```
1: .data
2:  #Еазвания Функций был немного изменены, но заверяю вас, что
3:  #Рекурсия организована так же, а остальные отличия незначительны.
4:
5:  #sequence:  .word   '+', 1, 2               #answer is 1 + 2 = 3 (Checked!)
6:
7:  #sequence:  .word   '-', '*', 6, 1, 7           #answer is (1*7) - 6 = 1 (Checked!)
8:
9:  #sequence:  .word   '+', '*', '*', 11, 77, 1, 56        #answer is (11*77) + (56*1) =
903 (Checked!)
10:
11:     #sequence:  .word   '-', '*', '*', 11, 77, 1, 56        #answer is (11*77) - (56*1
) = 791 (Checked!)
12:
13:     #sequence:  .word   '*', '*', '*', '*' , '*', '*', '*', 1, 1, 1, 1, 1, 1, 1, 1

14:     #answer is 1*1*1*1*1*1*1*1 = 1 (Checked!)
15:
16:     #sequence:  .word   '-', '*', '*', '*' , '*', '*', '*', 1, 1, 1, 1, 1, 1, 1, 1
17:     #answer is 1*1*1*1 - 1*1*1*1 = 0 (Checked!)
18:
19:     #sequence:  .word   '*', '*', '*', '*' , '*', 1, 1, 1, 1, 1, 1
20:     #answer is 1*1*1*1*1*1 = 1 (Checked!)
21:
22:     #sequence:  .word   '*', '*', '*', '*' , '*', 1, 1, 1, 1, 1, 2
23:     #answer is 1*1*1*2*1*1 = 2 (Checked!)
24:
25:     #sequence:  .word   '-', '*', '*', '*' , '*', 2, 1, 1, 1, 1, 1
26:     #answer is 1 - 2 = -1 (Checked!)
27:
28:
29:     sequence:   .word   '-', '*', '*', '*' , '*', 1, 1, 1, 1, 1, 2
30:     #answer is 2 - 1 = 1 (Checked!)
31:
32:     num_of_elems:   .word   11
33:
34:     space_str:  .asciiz " "
35: .text
36: main:
37:     li  $t1,    0
38:     lw  $t6,    num_of_elems($t1)
39:
40:     beqz    $t6,    exit_programm
41:     jal DF_order_operate
42:
43: exit_programm:
44:     li  $t3,    0
45:     lw  $a0,    sequence($t3)
46:     #Печать элемента лежащего в a0
47:     li  $v0,    1
48:     syscall
```

```
49:
50:     #exit (terminate execution)
51:     li  $v0,    10
52:     syscall
53:
54: DF_order_operate:
55:     subu    $sp,    $sp,    8
56:
57:     #сохраним в stack указатель на место, из которого была вызвана DF_order_operate
58:     #и индекс элемента, в котором мы сейчас находимся, мы делаем это при
59:     #каждом вызове DF_order_operate = при каждом переходе к следующему элементу
60:     #так организуется стековый фрейм для рекурсии
61:     sw  $ra,    0($sp)
62:     sw  $t1,    4($sp)
63:
64: #Левый лист
65:     #i_ch_l = i_p*2 + 1
66:     mul $t1,    $t1,    2
67:     add $t1,    $t1,    1
68:
69:     #Проверка на существоввание ребенка. В первом проходе она заводит
70:     #нас к листьям дерева. Однако факт перезаписи подгружаемого
71:     # из стека значения t1 и адреса перехода $ra, их подгрузка
72:     #Позволяют просто переходить от родителя к ребенку, а не в конец дерева.
73:     bge $t1,    $t6,    operate_element_from_stack
74:     jal DF_order_operate            #делая jal, мы перезаписываем $ra
75:     #$ra — содержит адрес инструкции, из которого была вызвана функция.
76:
77: #Правый лист
78:     #Проходя по правой ветке мы подгружаем значение записанное в стековом фрейме
79:     #Таким образом мы продолжаем рекурсивное движение от рассматриваемого родителя
80:     #В сторону не рассмотренного ребенка.
81:     lw  $t1,    4($sp)
82:
83:     #i_ch_r = i_p*2 + 2
84:     mul $t1,    $t1,    2
85:     add $t1,    $t1,    2
86:
87:     bge $t1,    $t6,    operate_element_from_stack
88:     jal DF_order_operate        #делая jal, мы перезаписываем $ra
89:     #$ra — содержит адрес инструкции, из которого была вызвана функция.
90:
91:     #После прохода по правому листу из стекового фрейма вытаскивается значение
92:     #которое переводит нас к позиции переде меткой operate_element_from_stack.
93:     #Так организован переход от выведенного ребенка
94:     #К родителю с дальнейшим выводом и его.
95:
96:
97: operate_element_from_stack:
98:
99:     lw  $t1,    4($sp)
```

```
100:    mul $t3,    $t1,    4
101:
102:    lw  $a0,    sequence($t3)
103:
104:    #операции записывают ответ в $a3
105:    beq $a0,    43, sum_a1_a2   #Если в a0 действие делаем действие
106:    beq $a0,    42, mul_a1_a2   #Если в a0 действие делаем действие
107:    beq $a0,    45, sub_a1_a2   #Если в a0 действие делаем действие
108:    after_operation:
109:
110:    #Подгружаем записанное в stack значение $ra, чтобы продолжить исполнение
111:    lw  $ra,    ($sp)       #кода с того места, откуда были вызваны.
112:                    # на данном шаге рекурсии. Так, например, после
113:            #рассмотрнения левого ребенка мы перехоим в рассмотрение правого,
114:            #из рассмотренного правого - в метку print_element_from_stack
115:            #где производится сдвиг стекового фрейма - переход к родителю
116:
117:    #Здесь и работает рекурсия
118:    add $sp,    $sp,    8   #По результату выполнения этой строчки происходит
119:                    #сдвиг $sp - указателя на стековый фрейм, который
120:                    #определяет в какой участок кода мы попадем
121:                    # и с каким значением работаем.
122:
123:    #Далее происходит перезапись в DF_order_operate стекового фрейма с тем, чтобы
124:    #продолжить рекурсивное движение по нашему дереву.
125:
126:    #Поднявшись до самой верхушки рекурсивной лестницы мы попадем в начало стека,
127:    #где, пройдем по $ra,и окажемся перед меткой первого вызова DF_order_operate
128:    # - exit_programm.
129:
130:
131:    #Переход из ранее подгруженного элемента
132:    jr  $ra
133:
134: sum_a1_a2:
135: #у нас есть t1 - это указатель на дествие
136:    #t4 - указаетль на левого ребенка
137:    mul $t4,    $t1,    2
138:    add $t4,    $t4,    1
139:    mul $t4,    $t4,    4
140:
141:    #t5 - укаазетль на правкого ребенка
142:    mul $t5,    $t1,    2
143:    add $t5,    $t5,    2
144:    mul $t5,    $t5,    4
145:
146:    #подгуржаем значение из последовательности
147:    lw  $a1,    sequence($t4)
148:    lw  $a2,    sequence($t5)
149:
150:    #выполняем операцию
```

```
151:    add $a3,    $a2,    $a1
152:
153:    sw  $a3,    sequence($t3)
154:
155:    #возвращаемся
156:    j   after_operation
157:
158: mul_a1_a2:
159: #у нас есть t1 - это указатель на дествие
160:    #t4 - указаетль на левого ребенка
161:    mul $t4,    $t1,    2
162:    add $t4,    $t4,    1
163:    mul $t4,    $t4,    4
164:
165:    #t5 - укаазетль на правкого ребенка
166:    mul $t5,    $t1,    2
167:    add $t5,    $t5,    2
168:    mul $t5,    $t5,    4
169:
170:    #подгуржаем значение из последовательности
171:    lw  $a1,    sequence($t4)
172:    lw  $a2,    sequence($t5)
173:
174:    #выполняем операцию
175:    mul $a3,    $a2,    $a1
176:
177:    sw  $a3,    sequence($t3)
178:
179:    #возвращаемся
180:    j   after_operation
181:
182: sub_a1_a2:
183: #у нас есть t1 - это указатель на дествие
184:    #t4 - указаетль на левого ребенка
185:    mul $t4,    $t1,    2
186:    add $t4,    $t4,    1
187:    mul $t4,    $t4,    4
188:
189:    #t5 - укаазетль на правкого ребенка
190:    mul $t5,    $t1,    2
191:    add $t5,    $t5,    2
192:    mul $t5,    $t5,    4
193:
194:    #подгуржаем значение из последовательности
195:    lw  $a1,    sequence($t4)
196:    lw  $a2,    sequence($t5)
197:
198:    #выполняем операцию
199:    sub $a3,    $a1,    $a2
200:
201:    sw  $a3,    sequence($t3)
```

```
202:
203:    #возвращаемся
204:    j   after_operation
205:
```