```
1: .data
2:  #sequence:  .word   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,              #answer is 8 9 4 1
0 11 5 2 6 7 3 1
3:  #sequence:  .word   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15   #answer is 8 9
 4 10 11 5 2 12 13 6 14 15 7 3 1
4:  #sequence:  .word   32, 4, 56, 11, 77, 89                   #answer is 11 77 4 89 56 3
2
5:  #sequence:  .word   32, 4, 56, 11, 77, 89, 67               #answer is 11 77 4 89 67 5
6 32
6:  #sequence:  .word   32, 4, 56,                      #answer is 4 56 32
7:  sequence:   .word   32, 7                           #answer is 7 32
8:
9:
10:     num_of_elems:   .word   2
11:
12:     space_str:  .asciiz " "
13: .text
14: main:
15:     li  $t1,    0
16:     lw  $t6,    num_of_elems($t1)
17:
18:     beqz    $t6,    exit_programm
19:     jal DF_order_print
20:
21: exit_programm:
22:     #exit (terminate execution)
23:     li  $v0,    10
24:     syscall
25:
26: DF_order_print:
27:     subu    $sp,    $sp,    8
28:
29:     #сохраним в stack указатель на место, из которого была вызвана DF_order_print
30:     #и индекс элемента, в котором мы сейчас находимся, мы делаем это при
31:     #каждом вызове DF_order_print = при каждом переходе к следующему элементу
32:     #так организуется стековый фрейм для рекурсии
33:     sw  $ra,    0($sp)
34:     sw  $t1,    4($sp)
35:
36: #Левый лист
37:     #i_ch_l = i_p*2 + 1
38:     mul $t1,    $t1,    2
39:     add $t1,    $t1,    1
40:
41:     #Проверка на существоввание ребенка. В первом проходе она заводит
42:     #нас к листьям дерева. Однако факт перезаписи подгружаемого
43:     # из стека значения t1 и адреса перехода $ra, их подгрузка
44:     #Позволяют просто переходить от родителя к ребенку, а не в конец дерева.
45:     bge $t1,    $t6,    print_element_from_stack
46:     jal DF_order_print          #делая jal, мы перезаписываем $ra
47:     #$ra — содержит адрес инструкции, из которого была вызвана функция.
```

```
48:
49: #Правый лист
50:     #Проходя по правой ветке мы подгружаем значение записанное в стековом фрейме
51:     #Таким образом мы продолжаем рекурсивное движение от рассматриваемого родителя
52:     #В сторону не рассмотренного ребенка.
53:     lw  $t1,    4($sp)
54:
55:     #i_ch_r = i_p*2 + 2
56:     mul $t1,    $t1,    2
57:     add $t1,    $t1,    2
58:
59:     bge $t1,    $t6,    print_element_from_stack
60:     jal DF_order_print      #делая jal, мы перезаписываем $ra
61:     #$ra — содержит адрес инструкции, из которого была вызвана функция.
62:
63:     #После прохода по правому листу из стекового фрейма вытаскивается значение
64:     #которое переводит нас к позиции переде меткой print_element_from_stack.
65:     #Так организован переход от выведенного ребенка
66:     #К родителю с дальнейшим выводом и его.
67:
68:
69: print_element_from_stack:
70:
71:     lw  $t1,    4($sp)
72:     mul $t3,    $t1,    4
73:
74:     lw  $a0,    sequence($t3)
75:     #Печать элемента лежащего в a0
76:     li  $v0,    1
77:     syscall
78:
79:     #Печать символа пробел
80:     la  $a0,    space_str
81:     li  $v0,    4
82:     syscall
83:
84:     #Подгружаем записанное в stack значение $ra, чтобы продолжить исполнение
85:     lw  $ra,    ($sp)       #кода с того места, откуда были вызваны.
86:                     # на данном шаге рекурсии. Так, например, после
87:             #рассмотрнения левого ребенка мы перехоим в рассмотрение правого,
88:             #из рассмотренного правого - в метку print_element_from_stack
89:             #где производится сдвиг стекового фрейма - переход к родителю
90:
91:     #Здесь и работает рекурсия
92:     add $sp,    $sp,    8   #По результату выполнения этой строчки происходит
93:                     #сдвиг $sp - указателя на стековый фрейм, который
94:                     #определяет в какой участок кода мы попадем
95:                     # и с каким значением работаем.
96:
97:     #Далее происходит перезапись в DF_order_print стекового фрейма с тем, чтобы
98:     #продолжить рекурсивное движение по нашему дереву.
```

```
99:
100:    #Поднявшись до самой верхушки рекурсивной лестницы мы попадем в начало стека,
101:    #где, пройдем по $ra,и окажемся перед меткой первого вызова DF_order_print
102:    # - exit_programm.
103:
104:    #Переход из ранее подгруженного элемента
105:    jr  $ra
```