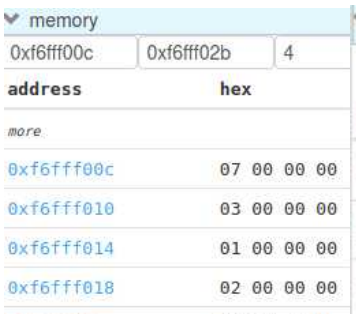
[숙제 7] lab3 실습 내용

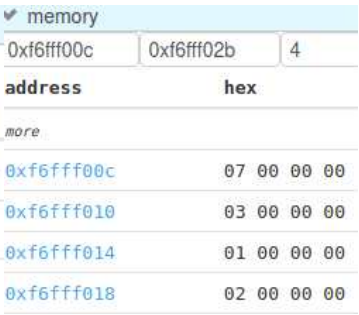
제출할 자료:

(1) 실습 A 관련(LAB3 P23-24)
2020-2-CA-assembly Lab3 p23-24를 참고하여, bubblesort를 수행하면서 outer loop 수행이 끝날 때마다 배열 값의 변동을 확인하세요. (14번 줄, 43번 줄에 breakpoint를 설정하고 p23에 있는 그림처럼 수행되는지 확인하고 그 과정을 캡처하여 보고서에 첨부하세요.)



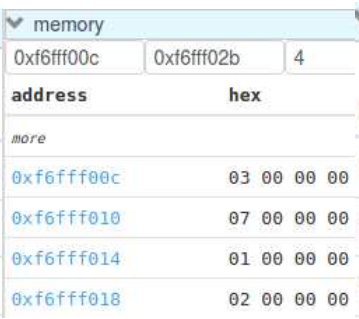
〈Initial mem〉

〈Pass 1 reg〉



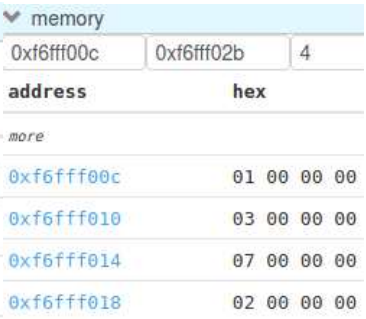
〈Pass 1 mem〉

〈Pass 2 reg〉



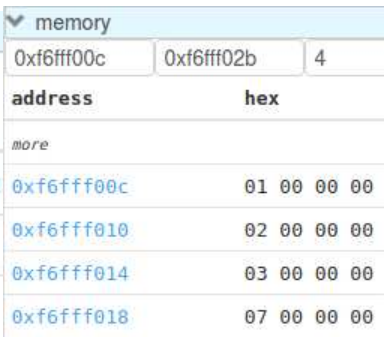
〈Pass 2 mem〉

〈Pass 3 reg〉



〈Pass 3 mem〉

〈Pass 4 reg〉



〈Pass 4 mem〉

(2) 실습 B 관련(LAB3 P25-27)
bubblesort를 수행하면서 i가 2일 때, inner loop를 수행하면서 배열 값의 변동을 확인하세요.
(14번 줄, 43번 줄에 breakpoint를 설정하고 r2가 2일 때까지 계속 수행합니다. r2가 2가 되고 outer loop(for1tst)를 수행하기 전(43번 줄), 35번 줄, 36번 줄, 38번 줄에 breakpoint를 설정하고 현재 arr에 저장된 값을 확인합니다. 35번 줄에서의 r3, arr에 저장된 값과 36번 줄에서의 r3, arr에 저장된 값을 비교합니다. 43번 줄에서 멈추면 r2, arr에 저장된 값을 p4에 있는 pass i 결과와 비교합니다. 위 과정에 대한 캡처화면을 설명과 함께 보고서에 첨부하세요.)

memory

0xf6fff00c	0xf6fff02b	4
address	hex	
more		
0xf6fff00c	07 00 00 00	
0xf6fff010	03 00 00 00	
0xf6fff014	01 00 00 00	
0xf6fff018	02 00 00 00	

registers

name	value (hex)
r0	0xf6fff00c
r1	0x4
r2	0x2

〈Initial mem〉 〈Initial reg〉

memory

0xf6fff00c	0xf6fff02b	4
address	hex	
more		
0xf6fff00c	03 00 00 00	
0xf6fff010	07 00 00 00	
0xf6fff014	01 00 00 00	
0xf6fff018	02 00 00 00	

registers

name	value (hex)
r0	0xf6fff00c
r1	0x1
r2	0x2
r3	0x1

〈Pass 3-1 mem 35번줄〉 〈Pass 3-1 reg〉

3-1 35번 줄: R3 (변수 j에 해당) = 1 Arr에 저장된 값 : 3, 7, 1, 2

memory

0xf6fff00c	0xf6fff02b	4
address	hex	
more		
0xf6fff00c	03 00 00 00	
0xf6fff010	07 00 00 00	
0xf6fff014	01 00 00 00	
0xf6fff018	02 00 00 00	

registers

name	value (hex)
r0	0xf6fff00c
r1	0x1
r2	0x7
r3	0x1

〈Pass 3-1 mem 36번줄〉 〈Pass 3-1 reg〉

memory

0xf6fff00c	0xf6fff02b	4
address	hex	
more		
0xf6fff00c	03 00 00 00	
0xf6fff010	01 00 00 00	
0xf6fff014	07 00 00 00	
0xf6fff018	02 00 00 00	

registers

name	value (hex)
r0	0xf6fff00c
r1	0x0
r2	0x2
r3	0x0

〈Pass 3-2 mem 35번줄〉 〈Pass 3-2 reg〉

3-1 36번 줄: R3 = 1 Arr에 저장된 값 : 3, 1, 7, 2 (7과 1 swap)

3-2 35번 줄: R3 (변수 j에 해당) = 0 Arr에 저장된 값 : 3, 1, 7, 2

memory

0xf6fff00c	0xf6fff02b	4
address	hex	
more		
0xf6fff00c	03 00 00 00	
0xf6fff010	01 00 00 00	
0xf6fff014	07 00 00 00	
0xf6fff018	02 00 00 00	

registers

name	value (hex)
r0	0xf6fff00c
r1	0x0
r2	0x3
r3	0x1

〈Pass 3-2 mem 36번줄〉 〈Pass 3-2 reg〉

memory

0xf6fff00c	0xf6fff02b	4
address	hex	
more		
0xf6fff00c	01 00 00 00	
0xf6fff010	03 00 00 00	
0xf6fff014	07 00 00 00	
0xf6fff018	02 00 00 00	

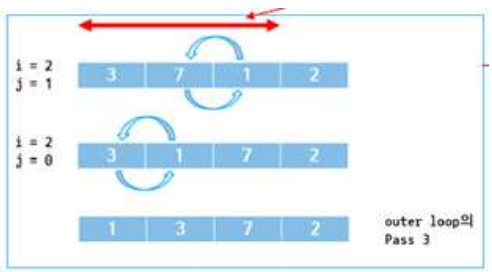
registers

name	value (hex)
r0	0xf6fff00c
r1	0x4
r2	0x3

〈Pass 3-3 mem 43번줄〉 〈Pass 3-3 reg〉

3-2 36번줄: R3 = 1 Arr에 저장된 값 : 1, 3, 7, 2 (3과 1 swap)

3-3 43번줄: R2 = 3 Arr에 저장된 값 : 1, 3, 7, 2



〈outer loop〉

outerloop의 pass 3 arr에 저장된 값: 1, 3, 7, 2

메모리에 저장된 값: 1, 3, 7, 2

(3) 실습 C 관련(LAB3 P28-29)

swap 함수 수행 전후 레지스터 저장 및 복구 루틴을 삭제한 후, swap 함수 수행 전후 i, j값을 확인하세요. (bubblesort 어셈블리 코드 32번 줄, 36번 줄을 주석 처리하고 재실행합니다. 15번 줄, 43번 줄에 breakpoint를 설정하고 arr에 저장된 값들을 확인합니다. 43번 줄에서 멈추면 현재 r2값을 확인하고, 만약 1이면 35번 줄, 38번 줄에 breakpoint를 설정합니다. 35번 줄과 38번 줄에서 r3값과 arr에 저장된 값을 각각 확인하고 비교합니다. 35번 줄에서의 r2, r3값과 38번 줄에서의 r2, r3값이 동일한지 확인하고 동일하지 않다면 이유를 분석한다. 위 과정에 대한 캡처화면을 설명과 함께 보고서에 첨부하세요.)

맨 처음 15번째 줄 멈추었을 때 arr에 저장된 값: 7, 3, 1, 2

맨 처음 43번째 줄 멈추었을 때 r2에 저장된 값: 1 arr에 저장된 값: 7, 3, 1, 2

registers

name	value (hex)
r0	0xf6fff00c
r1	0x0
r2	0x1
r3	0x0

memory

0xf6fff00c 0xf6fff02b 4

address	hex
more	
0xf6fff00c	07 00 00 00
0xf6fff010	03 00 00 00
0xf6fff014	01 00 00 00
0xf6fff018	02 00 00 00

〈35번 줄 reg〉 〈35번 줄 mem〉

registers

name	value (hex)
r0	0xf6fff00c
r1	0x0
r2	0x7
r3	0x3

memory

0xf6fff00c 0xf6fff02b 4

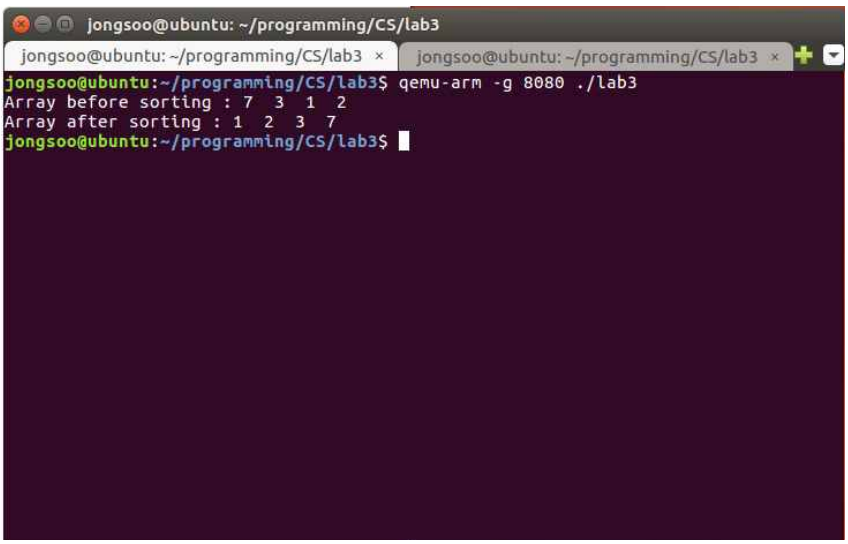
address	hex
more	
0xf6fff00c	03 00 00 00
0xf6fff010	07 00 00 00
0xf6fff014	01 00 00 00
0xf6fff018	02 00 00 00

〈38번 줄 reg〉 〈38번 줄 mem〉

값이 동일하지 않은 이유는 swap 함수 call 전후 레지스터 저장 및 복구 루틴을 삭제했기 때문이다. Swap 함수에서 r2와 r3을 사용한다. caller 입장에서 r2, r3를 스택에 저장하지 않아, swap call 이후 r2, r3의 값이 깨지게 되었다. 7과 3을 swap했기 때문에 r2, r3에 7, 3이 저장된다.

(4) 실습 D 관련(LAB3 P30)

p8에 있는 어셈블리 코드는 bubble sorting 방법 2로 구현되었습니다. 이를 bubble sorting 방법 1(p3)로 동작하도록 수정합니다. (만일 정상적으로 동작하지 않는다면 p3에 있는 예제에서 outer loop의 pass i에 해당된 결과가 나왔는지 단계별로 확인해보세요. 실습 A, B에서 설정했던 breakpoint를 활용해 디버깅해보세요). 정상적으로 동작한다면 line by line으로 설명된 어셈블리 코드와 수행 결과를 캡처하여 보고서에 첨부하세요.



〈수행 결과〉

```
.text
.global bubblesort
.global swap
.type bubblesort, STT_FUNC

bubblesort:
                sub     sp, sp, #20
                str     lr, [sp, #16]
                str     r7, [sp, #12]
                str     r6, [sp, #8]
                str     r3, [sp, #4]
                str     r2, [sp, #0]

                mov     r6, r0
                mov     r7, r1
                mov     r2, #0
                sub     r1, r1, #1

for1tst:
                cmp     r2, r1
                bge     exit1
#               sub     r3, r2, #1
                mov     r3, #0

for2tst:
                sub     r1, r1, r2
#               cmp     r3, #0
                cmp     r3, r1
                add     r1, r1, r2
#               blt     exit2
                bge     exit2
                add     r12, r0, r3, LSL #2
                ldr     r4, [r12, #0]
                ldr     r5, [r12, #4]
                cmp     r4, r5
                ble     exit2

                stmdb   sp!,{r0,r1,r2,r3,r12}
                mov     r0, r6
                mov     r1, r3
                bl      swap
                ldmia   sp!,{r0,r1,r2,r3,r12}

#               sub     r3, r3, #1
                add     r3, r3, #1
                b       for2tst

exit2:
                add     r2, r2, #1
                b       for1tst

exit1:
                ldr     r2, [sp, #0]
                ldr     r3, [sp, #4]
                ldr     r6, [sp, #8]
                ldr     r7, [sp, #12]
                ldr     lr, [sp, #16]
                add     sp, sp, #20

                mov     pc, lr

.end
```

〈어셈블리 코드〉

17번째 줄 sub r1, r1, #1
i와 n−1을 비교하므로 r1=r1−1을 함

23번째 줄 mov r3, #0
r3이 j에 해당된 값이므로 inner loop를 들어갈 때 0으로 초기화

26번째 줄 sub r1, r1, r2
inner loop에서 j와 n−1−i를 비교하므로 n−1상태인 r1에 r2를 뺌 r1=r1−r2

28번째 줄 cmp r3, r1
j와 n−1−i를 비교

29번째 줄 add r1, r1, r2
다시 r1=r1+r2를 해 r1을 원래대로 (n−1상태) 돌림

31번째 줄 bge exit2
R3이 r1보다 크거나 같으면 exit2로 점프 j 〉= n−1−i인 경우 상태비트를 확인해 판단하므로 line 29의 add는 영향이 없다.

45번째 줄 add r3, r3, #1
R3 = r3+1 / j++에 해당

결과들을 2020-2-ca-hw@q.ssu.ac.kr로 제출하시오.
제출마감: 11월 4일(수) 23시59분