

## [숙제 9] lab5 실습 내용

### (1) 실습 A 관련(LAB5 P20)

소스코드(matrix.s) 18번 라인에 break point를 주고, r6(s) = 1, r7(u) = 0, r8(v)=2가 되는 시점부터 trace 하세요. lab5 자료 P20 아래 그림상에서 1번값과 2번값이 곱해져 3번에 저장되는 과정에서, 1번 ~ 3번에 해당하는 메모리 번지 값(R11), 메모리 내용 값(1번 및 2번 읽어온 값, 3번에 저장한 값)을 캡처하 시오. 캡처결과가 어느 것에 해당하는 지 표시하세요.

```

1  .text
2  .global matrix
3
4  matrix:
5      stmfid sp!, {r8-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     ldr     r12,[r11]
19
20
21     mov     r11, #12
22     mul     r11,r7,r11
23     add     r11,r11,r8,LSL #2
24     add     r11,r11,r1
25
Continuing.
Breakpoint 3, Loop () at matrix.s:19
19  ldr     r12,[r11]
c
c
Continuing.
Breakpoint 3, Loop () at matrix.s:19
19  ldr     r12,[r11]
(gdb) enter gdb command. To interrupt inferior, send SIGINT.
  
```

name	value (hex)	value (decimal)	description
r0	0xf6ffefb0	4143968176	
r1	0xf6ffefd4	4143968212	
r2	0xf6ffeff0	4143968248	
r3	0x3	3	
r4	0xf6ffef038	4143968312	
r5	0x11148	69960	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x97f6c	622444	register 10 (64-bit)
r11	0xf6ffefbc	4143968188	register 11 (64-bit)
r12	0x0	0	register 12 (64-bit)
sp	0xf6ffef60	4143968104	

r6(s) = 1, r7(u) = 0, r8(v)=2가 되는 시점

address	hex	0xf6ffefd4	01 00 00 00	0xf6ffefec	03 00 00 00
more		0xf6ffefd8	01 00 00 00	0xf6ffeff0	03 00 00 00
0xf6ffefb0	01 00 00 00	0xf6ffefdc	01 00 00 00	0xf6ffeff4	03 00 00 00
0xf6ffefb4	02 00 00 00	0xf6ffefe0	02 00 00 00	0xf6ffeff8	0e 00 00 00
0xf6ffefb8	03 00 00 00	0xf6ffefe4	02 00 00 00	0xf6ffeffc	0e 00 00 00
0xf6ffefbc	01 00 00 00	0xf6ffefe8	02 00 00 00	0xf6fff000	0e 00 00 00
0xf6ffefc0	02 00 00 00	0xf6ffefec	03 00 00 00	0xf6fff004	0e 00 00 00
0xf6ffefc4	03 00 00 00	0xf6ffeff0	03 00 00 00	0xf6fff008	0e 00 00 00
0xf6ffefc8	01 00 00 00	0xf6ffeff4	03 00 00 00	0xf6fff00c	48 11 01 00
0xf6ffefcc	02 00 00 00			0xf6fff010	00 00 00 00
0xf6ffefd0	03 00 00 00			0xf6fff014	00 00 00 00

왼쪽부터 배열 A, B, C 1번 메모리 주소: 0xf6ffefbc 2번 메모리 주소: 0xf6ffefdc 3번 메모리 주소: 0xf6fff00c

Load Binary /path/to/target/nevercutable -and -flag

show filesystem fetch disassembly reload file jump to line /home/jongssoo/programming/CS/lab5/matrix.s:21 (62 lines total)

```

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     mov     r11,#12
37     mul     r11,r6,r11
38     add     r11,r11,r8,LSL #2
39
40

```

19 ldr r12,[r11]  
c  
c  
(gdb) enter gdb command. To interrupt inferior, send SIGINT.

signals

registers

name	value (hex)	value (decimal)	description
r0	0xf6ffefb0	4143968176	
r1	0xf6ffefd4	4143968212	
r2	0xf6ffeff8	4143968248	
r3	0x3	3	
r4	0xf6fff038	4143968312	
r5	0x11148	69960	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x97f6c	622444	register 10 (64-bit)
r11	0xf6ffefbc	4143968188	register 11 (64-bit)
r12	0x1	1	register 12 (64-bit)
sp	0xf6ffef68	4143968104	
lr	0x1088c	67724	
pc	0x10990	67984	
cpsr			
d0			VFP double-precision (Temporary register)

r12에 1번에서 읽어온 값 10이 저장되었다. r11을 보면 1번의 메모리 주소 확인이 가능하다

Load Binary /path/to/target/nevercutable -and -flag

show filesystem fetch disassembly reload file jump to line /home/jongssoo/programming/CS/lab5/matrix\_Original.s:26 (62 lines total)

```

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     mov     r11,#12
37     mul     r11,r6,r11
38     add     r11,r11,r8,LSL #2
39
40
41     add     r11,r11,r2
42

```

19 ldr r12,[r11]  
c  
c  
(gdb) enter gdb command. To interrupt inferior, send SIGINT.

registers

name	value (hex)	value (decimal)	description
r0	0xf6ffefb0	4143968176	
r1	0xf6ffefd4	4143968212	
r2	0xf6ffeff8	4143968248	
r3	0x3	3	
r4	0xf6fff038	4143968312	
r5	0x11148	69960	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x97f6c	622444	register 10 (64-bit)
r11	0xf6ffefdc	4143968220	register 11 (64-bit)
r12	0x1	1	register 12 (64-bit)
sp	0xf6ffef68	4143968104	
lr	0x1088c	67724	
pc	0x109a0	68000	
cpsr			
d0			VFP double-precision (Temporary register)

r11을 보면 2번의 메모리 주소 확인이 가능하다.

Load Binary /path/to/target/wwoutable -and -Eflag

show filesystem fetch disassembly reload file jump to line /home/jongssoo/programming/CS/lab5/matrix.s:28 (62 lines total)

```

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

```

Breakpoint 4, Loop () at matrix.s:28  
28 mul r12,r11,r12  
(gdb) enter gdb command. To interrupt inferior, send SIGINT.

signals  
registers

name	value (hex)	value (decimal)	description
r0	0xf6fffeb0	4143968176	
r1	0xf6fffed4	4143968212	
r2	0xf6fffeff8	4143968248	
r3	0x3	3	
r4	0xf6fff038	4143968312	
r5	0x11148	69960	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x97f6c	622444	register 10 (64-bit)
r11	0x1	1	register 11 (64-bit)
r12	0x1	1	register 12 (64-bit)
sp	0xf6fffe68	4143968104	
lr	0x1088c	67724	
pc	0x109a4	68004	
cpsr			
d0			VFP double-precision (Temporary register)

r11에 2번에서 읽어온 값 10이 저장되었다.

Load Binary /path/to/target/wwoutable -and -Eflag

show filesystem fetch disassembly reload file jump to line /home/jongssoo/programming/CS/lab5/matrix\_Original.s:43 (62 lines total)

```

21     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
42
43     str     r9,[r11]
44
45     mov     r9,#0
46
47     add     r8,r8,#1
48     cmp     r8,r3
49     bne     Loop
50
51     mov     r8,#0
52
53     add     r6,r6,#1

```

Breakpoint 4, Loop () at matrix\_Original.s:43  
43 str r9,[r11]  
(gdb) enter gdb command. To interrupt inferior, send SIGINT.

registers

name	value (hex)	value (decimal)	description
r0	0xf6fffeb0	4143968176	
r1	0xf6fffed4	4143968212	
r2	0xf6fffeff8	4143968248	
r3	0x3	3	
r4	0xf6fff038	4143968312	
r5	0x11148	69960	
r6	0x1	1	
r7	0x0	0	
r8	0x2	2	register 8 (64-bit)
r9	0xe	14	register 9 (64-bit)
r10	0x97f6c	622444	register 10 (64-bit)
r11	0xf6fff00c	4143968268	register 11 (64-bit)
r12	0x9	9	register 12 (64-bit)
sp	0xf6fffe68	4143968104	
lr	0x1088c	67724	
pc	0x109cc	68044	
cpsr			
d0			VFP double-precision (Temporary register)

r11을 보면 3번의 메모리 주소 확인이 가능하다.

0xf6fffeff8	0e 00 00 00
0xf6fffeffc	0e 00 00 00
0xf6fff000	0e 00 00 00
0xf6fff004	0e 00 00 00
0xf6fff008	0e 00 00 00
0xf6fff00c	0e 00 00 00
0xf6fff010	00 00 00 00
0xf6fff014	00 00 00 00

3번 메모리 주소: 0xf6fff00c 에 모든 루프가 끝나고 14가 저장되었다.

## (2) 실습 B 관련(LAB5 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 입력받아 각각 초기화합니다. 매트릭스 인덱스  $s, u, v$  값이 저장된 3 x 1 array 1개  $D$  를 선언합니다. C 프로그램에서 Matrix  $A, B, C$  에 대한 pointer값 3 개,  $D$  의 시작번지를 argument 로 하여 어셈블리 함수를 호출합니다. 어셈블리 코드에서는 matrix 곱셈을 수행하여 결과를 Matrix  $C$  에 저장합니다. (Lab5 자료 P22, 23, 24, 27 내용대로 구현하세요).

Test를 아래와 같이 수행하여 결과를 제출하세요.

- $s, u, v$  값을 전부 3 으로 입력하고 Matrix 배열 메모리를 할당받고 위와 같이 초기화 한후 Matrix 곱셈 수행후 결과 출력 후 캡처 첨부
  - $s=3, u=4, v=5$  로 입력하고 Matrix 배열 메모리를 할당받고 위와 같이 초기화 한후 Matrix 곱셈 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.

```
jongsoo@ubuntu: ~/programming/CS/lab5
jongsoo@ubuntu:~/programming/CS/lab5$ ./lab5
Input values of s, u, v: 345
This is 3 by 5 matrix version

Input value of A : 1 2 3 4 5 6 7 8 9 10 11 12
Input value of B : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Array A
1 2 3 4
5 6 7 8
9 10 11 12
Array B
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
Array C after Operating :
110 120 130 140 150
246 272 298 324 350
382 424 466 508 550
```

$s = 3, u = 4, v = 5$

```
jongsoo@ubuntu: ~/programming/CS/lab5
jongsoo@ubuntu:~/programming/CS/lab5$ ./lab5
Input values of s, u, v: 333
This is 3 by 3 matrix version

Input value of A : 1 2 3 4 5 6 7 8 9
Input value of B : 1 2 3 4 5 6 7 8 9

Array A
1 2 3
4 5 6
7 8 9
Array B
1 2 3
4 5 6
7 8 9
Array C after Operating :
30 36 42
66 81 96
102 126 150
jongsoo@ubuntu:~/programming/CS/lab5$
```

$s = 3, u = 3, v = 3$

보고서와 소스코드를 2020-2-ca-hw@q.ssu.ac.kr로 제출하시오.

제출마감: 11월 18일(수) 23시59분