어셈블리 프로그래밍 설계 및 실습

실험제목: Data\_transfer\_to\_or\_from\_Mem

실험일자: 2020년 09월 15일 (화)

제출일자: 2020년 09월 21일 (월)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

학 번: 2019202052

성 명: 김 호 성

1. 제목및목적
   1. 제목

1. problem 1

2. problem 2

* 1. 목적

- 조건부 실행을 할 수 있다.

- 원하는 데이터를 메모리에 저장 및 불러오기가 가능하다.

1. 설계 (Design)
   1. Pseudo code

1.

MOV R[0] 🡨 #1, R[1] 🡨 #10, R[2] 🡨 #15, R[3] 🡨 #10

LDR R[4] 🡨 TEMPADDR12

STRB R[4] 🡨 R[0], R[4] = R[4] +1

STRB R[4] 🡨 R[1], R[4] = R[4] +1

STRB R[4] 🡨 R[2],

LDRB R[0] 🡨 R[4, #-2]! ; R[0] = #1

CMP R[0], R[3]

MOVMI R[5] 🡨 #2

MOVGT R[5] 🡨 #1

MOVEQ R[5] 🡨 #3

LDRB R[0] 🡨 R[4, #1]!; R[0] = #10

MOVMI R[5] 🡨 #2

MOVGT R[5] 🡨 #1

MOVEQ R[5] 🡨 #3

LDRB R[0] 🡨 R[4, #1] ; R[0] = #15

MOVMI R[5] 🡨 #2

MOVGT R[5] 🡨 #1

MOVEQ R[5] 🡨 #3

TEMPADDR12 🡨 0x40000

MOV PC 🡨 LR

END

2.

MOV R[0] 🡸 #1, MOV R[1] 🡨 #2, MOV R[2] 🡨 #3, MOV R[3] 🡨 #4

LDR R[4] 🡨 TEMPADDR1

LDR R[6] 🡨 TEMPADDR2

STRB R[4] 🡨 R[3], R[4] = R[4] + 1

STRB R[4] 🡨 R[2], R[4] = R[4] + 1

STRB R[4] 🡨 R[1], R[4] = R[4] + 1

STRB R[4] 🡨 R[0]

STRB R[6] 🡨 R[0], R[6] = R[6] + 1

STRB R[6] 🡨 R[1], R[6] = R[6] + 1

STRB R[6] 🡨 R[2], R[6] = R[6] + 1

STRB R[6] 🡨 R[3], R[6] = R[6] + 1

TEMPADDR1 🡨 0x40000

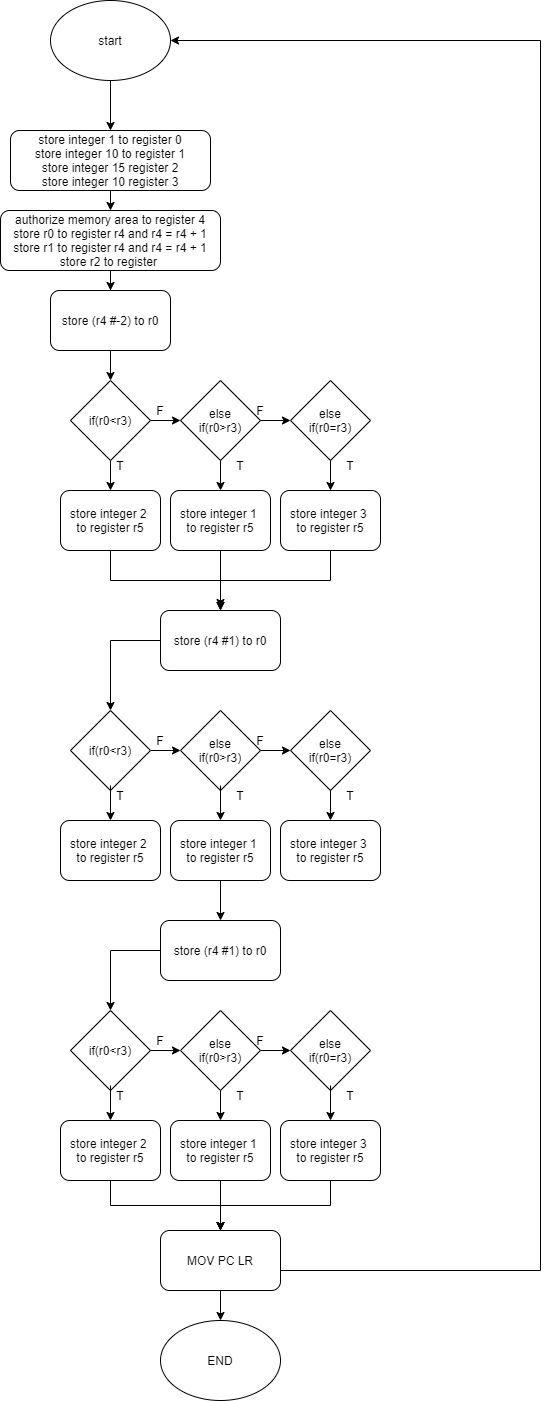
TEMPADDR2 🡨 0x40200

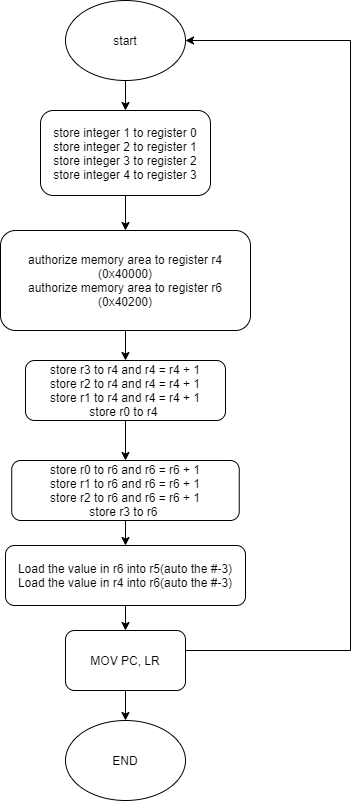
LDR R[5] 🡨 R[6, #-3]!

LDR R[6[ 🡨 R[4, #-3]!

MOV PC 🡨 LR

* 1. Flow chart 작성

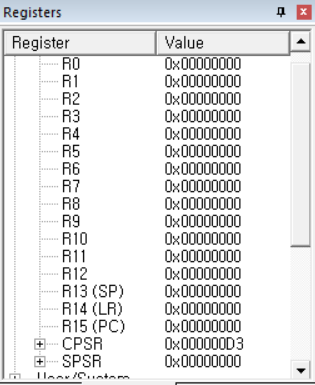
1.

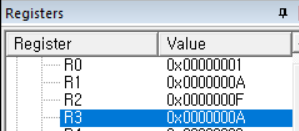
2.

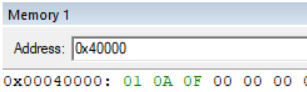
* 1. Result

1.

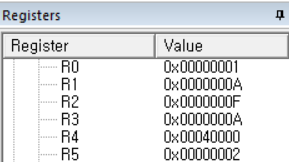
초기 세팅



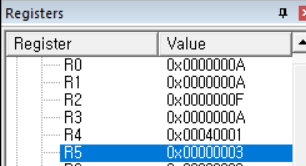
MOV R[0] ~ R[3]

Store R[0] ~ R[3] to memory(0x40000)

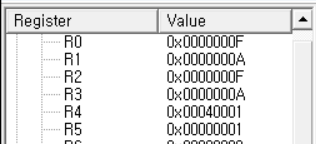
Compare r0 with r3 and store the value in r5 for the condition

(r0 = 1)

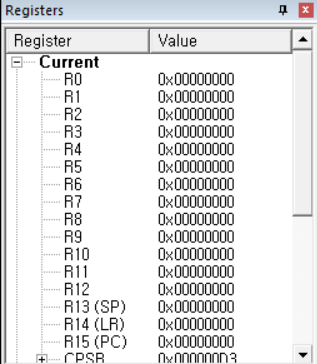
Compare r0 with r3 and store the value in r5 for the condition

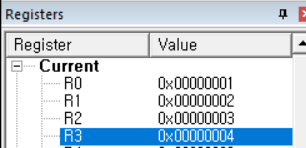
(r0 = 10)

Compare r0 with r3 and store the value in r5 for the condition

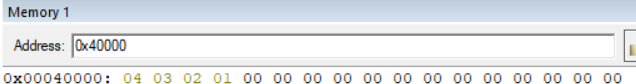
(r0 = 15)

2.

초기 세팅

MOV R[0] ~ R[3]

authorize memory area to use (r4 = 0x40000, r6 =0x40200)

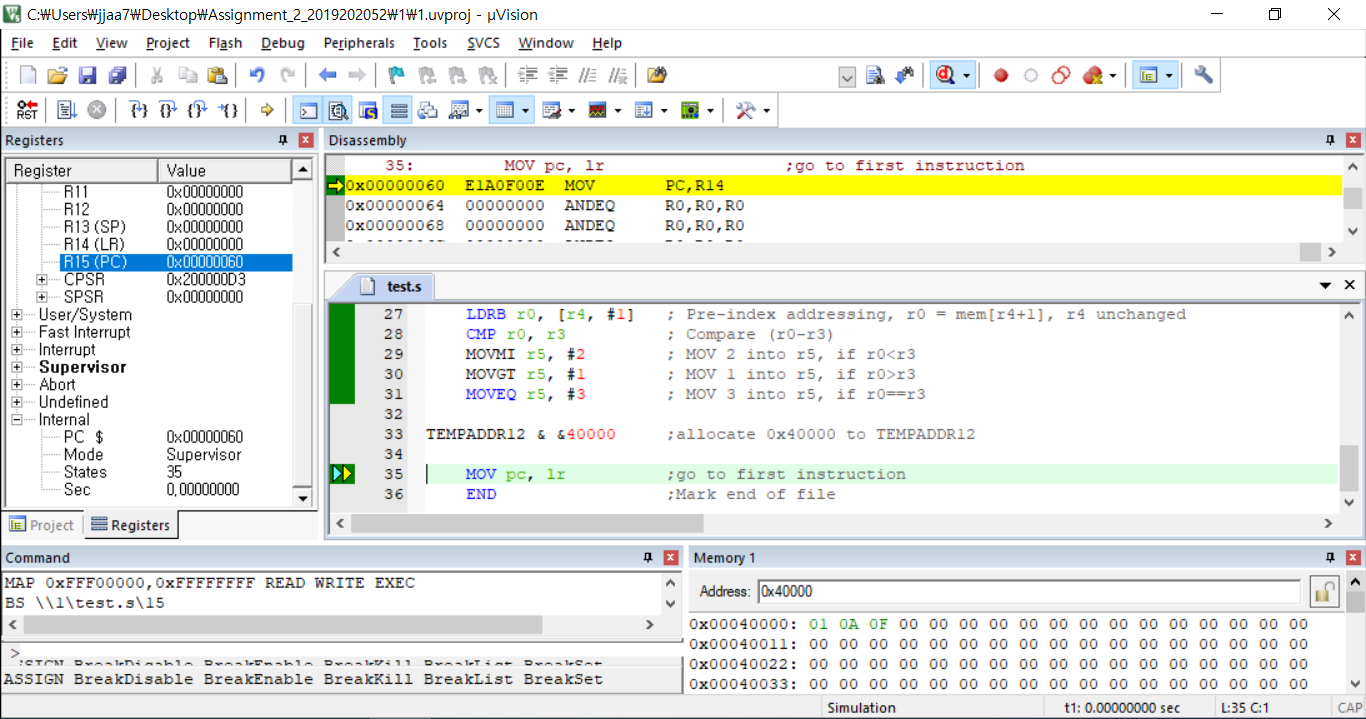
Store R[3] ~ R[0] to memory(0x40000)

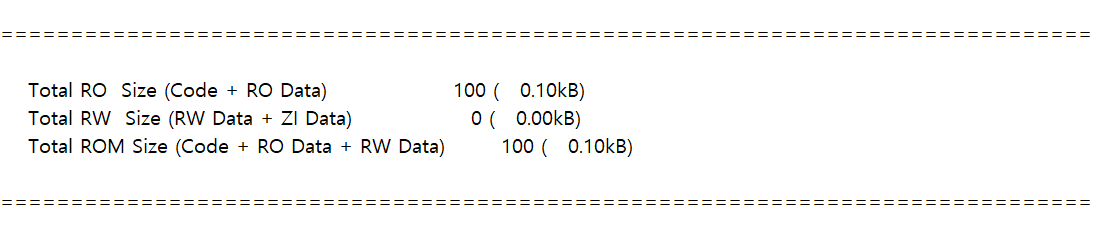
Store R[0] ~ R[3] to memory(0x40200)

Load R[6, #-3]! to R5(auto) and Load R[4, #-3]! to R[6]

* 1. Performance

1.

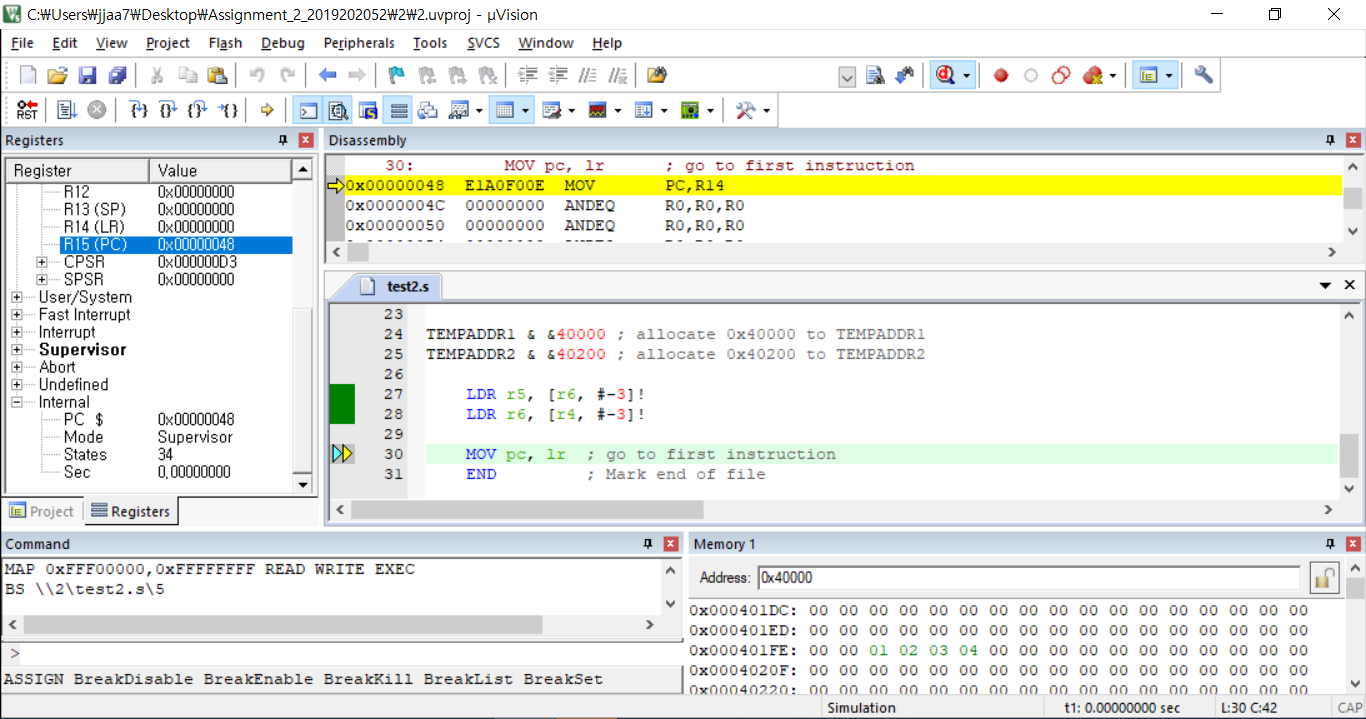
Performance = Code size \* Code size \* states=

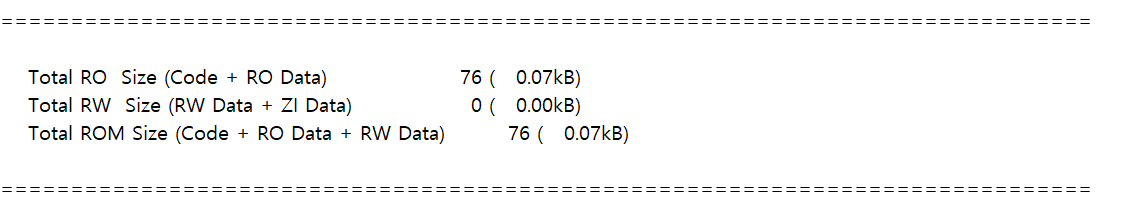
State = 35

Code size = 100

Performance = 350000

2.

Performance = Code size \* Code size \* states =

 State = 34

Code size = 76

Performance = 196,384

1. 고찰 및 결론
   1. 고찰

이번 실습을 통해서 두 개의 값을 비교하여 조건문을 연산하는 것과, 메모리로 데이터 저장(store) 및 가져오기(load)에 대해 배울 수 있었다. 첫 번째 문제는 메모리에 저장된 숫자 3개를 STRB를 활용하여 지정된 memory에 값을 저장하였다. Memory에 저장된 값을 차례로 LDRB 명령어를 통해 값을 읽어 명령어 CMP를 이용하여 0x0A와 값을 비교하여 flag를 update할 수 있도록 하였다. Update된 flag 값에 따라 MOV가 작동할 수 있도록 conditional field인 GT, EQ, LT를 이용하였다. 두 번째 문제는 명령어 MOV를 이용하여 각 register에 값을 저장한 후, r4 register에 메모리 주소 값을 저장하였다. 명령어 STRB를 활용하여 memory에 각각 r0, r1, r2, r3에 저장된 값을 저장할 수 있게 하였다. LDR를 통하여 r4 – 4 주소 값에 저장되어 있는 값을 register r5로 저장할 수 있게 하였다. 그리고 r0, r1, r2, r3의 값을 다른 memory에 거꾸로 store하여 거꾸로 load하여 r6에 값을 저장하였다.

* 1. 결론

- LDR: Load의 명령어로 첫 번째 인자는 register, 두 번째 인자는 주소로 쓰이게 된다.

- STR: Store의 명령어로 첫 번째 인자는 register, 두 번째 인자는 주소로 쓰이게 된다.

LDR과 STR 명령어는 ini파일을 통해 사용하고자 하는 메모리 영역에 read 또는 write 권한을 부여해야한다.

- MOV pc, lr

코드의 진행을 처음으로 돌리는 역할이다.

1. 참고문헌

이준환교수님/어셈블리설계및실습/광운대학교(컴퓨터정보공학부)/2020