

605.204 - Computer Organization

Module 11: Assignment

Nick Hinke

November 13, 2022

Brief Introduction

This assignment involves recursion within various assembly files in armv7l. All of my resulting code can be found at this *GitHub link* and can be cloned (along with pre-built binaries in a *bin/* folder) and viewed using the following commands:

```
git clone https://github.com/nhinke/computer-organization-repo.git
cd computer-organization-repo/assignments/module11/
```

The pre-built binaries can then be run using the following commands:

```
cd bin/
./pr1-recursiveMult
./pr2-recursiveFib
```

Note that each of the pre-built binaries will print out an example input-output sequence to the active terminal.

Problem 1

1. Implement a program to calculate multiplication using successive addition with recursion. For example, 5×4 is $5+5+5+5$. This can be defined recursively as:

Mult(m, n) = if n is 1, return m
 else return m + Mult(m, n-1)

Recursive Function:

```
63 .text
64 # recursive function mult(m,n) to multiply value in r0 (m) by value in r1 (n) using successive additions, and return result in r0
65 mult:
66
67     # push stack (and preserve inputs (m,n) in r4 and r5)
68     SUB sp, sp, #12
69     STR lr, [sp, #0]
70     STR r4, [sp, #4]
71     STR r5, [sp, #8]
72
73     # store inputs (m,n) in r4 and r5
74     MOV r4, r0
75     MOV r5, r1
76
77     # base case 0: if (n == 0) return 0
78     CMP r5, #0
79     MOVEQ r0, #0
80     BEQ return
81
82     # base case 1: if (n == 1) return m
83     CMP r5, #1
84     BNE else
85     MOV r0, r4
86     B return
87
88     # else return (m + mult(m,n-1))
89     else:
90     SUB r1, r5, #1
91     BL mult
92     ADD r0, r4, r0
93     B return
94
95     endif:
96
97     # pop stack and return
98     return:
99     LDR lr, [sp, #0]
100    LDR r4, [sp, #4]
101    LDR r5, [sp, #8]
102    ADD sp, sp, #12
103    MOV pc, lr
104
105 .data
106 # end mult
```

Figure 1: Screenshot of recursive function within multiplication program

Visit *this link* to see the rest of the program (it is too long to paste screenshots in a meaningful way).

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr1-recursiveMult
Please enter a positive integer m to compute the product m*n: 7
Please enter a positive integer n to compute the product m*n: 9
The product of 7*9 = 63
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr1-recursiveMult
Please enter a positive integer m to compute the product m*n: 6
Please enter a positive integer n to compute the product m*n: 1
The product of 6*1 = 6
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr1-recursiveMult
Please enter a positive integer m to compute the product m*n: 8
Please enter a positive integer n to compute the product m*n: 0
The product of 8*0 = 0
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr1-recursiveMult
Please enter a positive integer m to compute the product m*n: 9
Please enter a positive integer n to compute the product m*n: 5
The product of 9*5 = 45
```

Figure 2: Screenshot of program output

Problem 2

2. Implement a program to calculate a Fibonacci number recursively. A Fibonacci number is defined recursively as:

Fib(n) = if (n == 0 or n == 1) return 1
else return Fib(n-1) + Fib(n-2)

Recursive Function:

```
50 .text
51 # recursive function to compute the n'th Fibonacci number given an input n in r0 and returns result in r0
52 fibonacci:
53
54     # push stack
55     SUB sp, sp, #12
56     STR lr, [sp, #0]
57     STR r4, [sp, #4]
58     STR r5, [sp, #8]
59
60     # preserve input value of n in r4
61     MOV r4, r0
62
63     # base case 0: if (n == 0) return 0
64     CMP r4, #0
65     MOVEQ r0, #1
66     BEQ return
67
68     # base case 1: if (n == 1) return 1
69     CMP r4, #1
70     BNE else
71     MOV r0, #1
72     B return
73
74     # else return (fibonacci(n-1) + fibonacci(n-2))
75     else:
76     SUB r0, r4, #1
77     BL fibonacci
78     MOV r5, r0
79     SUB r0, r4, #2
80     BL fibonacci
81     ADD r0, r0, r5
82     B return
83
84     endif:
85
86     # pop stack and return
87     return:
88     LDR lr, [sp, #0]
89     LDR r4, [sp, #4]
90     LDR r5, [sp, #8]
91     ADD sp, sp, #12
92     MOV pc, lr
93
94 .data
95 # end fibonacci
```

Figure 3: Screenshot of recursive function within Fibonacci program

Visit *this link* to see the rest of the program (it is too long to paste screenshots in a meaningful way).

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 0
The n'th Fibonacci number for n = 0 is: 1
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 1
The n'th Fibonacci number for n = 1 is: 1
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 2
The n'th Fibonacci number for n = 2 is: 2
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 3
The n'th Fibonacci number for n = 3 is: 3
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 4
The n'th Fibonacci number for n = 4 is: 5
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 5
The n'th Fibonacci number for n = 5 is: 8
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 6
The n'th Fibonacci number for n = 6 is: 13
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 7
The n'th Fibonacci number for n = 7 is: 21
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module11 $ ./bin/pr2-recursiveFib
Please enter a positive integer value for n to compute the n'th Fibonacci number: 8
The n'th Fibonacci number for n = 8 is: 34
```

Figure 4: Screenshot of sample program output