04/25/14 06:28:38 /Users/Clayton/Desktop/Repos/Comp-Org/Assignment 3/Rieck_SPIM3.a

```
1  #-------------------------------------------------------------+
2  # Author: Clayton Rieck                                       |
3  #-------------------------------------------------------------+
4  # DESCRIPTION:                                                |
5  # This program finds the n'th Fibonacci number where          |
6  # 0 <= n <= 44                                                |
7  #-------------------------------------------------------------+
8  # INPUT:                                                      |
9  # The user will input an integer between 0 and 44 inclusive.|
10 #-------------------------------------------------------------+
11 # OUTPUT:                                                     |
12 # The program will output the n'th number in the Fibonacci   |
13 # sequence and the sum of all th terms leading up to that    |
14 # number. It was also display the sequence in reverse order.|
15 #-------------------------------------------------------------+
16         .data
17 prompt:         .asciiz         "Enter a number between 0 and 44: "
18 test:           .asciiz         "TEST"
19 endl:           .asciiz         "\n"
20         .text
21         .globl main
22
23 main:
24
25     # print the prompt
26     la   $a0,    prompt
27     li   $v0,    4
28     syscall
29
30     # read in int
31     li   $v0,    5
32     syscall
33
34     # moves inputted value to $a0
35     # $a0 = fib term
36     move $a0,    $v0
37
38     # $a1 = initial summation
39     li   $a1,    0
40
41     jal  fib
42
43 done:
44     li   $v0,    10
45     syscall
46
47 # ------------------------------
48         .data
49 zero:           .asciiz         "Summation: "
50         .text
51
```

```
52  # -------------------------------\
53  # Function that ultimately calls  \
54  # calls the recursive function     \
55  #-----------------------------------
56  fib:
57      # adjust stack pointer
58      addi  $sp,    $sp,    -20
59
60      # t0 = fib term
61      # t1 = summation
62      move  $t0,    $a0
63      move  $t1,    $a1
64
65      # if user entered 0
66      bne   $t0,    $zero,    start_fib
67
68      la    $a0,    zero
69      li    $v0,    4
70      syscall
71
72      # print 0
73      li    $a0,    0
74      li    $v0,    1
75      syscall
76
77      la    $a0,    endl
78      li    $v0,    4
79      syscall
80
81      j     $ra

82
83  # -------------------------------\
84  # Branch that jumps and links     \
85  # to the recursive function        \
86  #-----------------------------------
87  start_fib:
88
89      # a0 = fib term
90      # a1 = first fib term
91      # a2 = second fib term
92      # a3 = summation
93      move  $a0,    $t0
94      li    $a1,    0
95      li    $a2,    1
96      move  $a3,    $t1
97
98      jal   smart_fib
99
100     j     done
101
102 # -------------------------------
103         .data
104 hi_prompt:       .asciiz        "N'th Fibonacci Number: "
```

```
105 sum:            .asciiz        "Summation:              "
106 sequence:       .asciiz        "Sequence: \n"
107         .text
108
109 # ------------------------------\
110 # Recursive function definition   \
111 # ARGS: n    => term in sequence    \
112 #       p1   => previous' previous   \
113 #       p2   => previous number       \
114 #       sum  => summation (intial 0)   \
115 #-------------------------------------
116 smart_fib:
117
118     # adjust stack pointer
119     addi  $sp,    $sp,    -28
120
121     # move arguments to temp registers
122     # for later use
123     move  $t0,    $a0
124     move  $t1,    $a1
125     move  $t2,    $a2
126     move  $t3,    $a3
127
128     # compute the sum (sum = sum + second_arg)
129     add   $t3,    $t3,    $t2
130
131     bne   $t0,    0,      recurse  # if nth term equals 0
132
133     move  $v0,    $t2              # return current term (1)
134     move  $v1,    $t3              # return current sum (0+1)
135     add   $sp,    $sp,    28
136     j     $ra
137
138 # -------------------------------\
139 # Branch that continues the       \
140 # recursive function when the      \
141 # input is not 0                    \
142 #-----------------------------------
143 recurse:
144
145     # p1+p2
146     add   $t4,    $t2,    $t1
147
148     # store local variables in frame
149     sw    $t0,    0($sp)           # store fib term
150     sw    $t2,    4($sp)           # store p2
151     sw    $t4,    8($sp)           # store p1 + p2
152     sw    $t3,    12($sp)          # store summation
153     sw    $ra,    16($sp)          # store return address
154
155     sw    $t2,    20($sp)
156     sw    $t3,    24($sp)
157
158     # n = n - 1
```

```
159     addi  $t0,    $t0,    -1
160
161     # smart_fib(n-1, p2, p2+p1, summation) (python equivalent)
162     move  $a0,    $t0
163     move  $a1,    $t2
164     move  $a2,    $t4
165     move  $a3,    $t3
166
167     jal   smart_fib
168
169     # move down the stack
170
171     lw    $t1,    0($sp)              # fib term (n)
172
173     # if n doesn't equal 1
174     # just print out next fib number in
175     # the sequence
176     bne   $t1,    1,    move_down
177
178     la    $a0,    endl
179     li    $v0,    4
180     syscall
181
182     la    $a0,    hi_prompt
183     li    $v0,    4
184     syscall
185
186     # print p2 (fib number at n'th position)
187     lw    $a0,    4($sp)
188     li    $v0,    1
189     syscall
190
191     la    $a0,    endl
192     li    $v0,    4
193     syscall
194
195     la    $a0,    sum
196     li    $v0,    4
197     syscall
198
199     # print current summation
200     lw    $a0,    12($sp)
201     li    $v0,    1
202     syscall
203
204     la    $a0,    endl
205     li    $v0,    4
206     syscall
207
208     la    $a0,    endl
209     li    $v0,    4
```

```
210     syscall
211
212     # print sequence prompt
213     la    $a0,    sequence
214     li    $v0,    4
215     syscall
216
217 # -------------------------------\
218 # Handles the printing of the     \
219 # sequence as we move down the call\
220 # stack                            \
221 #-----------------------------------
222 move_down:
223
224     # load current fib number into a0
225     # to be printed
226     lw    $a0,    4($sp)
227     li    $v0,    1
228     syscall
229
230     la    $a0,    endl
231     li    $v0,    4
232     syscall
233
234     # load back the return address for
235     # the last call
236     lw    $ra,    16($sp)
237
238     # adjust the stack pointer
239     addi  $sp,    $sp,     28
240
241     j     $ra
```