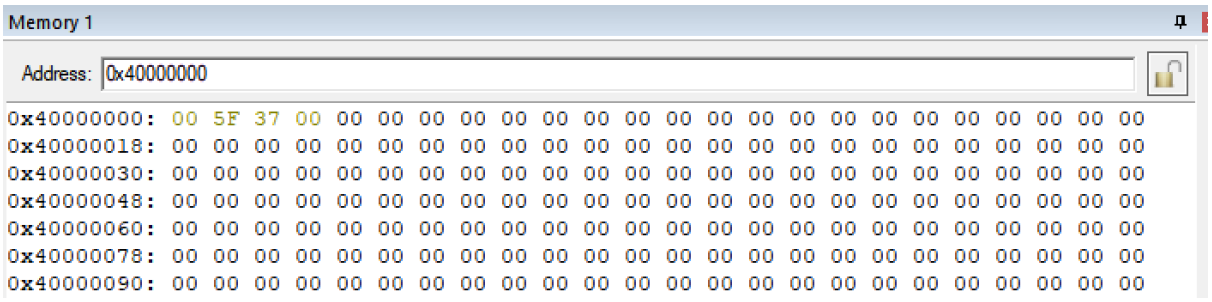
Problem1

Second operand를 이용하여 1~10까지 r1에 곱했으므로 r1에서 10!의 값이 저장되어 있을 것이다. 그리고 STR을 이용하여 해당 값을 r0의 메모리에 저장했으므로 문제1에서 요구하는 조건을 모두 만족한다고 생각한다.





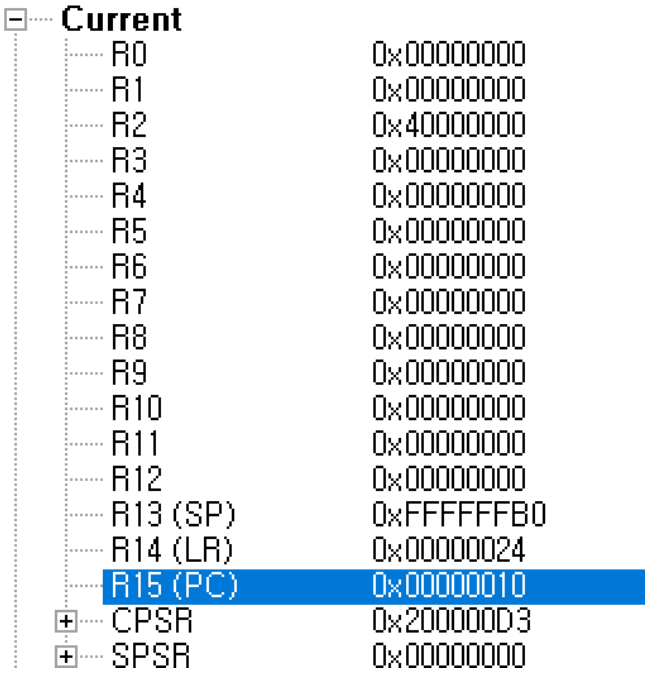
코드를 실행 후, 레지스터에 저장된 값을 보면 R1에 10!의 값이 16진수로 저장된 결과를 볼 수 있다.



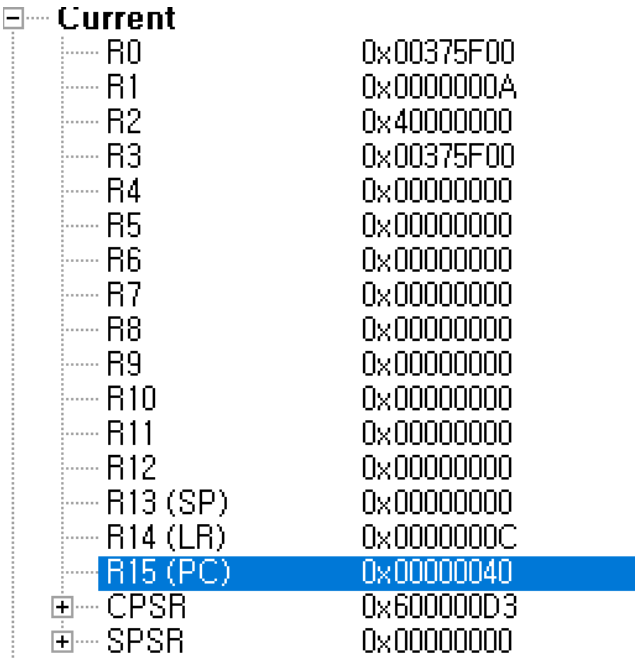
또한 결과 값이 r0의 주소인 0x40000000에 Little endian형식으로 저장된 결과를 볼 수 있다.

Problem2

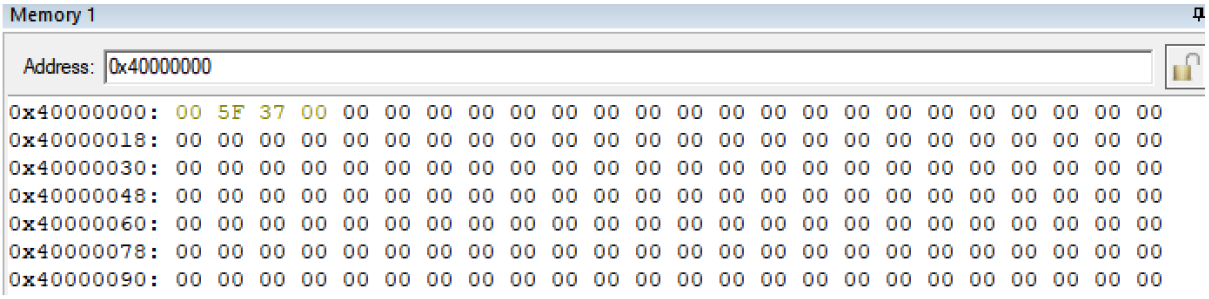
이번엔 재귀를 이용하여 10!을 계산하는 문제이다. Push를 통해서 레지스터의 상태를 스택에 저장하고 이후, Pop을 통해서 다시 레지스터를 복원하는 방식으로 구현했으므로 결과는 Problem1과 같이 10!이 저장될 것이라고 생각한다.



R0가 10~1까지 1씩 감소되면서 그 상태에 LR과 같이 스택에 Push하여 r0가 0이된 결과이다. SP를 보게되면 00000000에서 한번에 Push에 r0과 LR을 저장하므로 8비트씩 줄어들게 되어 총 80(16진수로 50)비트가 줄어든 FFFFFFB0가 된 결과를 볼 수 있다.

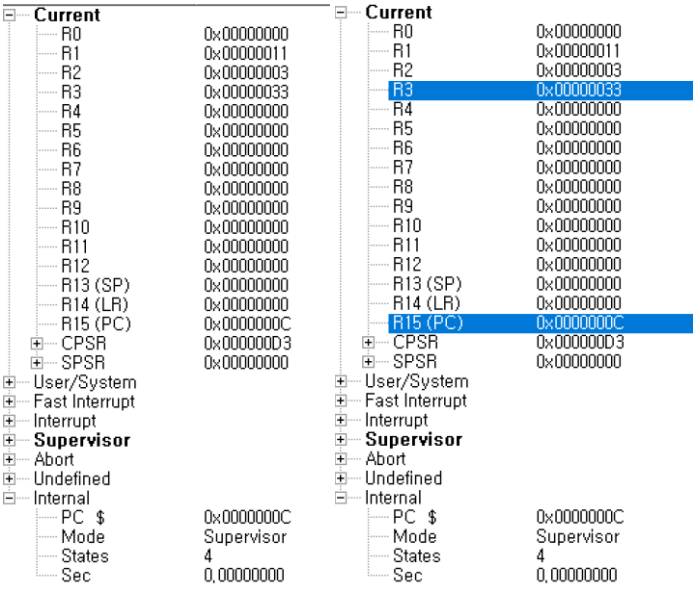


Pop을 통해서 레지스터의 값을 복원하고 R0에 1~10을 모두 곱하여 10!의 값이 저장되었고 SP는 다시 00000000으로 돌아온 결과를 볼 수 있다.



R2에 0x40000000의 주소가 저장되어 있으므로 r2가 가리키는 주소에 r0의 값(10!)을 저장한 결과이다. Problem1과 마찬가지로 Little endiand 방식으로 값이 저장되어있다.

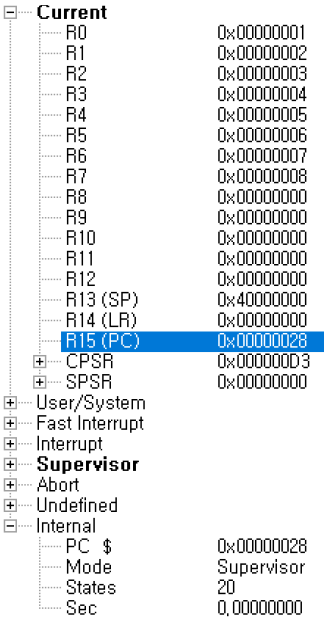
Problem3



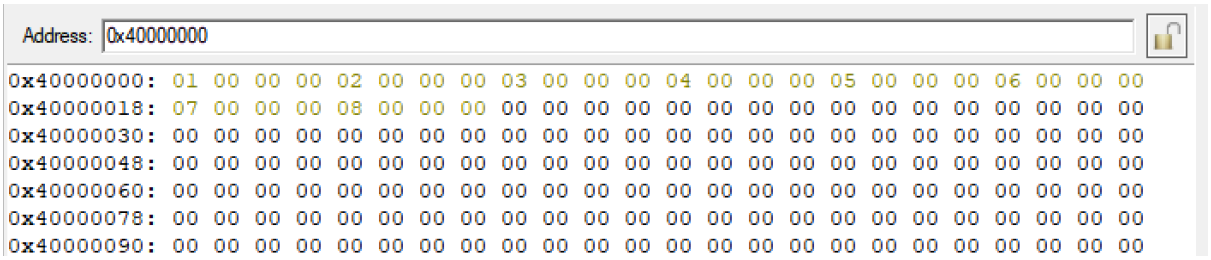
왼쪽이 17*3, 오른쪽이 3*17의 연산을 수행한 결과이다.

예전에 사용하던 ARM의 Booth 알고리즘은 2 bits 단위로 곱셈을 수행하여 Rs에 더 작은 값을 사용하면 Early termination method를 사용할 수 있어, 더 적은 Cycle로 연산이 가능하고 이는 더 효율적인 코드로 이어질 수 있었다. 하지만 현재 사용하는 ARM 아키텍쳐는 8~16 bits의 단위로 Booth 알고리즘을 적용하여 연산을 진행하는데, 17은 2진수로 00010001 이고 3은 2진수로 00000011이다. 따라서 8~16 bits 단위의 Booth 알고리즘을 적용하는 데에는 큰 차이가 없다. 따라서 현재 ARM9E-S에서 17*3과 3*17의 큰 성능 차이는 없다.

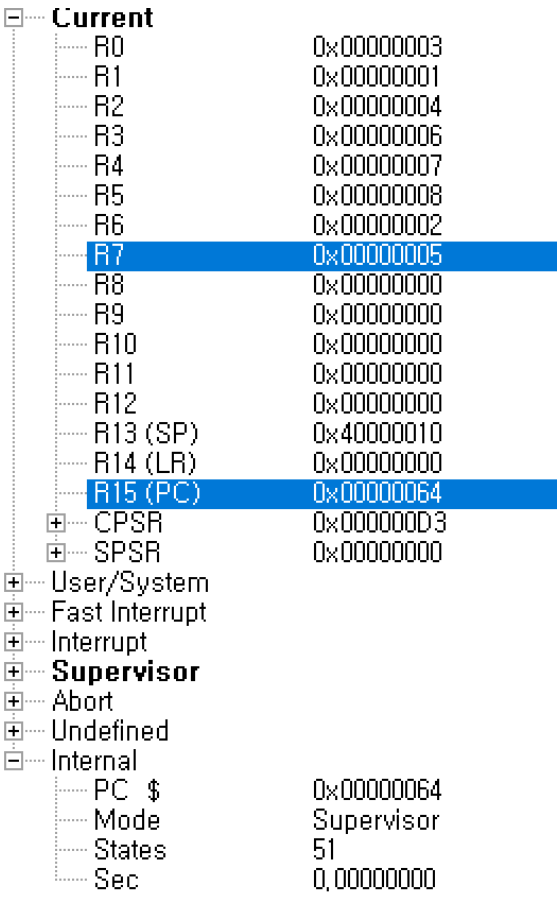
Problem4



R0~R7에 1~8의 값을 저장했다.



1~8을 0x40000000에 4byte씩 저장했다.



SP를 이용하여 0x40000000에 저장된 값을 알맞게 가져와서 레지스터의 값이 바뀐 결과이다.

Discussion and Conclusio

문제 2에서 재귀를 이용하여 facorial을 구현하는 방법을 C언어나 C++로는 많이 다루어 봤지만 어셈블리에서 막상 구현하려고 하니, 어떻게 할지 막막했다. 아직 브랜치와 레이블에 대한 개념이 부족하다는 생각이 들어, 실습시간에 질문을 통해서 개념을 다지고, 이론 시간에 배운 내용을 다시 복습하며 스택 포인터에 기능이 레지스터를 저장하고 다시 상태를 복원 할 수 있다는 점을 이용해서 문제 2번을 해결했다.

또한 과제의 공통 해당 사항으로

MOV r7, #1

MOV r0, #0
SVC 0의 코드를 이용하여 프로그램을 종료하라는 사항이 있었다. 하지만 해당 코드만 작성하면 계속 코드가 반복되는 상황이 발생하여 어떻게 종료가 되는 것인지 의아했다. 그래서 조교님께 질문도 해보고 검색을 통해 해당 코드는 ARM에서 시스템 호출을 통해 프로그램을 종료하는 코드라는 것을 알았다. 따라서 keil에서는 종료되지 않더라도 다른 프로그램이나 운영 체제에서는 잘 작동할 수 있다는 것도 깨달았다.

Reference

어셈블리프로그램설계및실습-실습 강의자료 5. Block_Data_Transfer_and_Stack
어셈블리프로그램설계및실습-이론 강의자료 Lecture3_Architecture_Programing Model
어셈블리프로그램설계및실습-이론 강의자료 Lecture6_Data_Processing Instructions

명예서약서 (Honor code)

본인은 [어셈블리프로그램설계및실습] 수강에 있어 다음의 명예서약을 준수할 것을 서약합니다.

1. 기본 원칙

- 본인은 공학도로서 개인의 품위와 전문성을 지키며 행동하겠습니다.
- 본인은 안전, 건강, 공평성을 수호하는 책임감 있는 공학도가 되겠습니다.
- 본인은 우리대학의 명예를 지키고 신뢰받는 구성원이 되겠습니다.

2. 학업 윤리

- 모든 과제와 시험에서 정직하고 성실한 태도로 임하겠습니다.
- 타인의 지적 재산권을 존중하고, 표절을 하지 않겠습니다.
- 모든 과제는 처음부터 직접 작성하며, 타인의 결과물을 무단으로 사용하지 않겠습니다.
- 과제 수행 중 받은 도움이나 협력 사항을 보고서에 명시적으로 기재하겠습니다.

3. 서약 이행

본인은 위 명예서약의 모든 내용을 이해하였으며, 이를 성실히 이행할 것을 서약합니다.

학번	2021202085	**이름**	전한아솔	**서명**	전한아솔