

[숙제 8] lab4 실습 내용

(1) 실습 A 관련(LAB4 P20)

소스 코드(matrix.s) 18번 라인에 break point를 주고, r6(s) = 1, r7(u) = 0, r8(v)=2가 되는 시점부터 trace 하세요. lab4 자료 P20 아래 그림 상에서 1번 값과 2번 값이 곱해져 3번에 저장되는 과정에서, 1번 ~ 3번에 해당하는 메모리 번지 값(R11), 메모리 내용 값(1번 및 2번 읽어온 값, 3번에 저장한 값)을 캡처하시오. 캡처 결과가 어느 것에 해당 하는지 표시하세요.

1번값 : 4, 2번값 : 7, 3번값 : 54

```

Array A
1 2 3
4 5 6
7 8 9
Array B
9 8 7
6 5 4
3 2 1
Array C after Operating :
30 24 18
84 69 54
138 114 90
    
```

1번

```

1  .text
2  .global matrix
3
4  matrix:
5      stmf sp!, {r0-r12,lr}
6
7      mov     r6,#0
8      mov     r7,#0
9      mov     r8,#0
10     mov     r9,#0
11
12  Loop:
13     mov     r11,#12
14     mul     r11,r6,r11
15     add     r11,r11,r7,LSL #2
16
17     add     r11,r11,r6
18
19     ldr     r12,[r11]
20
21     mov     r11,#12
22     mul     r11,r7,r11
23     add     r11,r11,r8,LSL #2
24     add     r11,r11,r1
25
26     ldr     r11,[r11]
27
28     mul     r12,r11,r12
29     add     r9,r9,r12
30
31     add     r7,r7,#1
32     cmp     r7,r3
33     bne     Loop
34     mov     r7,#0
35
36     mov     r11,#12
37     mul     r11,r6,r11
38     add     r11,r11,r8,LSL #2
39
40     add     r11,r11,r2
41
            
```

threads
local variables
expressions
Tree

memory
0xfffffc 0xffff01b 12

address	hex
more	
0xffffeffc	04 00 00 00 05 00 00 00 06 00 00 00
0xffffef08	07 00 00 00 08 00 00 00 09 00 00 00
0xffffef04	09 00 00 00 08 00 00 00
more	

breakpoints
signals

registers

name	value (hex)	value (decimal)	description
r0	0xffffeff0	4294897648	
r1	0xffffef04	4294897684	
r2	0xffffef038	4294897720	
r3	0x3	3	
r4	0x110a8	69800	
r5	0x10158	65880	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x89b04	563972	register 10 (64-bit)
r11	0xffffeffc	4294897660	register 11 (64-bit)
r12	0xc	12	register 12 (64-bit)

2번

```

1 .text
2 .global matrix
3
4 matrix:
5     stmfid sp!, [r0-r12,lr]
6
7     mov     r6,#0
8     mov     r7,#0
9     mov     r8,#0
10    mov     r9,#0
11
12 Loop:
13     mov     r11, #12
14     mul     r11,r6,r11
15     add     r11,r11,r7,LSL #2
16
17     add     r11,r11,r0
18
19     ldr     r12,[r11]
20
21     mov     r11, #12
22     mul     r11,r7,r11
23     add     r11,r11,r8,LSL #2
24     add     r11,r11,r1
25
26     ldr     r11,[r11]
27
28     mul     r12,r11,r12
29     add     r9, r9, r12
30
31     add     r7,r7,#1
32     cmp     r7,r3
33     bne     Loop
34     mov     r7,#0
35
36
37     mov     r11, #12
38     mul     r11,r6,r11
39     add     r11,r11,r8,LSL #2
40
41     add     r11,r11,r2

```

memory

0xfffff01c 0xfffff03b 12

address	hex
more	
0xfffff01c	07 00 00 00 06 00 00 00 05 00 00 00
0xfffff028	04 00 00 00 03 00 00 00 02 00 00 00
0xfffff034	01 00 00 00 1e 00 00 00
more	

breakpoints

signals

registers

name	value (hex)	value (decimal)	description
r0	0xffffeff0	4294897648	
r1	0xfffff014	4294897684	
r2	0xfffff038	4294897720	
r3	0x3	3	
r4	0x110a8	69800	
r5	0x10158	65880	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x89b04	563972	register 10 (64-bit)
r11	0xfffff01c	4294897692	register 11 (64-bit)
r12	0x4	4	register 12 (64-bit)
sp	0xffffefa8	4294897576	
lr	0x10800	67584	
pc	0x10918	67864	

3번

```

41     mov     r11, #12
42     mul     r11,r7,r11
43     add     r11,r11,r8,LSL #2
44     add     r11,r11,r1
45
46     ldr     r11,[r11]
47
48     mul     r12,r11,r12
49     add     r9, r9, r12
50
51     add     r7,r7,#1
52     cmp     r7,r3
53     bne     Loop
54     mov     r7,#0
55
56
57     mov     r11, #12
58     mul     r11,r6,r11
59     add     r11,r11,r8,LSL #2
60
61     add     r11,r11,r2
62
63     str     r9, [r11]
64
65     mov     r9,#0
66
67     add     r8,r8,#1
68     cmp     r8, r3
69     bne     Loop
70
71     mov     r8,#0
72
73     add     r6,r6,#1
74     cmp     r6,r3
75     bne     Loop
76
77     ldmfd sp!, [r0-r12,pc]
78
79 .end
80
81

```

memory

0xfffff038 0xfffff057 12

address	hex
more	
0xfffff038	1e 00 00 00 18 00 00 00 12 00 00 00
0xfffff044	54 00 00 00 45 00 00 00 36 00 00 00
0xfffff050	08 b4 08 00 58 01 01 00
more	

breakpoints

signals

registers

name	value (hex)	value (decimal)	description
r0	0xffffeff0	4294897648	
r1	0xfffff014	4294897684	
r2	0xfffff038	4294897720	
r3	0x3	3	
r4	0x110a8	69800	
r5	0x10158	65880	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x36	54	register 9 (64-bit)
r10	0x89b04	563972	register 10 (64-bit)
r11	0xfffff04c	4294897740	register 11 (64-bit)
r12	0x6	6	register 12 (64-bit)
sp	0xffffefa8	4294897576	
lr	0x10800	67584	
pc	0x10948	67912	

(2) 실습 B 관련(LAB4 P21)

C 프로그램상에서 다른 크기의 s, u, v 를 입력받고 malloc 을 이용하여 matrix A[s,u], B[u,v], C[s,v] 를 위한 배열 메모리를 할당합니다 . 두 matrix A, B 의 각 원소값을 입력 받아 각각 초기화합니다. 매트릭스 인덱스 s, u, v 값이 저장된 3 x 1 array 1개 D 를 선언 합니다. C 프로그램에서 Matrix A, B, C 에 대한 pointer값 3 개, D 의 시작번지를 argument 로 하여 어셈블리 함수를 호출합니다. 어셈블리 코드에서는 matrix 곱셈을 수행 하여 결과를 Matrix C 에 저장합니다. (Lab4 자료 P22, 23, 24, 27 내용대로 구현하세요). Test를 아래와 같이 수행하여 결과를 제출하세요.

```
.text
```

```
.global matrix
```

```
matrix:
```

```
    stmfd sp!, {r0-r12,lr}           # 사용할 레지스터 스택영역에 push

    ldr    r4, [r3, #4]               # u 값을 r4에 로드
    ldr    r5, [r3, #8]               # v 값을 r3에 로드
    ldr    r10, [r3, #0]              # s 값을 r10에 로드
    mov    r6,#0                      # s까지의 인덱스
    mov    r7,#0                      # u까지의 인덱스
    mov    r8,#0                      # v까지의 인덱스
    mov    r9,#0                      # 곱연산을 저장함
```

```
Loop:
```

```
    mov    r11, r0                    # A의 원소를 저장할 레지스터
    ldr    r11, [r11, r6, lsl#2]

    ldr    r11, [r11, r7, lsl#2]      # r6, r7 인덱스에 대한 값을 r11에 로드

    mov    r12, r1                    # B의 원소를 저장할 레지스터
    ldr    r12, [r12, r7, lsl#2]

    ldr    r12, [r12, r8, lsl#2]      # r7, r8 인덱스에 대한 값을 r12에 로드

    mul    r12, r11, r12              # A와 B의 원소를 곱해 r12에 저장
    add    r9, r9, r12               # 곱한 값을 r9에 누적시킴

    add    r7, r7, #1                # r7 인덱스를 증가시킴
    cmp    r7, r4                    # 인덱스값이 u와 같을 때 까지 반복
    bne    Loop
    mov    r7, #0                    # r7에 0을 저장
```

```

mov    r11, r2                # C의 원소를 저장할 레지스터
ldr    r11, [r11, r6, lsl#2]

add    r11, r11, r8, lsl#2    # r6, r8 인덱스에 대한 값을 r11에 지정

str    r9, [r11]              # 연산이 끝난 r9을 r11위치에 저장

mov    r9, #0                  # r9 초기화

add    r8,r8,#1                # r8 인덱스 증가
cmp    r8, r5
bne    Loop                    # 인덱스값이 v와 같을 때 까지 반복

mov    r8,#0                   # r8 인덱스 초기화

add    r6,r6,#1                # r6 인덱스 증가
cmp    r6,r10
bne    Loop                    # 인덱스값이 s와 같을 때 까지 반복

ldmfd sp!, {r0-r12,pc}        # 사용한 레지스터 스택에서 pop, 함수 호출
                                # 이전으로 되돌림

```

.end

- s, u, v 값을 전부 3 으로 입력하고 Matrix 배열 메모리를 할당받고 위와 같이 초기화 한후 Matrix 곱셈 수행후 결과 출력 후 캡처 첨부

```
Sungwon@ubuntu:~/lab4$ ./lab4
input values of s, u, v333
This is 3 by 3 matrix version

Input value of A : 123123123

Input value of B : 111222333

Array A
1 2 3
1 2 3
1 2 3
Array B
1 1 1
2 2 2
3 3 3
Array C after Operating :
14 14 14
14 14 14
14 14 14
Sungwon@ubuntu:~/lab4$
```

- s=3, u=4, v=5 로 입력하고 Matrix 배열 메모리를 할당받고 위와 같이 초기화 한후 Matrix 곱셈

```
Sungwon@ubuntu:~/lab4$ ./lab4
input values of s, u, v345
This is 3 by 3 matrix version

Input value of A : 123412341234

Input value of B : 11111222223333344444

Array A
1 2 3 4
1 2 3 4
1 2 3 4
Array B
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
Array C after Operating :
30 30 30 30 30
30 30 30 30 30
30 30 30 30 30
```

수행후 결과출력 후 캡처 첨부

- s, u, v 는 1보다 크고 9보다 작은 수를 사용
- 어셈블리 코드에 주석을 추가하세요. 결과 화면을 캡처하세요. - C 및 어셈블리 소스 코드를 파일로 제출하세요. 보고서와 소스코드를 압축하여 하나의 file(파일이름명:HW8-학번-이름)로 스마트캠퍼스에 제출하세요.