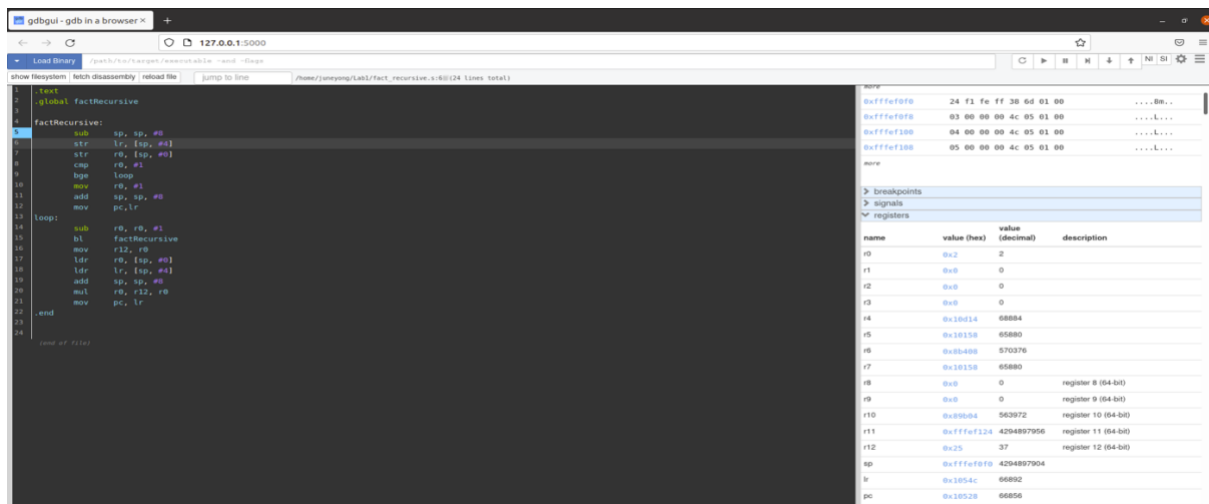
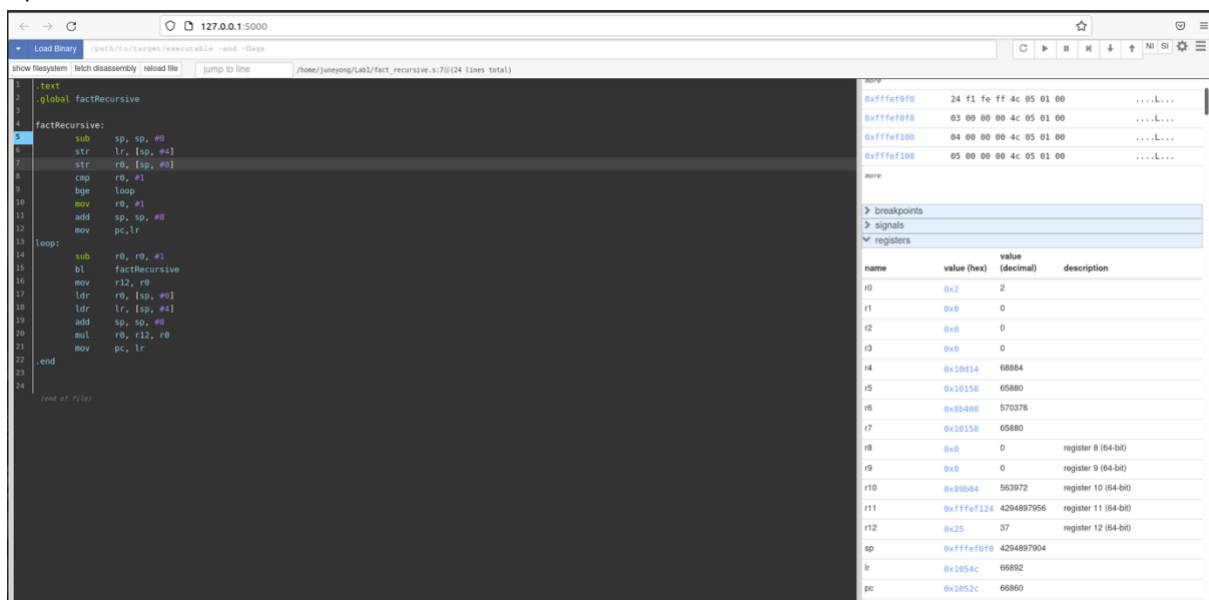


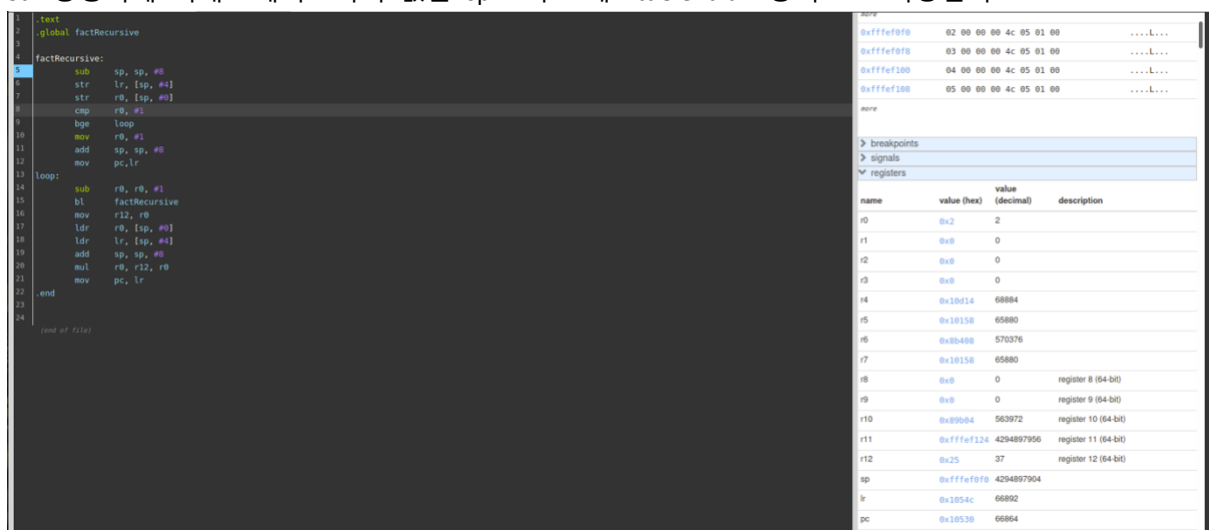
실습 a



Sub 명령어에 의해 $sp = sp - 8$ 이다. sp의 주소값이 8비트 작아진다. 이때 sp의 주소값은 0xfffff0f0이다



Str 명령어에 의해 lr레지스터의 값을 sp+4 주소에 little endian 방식으로 저장한다.



Str 명령어에 의해 r0레지스터의 값을 sp 주소에 little endian 방식으로 저장한다.

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub sp, sp, #8
6     str lr, [sp, #4]
7     str r0, [sp, #0]
8     cmp r0, #1
9     bge loop
10    mov r0, #1
11    add sp, sp, #8
12    mov pc, lr
13 loop:
14     sub r0, r0, #1
15     bl factRecursive
16     mov r12, r0
17     ldr r0, [sp, #0]
18     ldr lr, [sp, #4]
19     add sp, sp, #8
20     mul r0, r12, r0
21     mov pc, lr
22 .end
23
24 (end of file)

```

none

0xffffef0 02 00 00 00 4c 05 01 00 ...

0xffffef0 03 00 00 00 4c 05 01 00 ...

0xffffef0 04 00 00 00 4c 05 01 00 ...

0xffffef0 05 00 00 00 4c 05 01 00 ...

none

> breakpoints

> signals

✓ registers

name	value (hex)	value (decimal)	description
r0	0x1	1	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10158	65880	
r6	0xb400	570376	
r7	0x10158	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x90b04	563972	register 10 (64-bit)
r11	0xfffff124	4294897956	register 11 (64-bit)
r12	0x25	37	register 12 (64-bit)
sp	0xffffef0	4294897904	
lr	0x1054c	66892	
pc	0x10540	66888	

Sub 명령어에 의해 $r0 = r0 - 1$ 이다.

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub sp, sp, #8
6     str lr, [sp, #4]
7     str r0, [sp, #0]
8     cmp r0, #1
9     bge loop
10    mov r0, #1
11    add sp, sp, #8
12    mov pc, lr
13 loop:
14     sub r0, r0, #1
15     bl factRecursive
16     mov r12, r0
17     ldr r0, [sp, #0]
18     ldr lr, [sp, #4]
19     add sp, sp, #8
20     mul r0, r12, r0
21     mov pc, lr
22 .end
23
24 (end of file)

```

none

0xffffef0 02 00 00 00 4c 05 01 00 ...

0xffffef0 03 00 00 00 4c 05 01 00 ...

0xffffef0 04 00 00 00 4c 05 01 00 ...

0xffffef0 05 00 00 00 4c 05 01 00 ...

none

> breakpoints

> signals

✓ registers

name	value (hex)	value (decimal)	description
r0	0x1	1	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10158	65880	
r6	0xb400	570376	
r7	0x10158	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x90b04	563972	register 10 (64-bit)
r11	0xfffff124	4294897956	register 11 (64-bit)
r12	0x25	37	register 12 (64-bit)
sp	0xffffef0	4294897904	
lr	0x1054c	66892	
pc	0x10520	66856	

Sub 명령어에 의해 $sp = sp - 8$ 이다. sp 의 주소값이 8비트 작아진다. 이때 sp 의 주소값은 0xffff0e8이다

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub sp, sp, #8
6     str lr, [sp, #4]
7     str r0, [sp, #0]
8     cmp r0, #1
9     bge loop
10    mov r0, #1
11    add sp, sp, #8
12    mov pc, lr
13 loop:
14     sub r0, r0, #1
15     bl factRecursive
16     mov r12, r0
17     ldr r0, [sp, #0]
18     ldr lr, [sp, #4]
19     add sp, sp, #8
20     mul r0, r12, r0
21     mov pc, lr
22 .end
23
24 (end of file)

```

none

0xffff0e8 00 00 00 00 4c 05 01 00 ...

0xffffef0 02 00 00 00 4c 05 01 00 ...

0xffffef0 03 00 00 00 4c 05 01 00 ...

0xffffef0 04 00 00 00 4c 05 01 00 ...

none

> breakpoints

> signals

✓ registers

name	value (hex)	value (decimal)	description
r0	0x1	1	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10158	65880	
r6	0xb400	570376	
r7	0x10158	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x90b04	563972	register 10 (64-bit)
r11	0xfffff124	4294897956	register 11 (64-bit)
r12	0x25	37	register 12 (64-bit)
sp	0xffff0e8	4294897896	
lr	0x1054c	66892	
pc	0x1052c	66860	

Str 명령어에 의해 lr레지스터의 값을 $sp+4$ 주소에 little endian 방식으로 저장한다.


```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub    sp, sp, #8
6     str    lr, [sp, #4]
7     str    r0, [sp, #0]
8     cmp    r0, #1
9     bge    loop
10    mov    r0, #1
11    add    sp, sp, #8
12    mov    pc, lr
13
14 loop:
15    sub    r0, r0, #1
16    bl     factRecursive
17    mov    r12, r0
18    ldr    r0, [sp, #0]
19    ldr    lr, [sp, #4]
20    add    sp, sp, #8
21    mul    r0, r12, r0
22    mov    pc, lr
23
24 .end

```

```

mov     0xffff0eb0  58 01 01 00 4c 05 01 00  ....X...
0xffff0eb0  01 00 00 00 4c 05 01 00  ....L...
0xffff0ef0  02 00 00 00 4c 05 01 00  ....L...
0xffff0ff0  03 00 00 00 4c 05 01 00  ....L...

```

name	value (hex)	value (decimal)	description
r0	0x0	0	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10150	65880	
r6	0x0b400	570376	
r7	0x10150	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x09064	563972	register 10 (64-bit)
r11	0xffff124	4294897956	register 11 (64-bit)
r12	0x25	37	register 12 (64-bit)
sp	0xffff0eb0	4294897888	
lr	0x1054c	66892	
pc	0x1052c	66880	

Str 명령어에 의해 lr레지스터의 값을 sp+4 주소에 little endian 방식으로 저장한다.

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub    sp, sp, #8
6     str    lr, [sp, #4]
7     str    r0, [sp, #0]
8     cmp    r0, #1
9     bge    loop
10    mov    r0, #1
11    add    sp, sp, #8
12    mov    pc, lr
13
14 loop:
15    sub    r0, r0, #1
16    bl     factRecursive
17    mov    r12, r0
18    ldr    r0, [sp, #0]
19    ldr    lr, [sp, #4]
20    add    sp, sp, #8
21    mul    r0, r12, r0
22    mov    pc, lr
23
24 .end

```

```

mov     0xffff0eb0  00 00 00 00 4c 05 01 00  ....L...
0xffff0eb0  01 00 00 00 4c 05 01 00  ....L...
0xffff0ef0  02 00 00 00 4c 05 01 00  ....L...
0xffff0ff0  03 00 00 00 4c 05 01 00  ....L...

```

name	value (hex)	value (decimal)	description
r0	0x0	0	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10150	65880	
r6	0x0b400	570376	
r7	0x10150	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x09064	563972	register 10 (64-bit)
r11	0xffff124	4294897956	register 11 (64-bit)
r12	0x25	37	register 12 (64-bit)
sp	0xffff0eb0	4294897888	
lr	0x1054c	66892	
pc	0x10530	66864	

Str 명령어에 의해 r0레지스터의 값을 sp 주소에 little endian 방식으로 저장한다.

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub    sp, sp, #8
6     str    lr, [sp, #4]
7     str    r0, [sp, #0]
8     cmp    r0, #1
9     bge    loop
10    mov    r0, #1
11    add    sp, sp, #8
12    mov    pc, lr
13
14 loop:
15    sub    r0, r0, #1
16    bl     factRecursive
17    mov    r12, r0
18    ldr    r0, [sp, #0]
19    ldr    lr, [sp, #4]
20    add    sp, sp, #8
21    mul    r0, r12, r0
22    mov    pc, lr
23
24 .end

```

```

mov     0xffff0eb0  00 00 00 00 4c 05 01 00  ....L...
0xffff0eb0  01 00 00 00 4c 05 01 00  ....L...
0xffff0ef0  02 00 00 00 4c 05 01 00  ....L...
0xffff0ff0  03 00 00 00 4c 05 01 00  ....L...

```

name	value (hex)	value (decimal)	description
r0	0x1	1	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10150	65880	
r6	0x0b400	570376	
r7	0x10150	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x09064	563972	register 10 (64-bit)
r11	0xffff124	4294897956	register 11 (64-bit)
r12	0x25	37	register 12 (64-bit)
sp	0xffff0eb0	4294897888	
lr	0x1054c	66892	
pc	0x10540	66880	

Mov 명령어에 의해 1의 값을 r0로 복사한다. 그리고 add 명령어에 의해 $sp = sp + 8$ 이 되어 sp의 주소를 8비트 상위 주소로 바꾼다. 바꾸게 되면 0xffff0e8이 된다.

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub    sp, sp, #8
6     str    lr, [sp, #4]
7     str    r0, [sp, #0]
8     cmp    r0, #1
9     bge    loop
10    mov    r0, #1
11    add    sp, sp, #8
12    mov    pc, lr
13
14 loop:
15    sub    r0, r0, #1
16    bl     factRecursive
17    mov    r12, r0
18    ldr    r0, [sp, #0]
19    ldr    lr, [sp, #4]
20    add    sp, sp, #8
21    mul    r0, r12, r0
22    mov    pc, lr
23
24 .end

```

name	value (hex)	value (decimal)	description
r0	0x1	1	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10158	65880	
r6	0x0b400	570376	
r7	0x10158	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x090b4	563972	register 10 (64-bit)
r11	0xfffff124	4294897956	register 11 (64-bit)
r12	0x1	1	register 12 (64-bit)
sp	0xfffff0f0	4294897904	
lr	0x1054c	66892	
pc	0x1055c	66908	

Mov 명령어에 의해 pc 레지스터에 lr의 값을 복사하고 r12에 r0의 값을 복사한다. 그리고 ldr 명령어로 sp에서 4비트 값을 r0에 복사하고 sp+4에서 4비트 값을 lr에 복사한다.

```

1 .text
2 .global factRecursive
3
4 factRecursive:
5     sub    sp, sp, #8
6     str    lr, [sp, #4]
7     str    r0, [sp, #0]
8     cmp    r0, #1
9     bge    loop
10    mov    r0, #1
11    add    sp, sp, #8
12    mov    pc, lr
13
14 loop:
15    sub    r0, r0, #1
16    bl     factRecursive
17    mov    r12, r0
18    ldr    r0, [sp, #0]
19    ldr    lr, [sp, #4]
20    add    sp, sp, #8
21    mul    r0, r12, r0
22    mov    pc, lr
23
24 .end

```

name	value (hex)	value (decimal)	description
r0	0x1	1	
r1	0x0	0	
r2	0x0	0	
r3	0x0	0	
r4	0x10d14	68884	
r5	0x10158	65880	
r6	0x0b400	570376	
r7	0x10158	65880	
r8	0x0	0	register 8 (64-bit)
r9	0x0	0	register 9 (64-bit)
r10	0x090b4	563972	register 10 (64-bit)
r11	0xfffff124	4294897956	register 11 (64-bit)
r12	0x1	1	register 12 (64-bit)
sp	0xfffff0f0	4294897904	
lr	0x1054c	66892	
pc	0x1054c	66892	

Add 명령어를 통해 $sp = sp + 8$ 이 되어 sp의 주소가 0xffff0f0이 된다. Mul 명령어에 의해 $r0 = r0 * r12$ 이 되어 r0에 저장된다.

실습B

```
info reg cpsr
cpsr      0x20000010      536870928
```

R0의 값이 1보다 클 경우 cpsr의 값이 0x20000010로 $n=0, z=0, c=1, v=0$ 이다. 이 경우 8번, 9번 실행 후 14번으로 이동한다.

```
info reg cpsr
cpsr      0x60000010      1610612752
```

R0의 값이 1인 경우 cpsr의 값이 0x60000010로 $n=0, z=1, c=1, v=0$ 이다. 이 경우 8번, 9번 실행 후 14번으로 이동하였다.

```
info reg cpsr
cpsr      0x80000010      -2147483632
```

R0의 값이 0인 경우 cpsr의 값이 0x80000010로 $n=1, z=0, c=0, v=0$ 이다. 이 경우 8번, 9번 실행 후 10번으로 이동하였다.