

Roma Razdan

I pledge my honor that I have abided by the Stevens Honor System. Roma Razdan

My Approach

While a little challenging, this project allowed me to grow my skills in arm assembly as well as work to correctly implement a code that performs the Taylor series.

For the main method it was primitive I load the address of i into a register. As this value was an index, I had to load it into another parameter register. Moving this finally to a temporary register allowed me to prevent the value from changing. This concept in addition to subtracting 1 was used to index, since we are counting up to $i - 1$. Then I loaded x into $d1$, the double register, making sure $d1$ was working properly (which was one of the core registers). Additionally creating a count variable set to 1 allowed me to work seamlessly with the next loop where I would work through the Taylor series. An additional variable set at 1 would also work for float conversion in attempts of Taylor estimation. At the end of my main method I had to include the condition that if $i = 0$, there are no additional terms that must be included in the final result, then the program can go to done. Else, we utilize a double value for calculating the numerators of the remaining terms of the series and branch to the loop where this calculation is done (Taylor loop (L1)).

In loop one, $d1$ is set to x and $d3$ is set to 1, thus x is the first iteration and x^2 is the second iteration and we continue these iterations until the end of this loop. The rest of this method entails calculating the denominator which is in terms of double as seen through the initialization of a variable to 1.

For factorial and loop 2 it was vital to calculate the factorial of the count using the same idea for loop one where conversions to double occurred. By branching back to the caller, the loop ran more seamlessly since branching would not occur in every case (less breaks). As recursion was not necessary nor the use of stack, I found branching in these cases to be the best option. If this case is not used, the loop runs until it reaches the count. By checking if count is at $i - 1$ we can say we are done with this method. Else, increment the count and repeat loop.

Finally, we check if $d0$ is 0 and then add the current estimation to $d0$. By loading $fstring$ into $x0$ allowing $d0$ to print, we get the result of our Taylor series and can exit the code in its entirety.

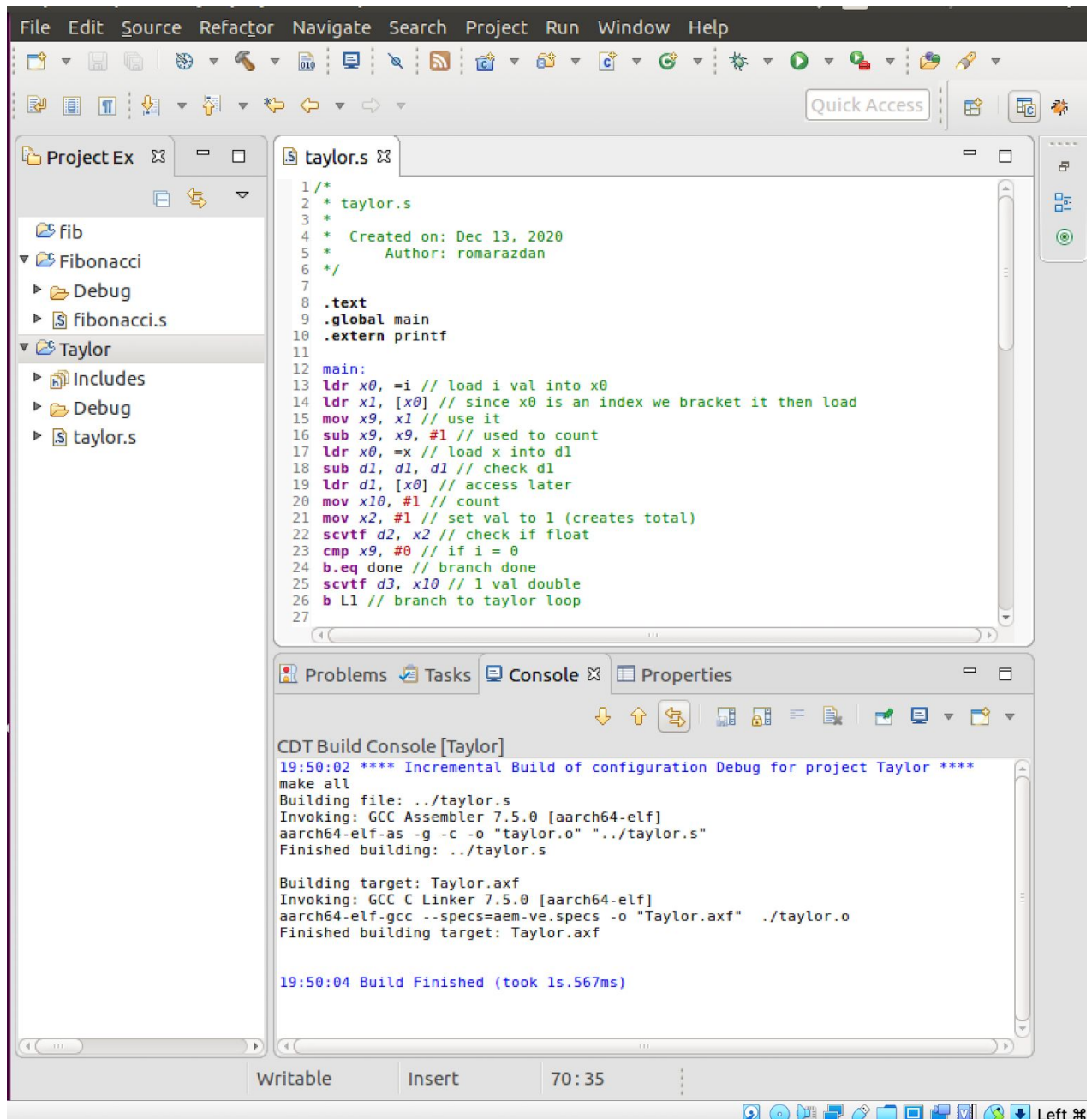
Overall, I found this project to be a great experience as I choose to utilize branching more so than recursion and stack as opposed to the first project. I also learned how double registers work and how to seamlessly run loops within other methods. Also I learned how to use `mul` (integer multiplication), `fmul` (double multiplication), `fdiv` (double division), and `scvtf` (conversion int to double). Lastly, I found it interesting to separate the numerator and denominator and then bring them together at the end of the code to produce the correct estimation!

Roma Razdan

I pledge my honor that I have abided by the Stevens Honor System. Roma Razdan

Screenshots:

Build Mode →



Roma Razdan

I pledge my honor that I have abided by the Stevens Honor System. Roma Razdan

Debug Mode →

File Edit Navigate Search Project Run Window Help

Quick Access

Debug C Project Remote

fibonacci_configuration connected
ARMv8-A #1 stopped on breakpoint (EL3h)
taylor_configuration connected
ARMv8-A #1 stopped on breakpoint (EL3h)

Status: connected

Linked: taylor_configuration

Register Set: All registers

Name	Value
AArch64	699 of 699 r
AArch32	534 of 534 r

taylor.s

```
1 /*
2  * taylor.s
3  *
4  * Created on: Dec 13, 2020
5  * Author: romarazdan
6  */
7
8 .text
9 .global main
10 .extern printf
11
12 main:
13 ldr x0, =1 // load i val into x0
14 ldr x1, [x0] // since x0 is an index we bracket
15 mov x9, x1 // use it
16 sub x9, x9, #1 // used to count
17 ldr x0, =x // load x into d1
18 sub d1, d1, d1 // check d1
19 ldr d1, [x0] // access later
20 mov x10, #1 // count
21 mov x2, #1 // set val to 1 (creates total)
22 scvtf d2, x2 // check if float
23 cmp x9, #0 // if i = 0
24 b.eq done // branch done
25 scvtf d3, x10 // 1 val double
26 b L1 // branch to taylor loop
27
28 L1:
29 fmul d3, d3, d1 // num_val (total)
```

App Con Target C Error Lo Console

Linked: taylor_configuration

Fast Models [11.4.35 (Jun 18 2018)]
Copyright 2000-2018 ARM Limited.
All Rights Reserved.

terminal_1: Listening for serial connection on port 5001
terminal_3: Listening for serial connection on port 5002
terminal_2: Listening for serial connection on port 5000
terminal_0: Listening for serial connection on port 5003
CADI server started listening to port 7001

Info: Foundation_AEMv8A: CADI Debug Server started for ARM Model
CADI server is reported on port 7001

Info: /OSCI/SystemC: Simulation stopped by user.

taylor_configuration connected (ARM...el - ARMv8-Ax1 Foundation Platform)