

어셈블리 프로그래밍 설계 및 실습 보고서

실험제목: Control flow & Data processing

실험일자: 2018년 10월 02일 (화)

제출일자: 2018년 10월 09일 (화)

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습 분반: 화 6,7 , 수 5

학 번: 2017202010

성 명: 박유림

1. 제목 및 목적

A. 제목

Second_Operand_&_Multiplication

B. 목적

Second Operand가 무엇인지에 대해서 배우고 이를 사용해 실제 연산에 도입해 본다. Multiplication operation을 사용하는 것과 Second operand를 사용한 코드의 성능 차이를 비교해 본다. Operand의 순서에 따른 성능의 차이에 대해 알아본다,

2. 설계 (Design)

A. Pseudo code

1. Problem1

$R0 = 0x40000$

$R1 = 1$

$R2 = R1 \ll 1$

$R3 = R2 + (R2 \ll 1)$

$R4 = R3 \ll 2$

$R5 = R4 + (R4 \ll 2)$

$R6 = (R5 \ll 1) + (R5 \ll 2)$

$R7 = (R6 \ll 3) - R6$

$R8 = (R7 \ll 3)$

$R9 = R8 + (R8 \ll 3)$

$R10 = R9 + \{ R9 + (R9 \ll 3) \}$

$[R0] = R10$

2. Problem2

$R0 = 0x40000$

$R1 = 1$

$R2 = 2$

$R3 = 3$

$R4 = 4$

$R5 = 5$

$$R6 = 6$$

$$R7 = 7$$

$$R8 = 8$$

$$R9 = 9$$

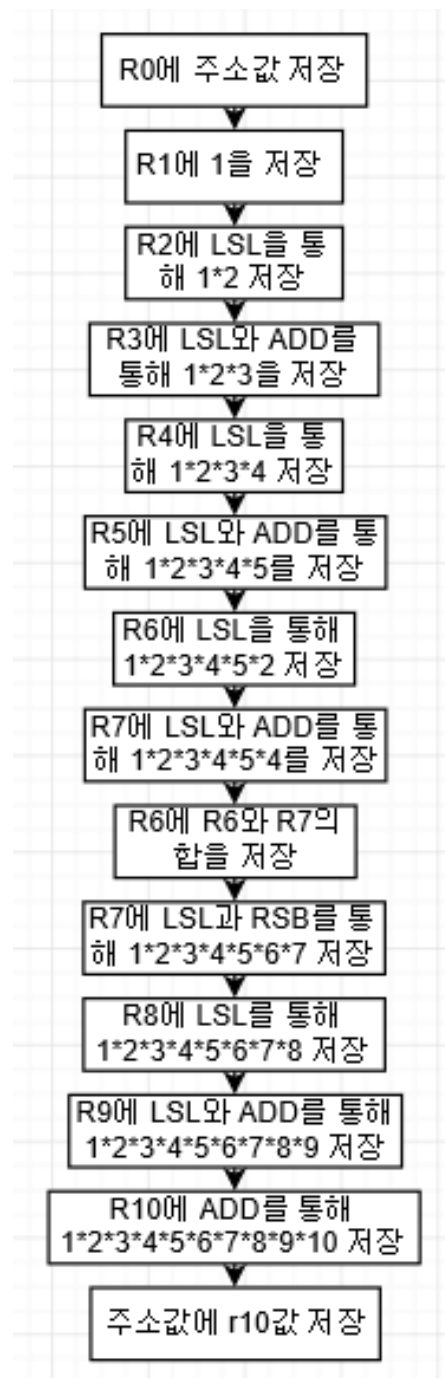
$$R10 = 10$$

$$R11 = R1 * R2 * R3 * R4 * R5 * R6 * R7 * R8 * R9 * R10$$

$$[R0] = R11$$

B. Flow chart 작성

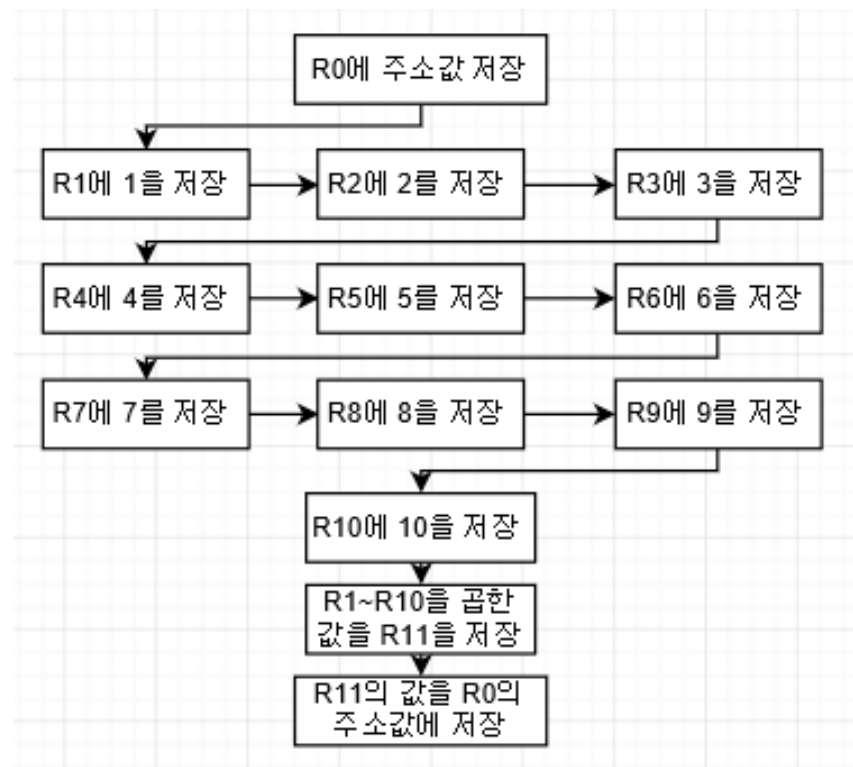
1. Problem1



R0에 tempaddr의 주소값을 준다. 그리고 아래에 0x40000의 값을 tempaddr에 할당해준다. R1에 1의 값을 mov를 통해 저장해준다. R2에 R1에 LSL을 1만큼 해서 넣어준다. 이는 R1에 2를 곱한것과 같으므로 결과적으로 $1*2$ 이 R2에 저장되는 것이다. R3에 R2와 R2를 LSL을 1만큼 한 값을 더해서 넣어준다. 이는 $1*2+1*2*2=1*2*(1+2)$ 이므로 결과적으로 R3에 $1*2*3$ 이 들어간다. R4에는 이 R3의 값에 LSL을 2만큼한 결과를 저장해준다. 이는 R3의 값에 4를 곱해준 것이므로 $1*2*3*4$ 의 결과값이 R4에 저장

된 것이다. R3를 만든 것과 비슷하게 R5에 대입해준다. 이는 $1*2*3*4+1*2*3*4*4=1*2*3*4*(1+4)=1*2*3*4*5$ 가 된다. R6는 $1*2*3*4*5*2$ 와 $1*2*3*4*5*4$ 를 더해 만들어준다. R7은 $1*2*3*4*5*6*8-1*2*3*4*5*6$ 로 계산이 가능하다. 비슷한 방법으로 쭉 이어 나가면 R10인 $1*2*3*4*5*6*7*8*9*10$ 이 Second Operand를 통해 나오게 된다. R10에 저장된 값을 R0에 저장된 memory의 주소에 넣어준다.

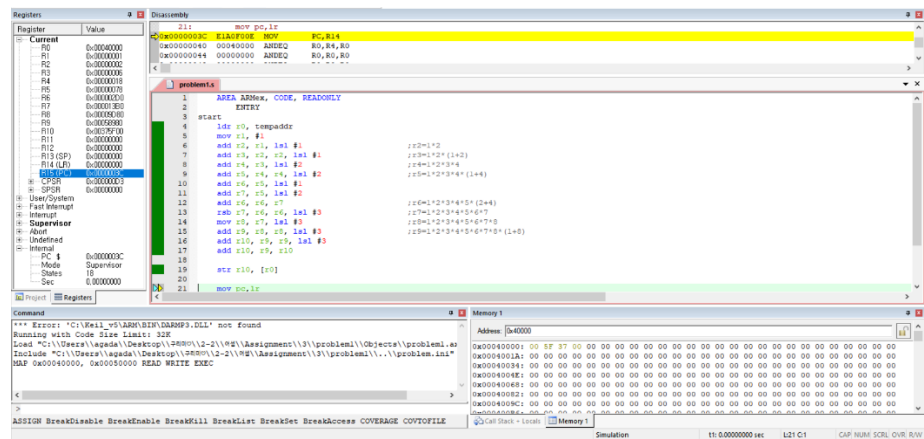
2. Problem2



Mov를 통해 R1에 1을, R2에 2를, 계속해서 R3~R10에 각각 1, 2, 3, 4, 5, 6, 7, 8, 9, 10의 값을 저장해주고 mul을 통해서 R11에 $1*2*3*4*5*6*7*8*9*10$ 의 계산 결과값을 저장한다. 마지막으로 R11에 저장된 값을 R0에 저장된 memory의 주소에 넣어준다.

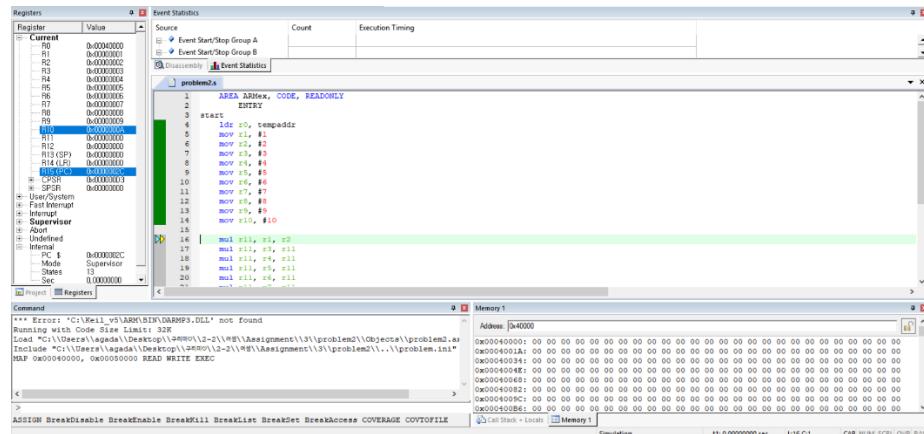
C. Result

1. Problem1

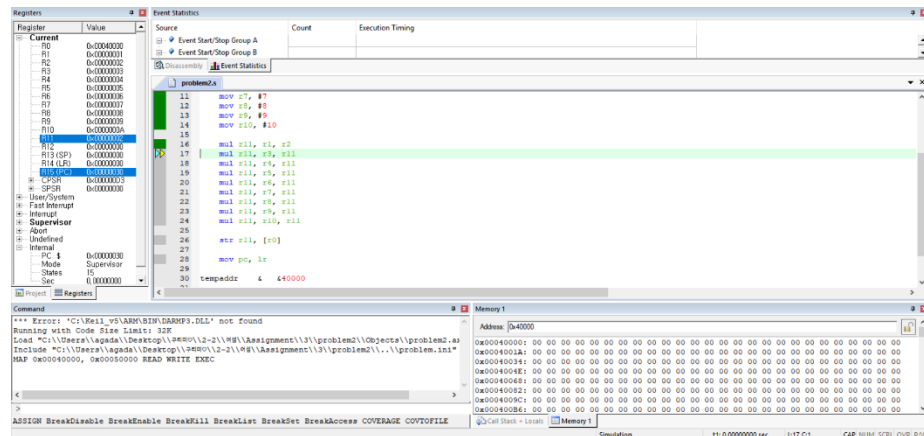


R0에 0x40000가 들어가고 순서대로 R1=1!, R2=2!, R3=3!, R4=4! ... R10=10!의 값이 들어간 것을 확인할 수 있다. 마지막으로 memory의 0x40000번지에 R11의 값인 375F00이 Little Endian방식으로 삽입이 된 것을 확인할 수 있다.

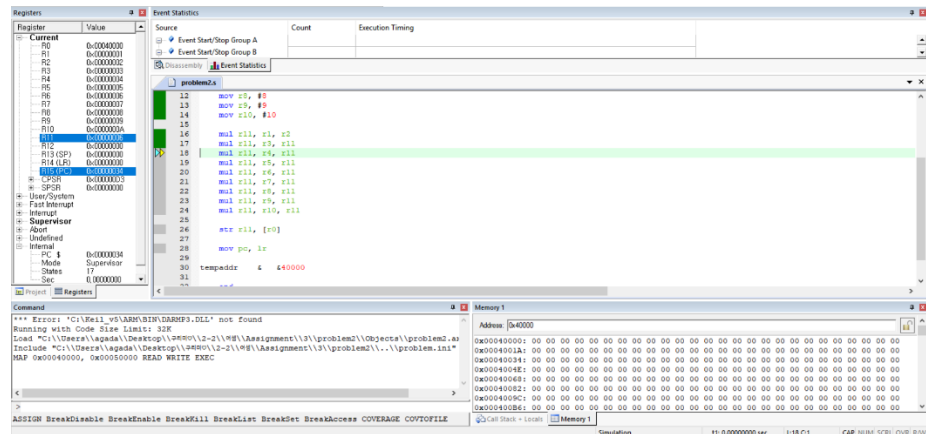
2. Problem2



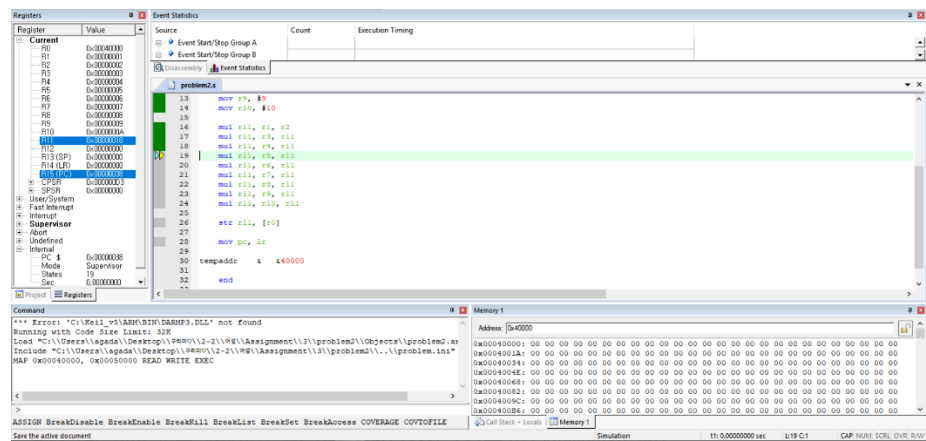
R0에 0x40000가 들어가고 순서대로 R1=1, R2=2, R3=3, R4=4 ... R10=10의 값이 들어간 것을 확인할 수 있다.



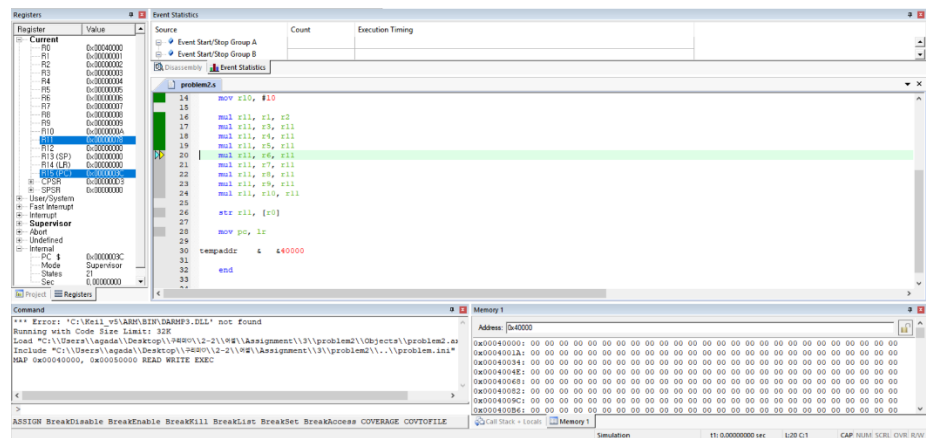
R11에 R1과 R2를 곱한 값이 삽입됐다.



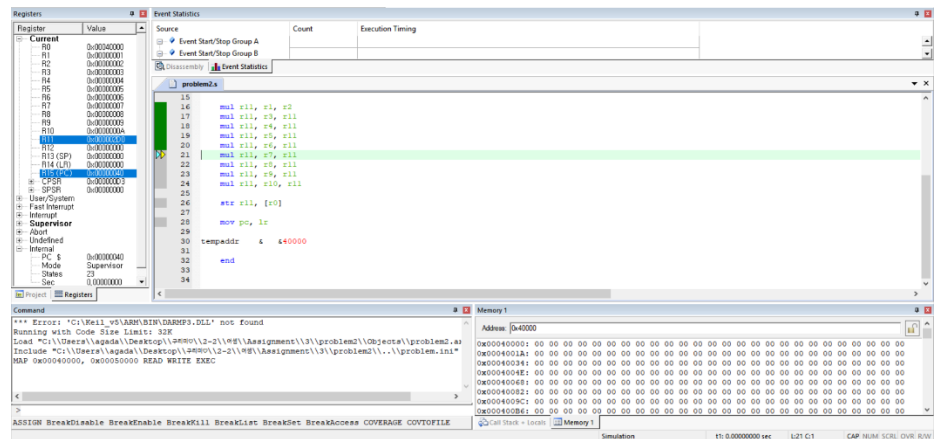
R11에 R1, R2, R3를 곱한 값이 삽입됐다.



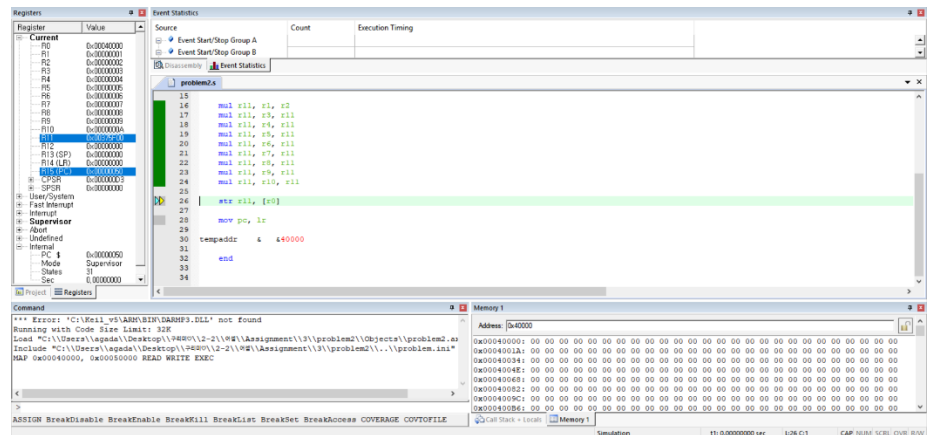
R11에 R1, R2, R3, R4를 곱한 값이 삽입됐다.



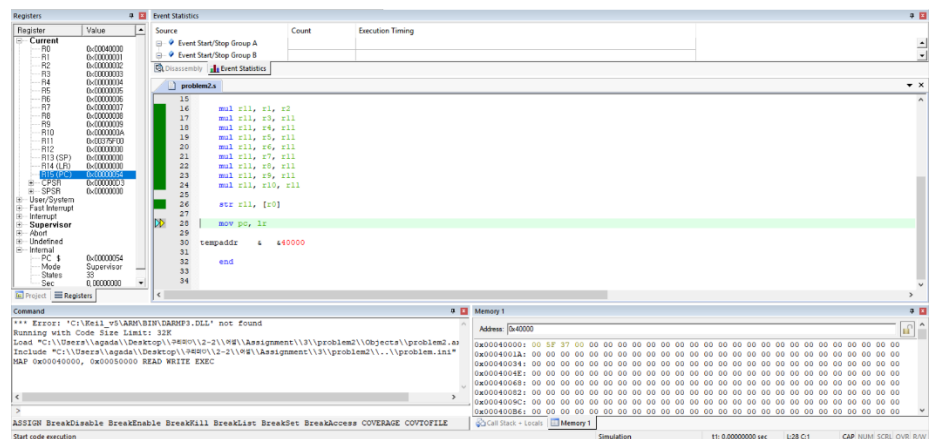
R11에 R1, R2, R3, R4, R5를 곱한 값이 삽입됐다.



R11에 R1, R2, R3, R4, R5, R6를 곱한 값이 삽입됐다. 이와 같은 방식을 4번 더 반복하게 되면 $R11 = R1 * R2 * R3 * R4 * R5 * R6 * R7 * R8 * R9 * R10$ 가 저장이 된다.



R11의 값을 R0에 저장된 주소값에 저장이 된다.



결과적으로 R11과 MEMORY의 0x40000번지에 375F00의 값이 들어가 있는 것을 확인할 수 있다.

3. 고찰 및 결론

A. 고찰

이번 문제는 factorial의 값을 두개의 다른 연산으로 표현하는 과제였다. 하나는 이번 시간에 배운 Second Operand이고 나머지는 Multiplication Operation이었다.

Problem1에서는 이번 시간에 배운 Second Operand를 이용하여 10!을 계산하였다. 수업을 들으면서 Second Operand에 대해서 조금은 이해를 했지만 완벽히 하지 못한 것을 스스로 알고 있었다. 이에 강의자료에 나와있는 예시를 반복하여 읽으며 Second Operand의 원리와 강점에 대해 생각해보았다. 후에 이번 과제를 진행하게 되었다.

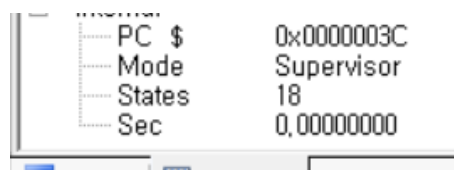
다음으로는 숫자의 진수에 관한 착각이었다. 일상생활에서는 10진수를 쓰기 때문에 이에 익숙해져 있었다. Debugging을 하던 중에 $1*2*3$ 까지는 6이라는 값이 잘 나오다가 $1*2*3*4$ 가 18이 나오는 것을 확인 할 수 있었다. 16진수로 하면 24로 올바른 값이 들어간 것 이지만 10진수라고 착각하고 있었기 때문에 원리에 대해 잘 못 이해한 것이라고 판단하여 강의자료와 본인의 코드를 반복하여 보며 허점을 찾으려 했지만 찾을 수 없었다. 과제자료에 memory주소가 0x40000번지로 나와있는 것을 보고 10진수가 아닌 16진수로 어셈블리는 주로 표현한다는 것을 다시 각인되고 문제를 해결할 수 있었다.

B. 결론

4. Performance

이번 과제에서는 같은 문제를 두가지의 방식으로 프로그래밍하였다. 그리고 그 결과 어느 방식의 performance가 더 효과적인지 알아보기 위해서 states의 수를 비교하여 본다.

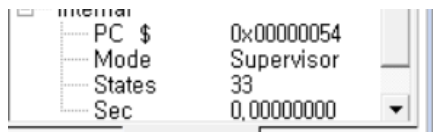
A.Problem1_Second Operand



PC	\$	0x0000003C
Mode		Supervisor
States		18
Sec		0,00000000

States 수 : 18

B.Problem2_Multiplication Operation



PC \$	0x00000054
Mode	Supervisor
States	33
Sec	0,00000000

States수 : 33

States 수로 판단하였을 때, performance는 1번째 방법인 Second Operand가 2번째 방법인 Multiplication Operation보다 뛰어났다.

5. 참고문헌

이형근/ 어셈블리프로그래밍/ 광운대학교/ 2018