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:

```
.text
# recursive function mult(m,n) to multiply value in r0 (m) by value in r1 (n) using successive additions, and return result in r0
mult:

  # push stack (and preserve inputs (m,n) in r4 and r5)
  SUB sp, sp, #12
  STR lr, [sp, #0]
  STR r4, [sp, #4]
  STR r5, [sp, #8]

  # store inputs (m,n) in r4 and r5
  MOV r4, r0
  MOV r5, r1

  # base case 0: if (n == 0) return 0
  CMP r5, #0
  MOVEQ r0, #0
  BEQ return

  # base case 1: if (n == 1) return m
  CMP r5, #1
  BNE else
    MOV r0, r4
    B return

  # else return (m + mult(m,n-1))
  else:
    SUB r1, r5, #1
    BL mult
    ADD r0, r4, r0
    B return

  endif:

  # pop stack and return
  return:
  LDR lr, [sp, #0]
  LDR r4, [sp, #4]
  LDR r5, [sp, #8]
  ADD sp, sp, #12
  MOV pc, lr

.data
# 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:

```
.text
# recursive function to compute the n'th Fibonacci number given an input n in r0 and returns result in r0
fibonacci:

  # push stack
  SUB sp, sp, #12
  STR lr, [sp, #0]
  STR r4, [sp, #4]
  STR r5, [sp, #8]

  # preserve input value of n in r4
  MOV r4, r0

  # base case 0: if (n == 0) return 0
  CMP r4, #0
  MOVEQ r0, #1
  BEQ return

  # base case 1: if (n == 1) return 1
  CMP r4, #1
  BNE else
    MOV r0, #1
    B return

  # else return (fibonacci(n-1) + fibonacci(n-2))
  else:
    SUB r0, r4, #1
    BL fibonacci
    MOV r5, r0
    SUB r0, r4, #2
    BL fibonacci
    ADD r0, r0, r5
    B return

  endif:

  # pop stack and return
  return:
  LDR lr, [sp, #0]
  LDR r4, [sp, #4]
  LDR r5, [sp, #8]
  ADD sp, sp, #12
  MOV pc, lr

.data
# 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:

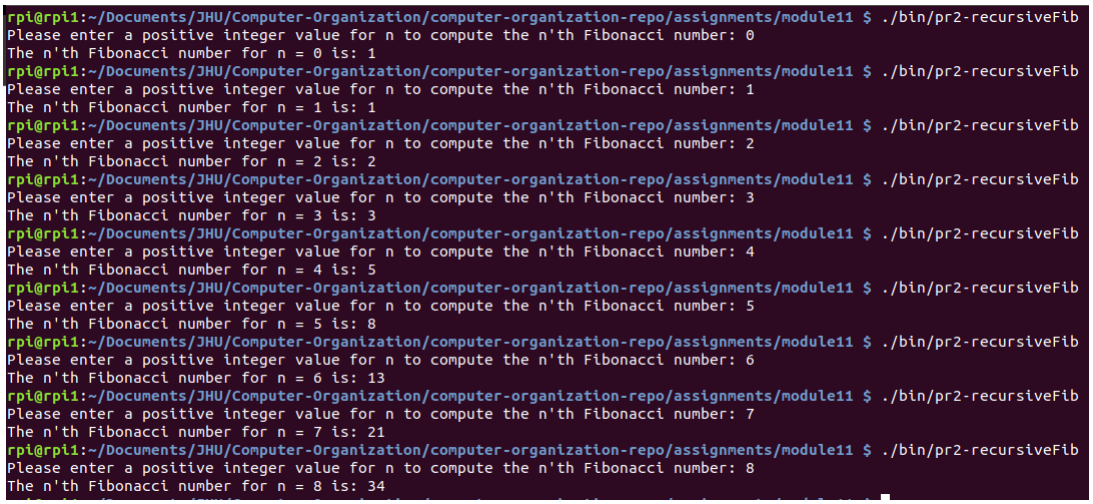

Figure 4: Screenshot of sample program output