

[숙제 10&11] lab6 실습 내용

lab6를 참고해 quick sort를 assembly language로 구현해보세요.

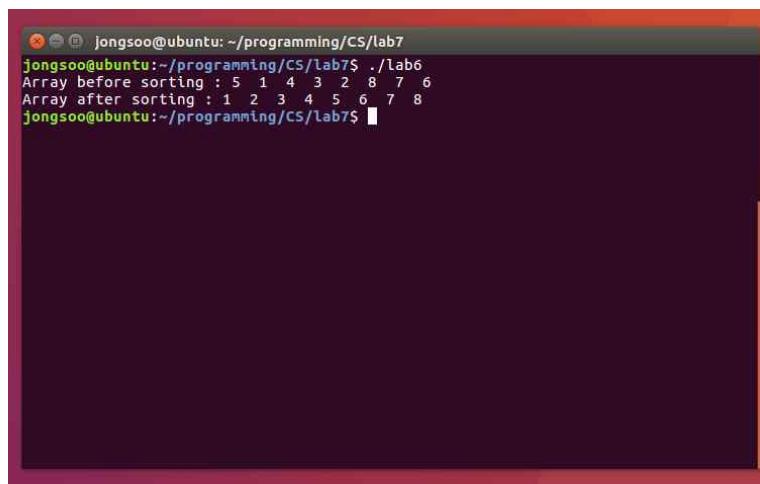
● 구현 시 주의사항

- Lab 3의 bubble sorting 코드를 활용
- 수정 시 주의사항

Recursive call 할 때에는 call 전 레지스터 값을 스택에 저장했다가 call 후 레지스터 값을 복구하는 과정이 필요

● 보고서에 포함할 내용

- 결과내용을 캡쳐



```
jongsoo@ubuntu:~/programming/CS/lab7$ ./lab6
Array before sorting : 5 1 4 3 2 8 7 6
Array after sorting : 1 2 3 4 5 6 7 8
jongsoo@ubuntu:~/programming/CS/lab7$
```

- 소스코드에 대한 주석/설명

```
asm quicksort.s
1 .text
2 .global quicksort
3 .global swap
4 .type quicksort, STT_FUNC
5
6 //r0, r1, r2 = array address, first, last
7 //r4, r5, r6, r7, r11, r12 = array address, i, j, pivot, temp1, temp2
8 //r8, r9, r10 = array[i], array[j], array[pivot]
9
10 quicksort:
11     stmdfd sp!, {r0-r12, lr} //save register state
12
13     cmp r1, r2 //if first < last, branch
14     blt content
15     b exit
16
17 content:
18     mov r4, r0 //save mem address
19     mov r5, r1 //initialize i
20     mov r6, r2 //initialize j
21     mov r7, r5 //initialize pivot
22
23 loop1:
24     cmp r5, r6 //if i < j, branch to loop2
25     ldrlt r10, [r4, r7, lsl #2] //load array[pivot] value
26     bge contentexit
27
28 loop2:
29     ldr r8, [r4, r5, lsl #2] //load array[i] value
30     cmp r8, r10 //branch if array[i] <= array[pivot]
31     bgt loop3
32     cmp r5, r2 //branch if i < last
33     bge loop3
34     add r5, r5, #1
35     b loop2
36
37     loop3:
38         ldr r9, [r4, r6, lsl #2] //load array[j] value
39         cmp r9, r10 //branch if array[j] <= array[pivot]
40         ble loopexit
41         sub r6, r6, #1
42         b loop3
43
44     loopexit:
45         cmp r5, r6
46         ldrlt r11, [r4, r5, lsl #2] //temp1 = array[i]
47         ldrlt r12, [r4, r6, lsl #2] //temp2 = array[j]
48
49         strlt r12, [r4, r5, lsl #2] //array[i] = temp2
50         strlt r11, [r4, r6, lsl #2] //array[j] = temp1
51
52         b loop1
53
54     contentexit:
55         ldr r11, [r4, r6, lsl #2] //temp1 = array[j]
56         ldr r12, [r4, r7, lsl #2] //temp2 = array[pivot]
57
58         str r12, [r4, r6, lsl #2] //array[j] = temp2
59         str r11, [r4, r7, lsl #2] //array[pivot] = temp1
60
61         //parameter 1 = memaddress, parameter 2 = first
62         mov r11, r2 //save last to temp1
63         sub r2, r6, #1 //parameter 3 = j-1
64         b1 quicksort
65         mov r2, r11 //restore last
66
67         //parameter 1 = memaddress, parameter 3 = last
68         mov r11, r1 //save first to temp1
69         add r1, r6, #1 //parameter 2 = j+1
70         b1 quicksort
71         mov r1, r11 //restore first
72
73     exit:
74         ldmfd sp!, {r0-r12, pc}
75
76 .end
```

10~15번 줄: 가장 먼저 원래의 레지스터 상태들을 스택에 푸시한다. 시작이 끝점보다 크다면 그냥 함수를 종료하고 같거나 작으면 계속 진행한다.

17~21번 줄: 변수 i, j, pivot에 대해서 레지스터를 각각 할당하고 값을 초기화한다.

23~26번 줄: 첫 번째 루프에서 i, j 값을 비교 후 작다면 계속 진행한다. array[pivot]값을 메모리에서 가져와 r10레지스터에 저장한다. 크다면 루프를 나온다.

28~35번 줄: 두 번째 루프에서 pivot보다 큰 값이 나올 때 까지 i를 증가시킨다. array[pivot] 보다 큰 array[i]값이 나오거나 i값이 last보다 커지면 다음 루프로 넘어간다. 그렇지 않으면 계속 반복한다.

37~42번 줄: 세 번째 루프에서 pivot보다 작거나 같은 값이 나올 때 까지 j를 증가시킨다. array[pivot] 보다 작거나 같은 array[j]값이 나오면 다음 루프로 넘어간다. 그렇지 않으면 계속 반복한다. pivot이 처음을 의미하기 때문에 따로 first와 비교하지 않는다.

44~52번 줄: 두 번째 세 번째 루프를 빠져나와 i, j값을 비교한다. i가 j보다 작은 경우에만 array[i]와 array[j]의 값을 바꾼다. loop1으로 분기해 반복문을 계속 진행한다.

54~71번 줄: 첫 번째 루프가 종료되고 array[pivot]값과 array[j]값을 바꾸어준다. 그리고 첫 번째 재귀를 위해 last값을 temp1에 저장해둔다. 재귀의 last변수로 j-1을 넘겨준다. 재귀 종료 후 다시 last를 원래대로 복구 시켜준다. 두 번째 재귀를 위해 first값을 temp1에 저장해둔다. 재귀의 first 변수로 j+1을 넘겨준다. 재귀 종료 후 다시 first를 원래대로 복구 시켜준다.

73~74번 줄: 함수가 종료되고 원래의 레지스터 상태들을 스택에서 팝한다.

보고서 제출 시 소스코드와 함께 하나의 압축 file로 만들어 이메일로 보내세요.

HW10&11에 해당됩니다.

2020-2-ca-hw@q.ssu.ac.kr로 제출하시오.

제출마감: 11월29일 23시59분