어셈블리 프로그래밍 설계 및 실습

실험제목: Basic Example

실험일자: 2020년 09월 8일 (화)

제출일자: 2020년 09월 13일 (일)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

학 번: 2019202052

성 명: 김호성

1. 제목 및 목적
   1. 제목

Basic Exapmle

* 1. 목적

예제 코드를 수행하여 기본적인 동작에 대해 이해하고, register 값을 확인하는 법을 숙지하기 위해 해당 과제를 진행한다.

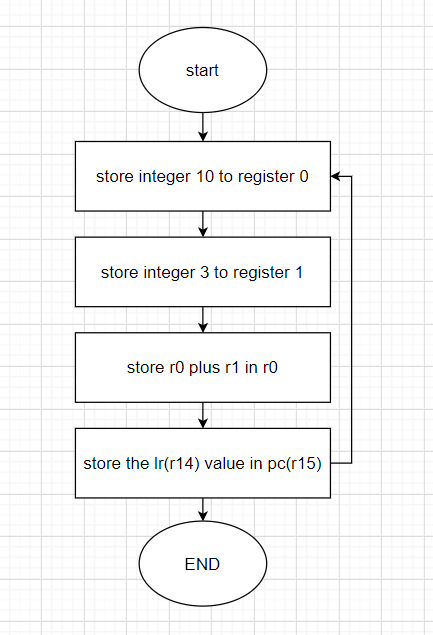
1. 설계 (Design)
   1. Pseudo code

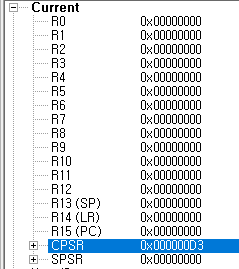
MOV R[0] 🡨 10

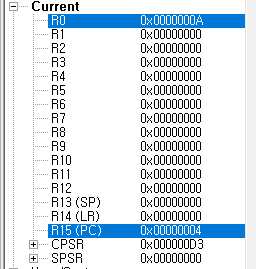
MOV R[1] 🡨 3

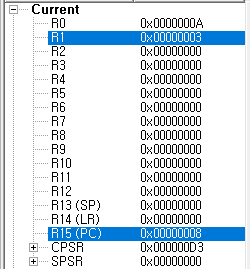
ADD R[0] 🡨 R0, R1

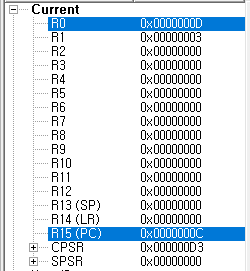
MOV PC 🡨 LR

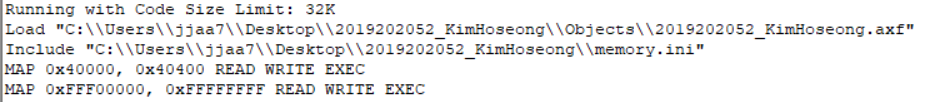
* 1. Flow chart 작성
  2. Result

1. 초기 세팅

2. MOV R[0] 🡨 10

3. MOV R[0] 🡨 R[0], R[1]

4. MOV PC 🡨 LR

5. Memory.ini

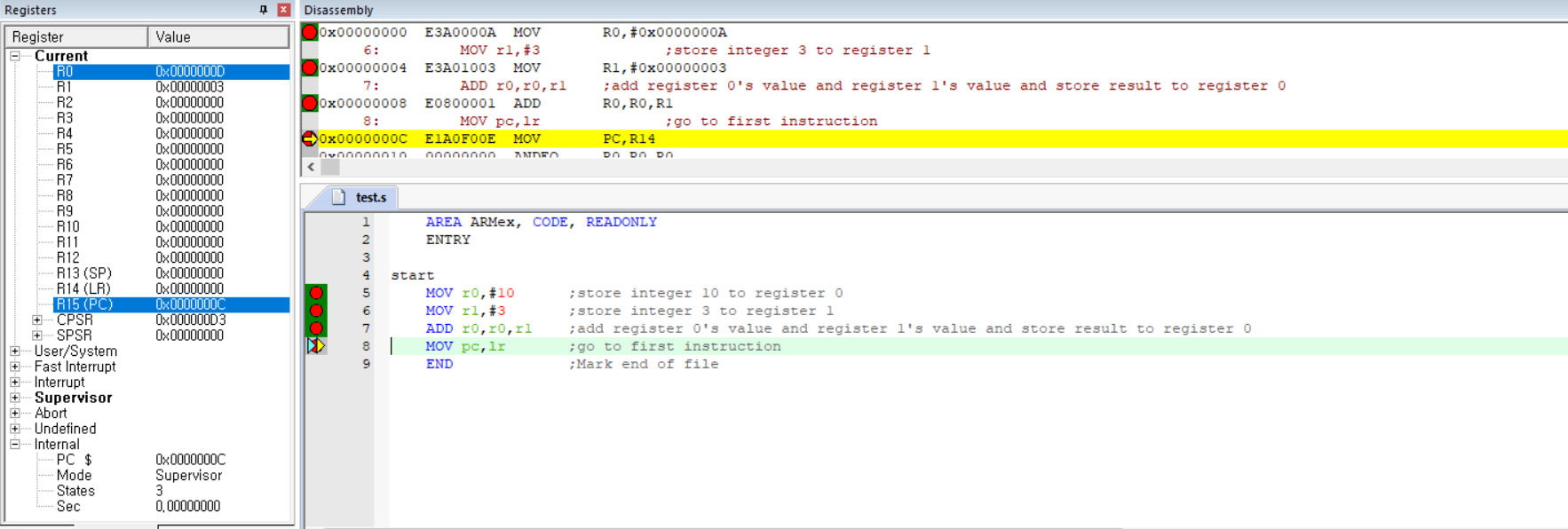
* 1. Performance

Performance = Code size \* Code size \* states

Total RO Size (Code + RO Data) 16 ( 0.02kB)

Total RW Size (RW Data + ZI Data) 0 ( 0.00kB)

Total ROM Size (Code + RO Data + RW Data) 16 ( 0.02kB)

 Code size = 16

State = 3 결론: performance = 16\*16\*3 = 768

1. 고찰 및 결론
   1. 고찰

- 레지스터의 value의 경우 16진수를 사용한다. [10 = A, 13 =D]

- MOV pc lr은 이번 과제에선 코드의 진행을 가장 처음으로 돌리는 역할을 한다.

- states의 경우 ㅣ진행 전 pc의 value – 진행 후 pc의 valueㅣ /4만큼 증가한다.

* 1. 결론

- MOV pc lr의 의미는 r14(lr) 값을 r15(pc)에 저장하라는 명령이다. 여기서 pc는 program counter의 약자로 다음에 수행할 명령어의 주소를 가지는 instruction으로, pc의 value를 제어하지 않을 경우 자동으로 value값이 4씩 증가하면서 코드를 진행한다. 여기서 lr의 value는 처음 초기화된 0x00000000 값을 가지고 있고, pc의 value에 lr의 value가 저장되면서 무한루프가 형성된다.

- 어셈블리코드 문법의 규칙은

label <space> opcode <space> operands <space> ; comment

이 순서를 지켜줘야 한다. 그래서 코드를 보면 label을 쓰지 않더라도 tab이나 스페이스바를 눌러 label의 공간을 확보한 것을 알 수 있다.

1. 참고문헌

- Basic\_Example.pdf

- 1.MDK-ARM\_Setup\_&\_Basic\_Example.pdf

- MDK-ARM\_Setup\_&\_Basic\_Example\_ver1.pdf