

Semal Shah

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

Code for Taylor.s

```
1 .text
2 .global main
3 .extern printf
4
5 main:
6     SUB sp, sp, #16
7     STR x30, [sp]
8     ldr x8, =terms
9     ldr x9, =x
10    ldr x0, [x8] // terms
11    ldr d1, [x9] // x
12    fmov d0, #1.0 // keeps the running count of answer
13    fmov d2, #1.0 // initializes the x keeper with 1 because of multiplication
14    fmov d4, #1.0 // factorial keeper
15    fmov d5, #1.0 // iteration counter
16    fmov d6, #1.0 // keeper for each iteration
17    fmov d9, #1.0 // a 1 to use
18    mov x10, #1
19    bl compute
20    ldr x30, [sp]
21    ldr x0, =prt_str1
22    add sp, sp, #16
23    bl printf
24    br x30
25 compute:
26    cmp x0, #0 // if terms is 0 then done
27    beq done
28    cmp x10, #1
29    beq store
30    fmul d2, d2, d1 // for the exponent so multiplies x by the previous values
31    fmul d4, d4, d5 // multiplies the iteration by the previous factorial
32    fdiv d3, d2, d4
33    fadd d0, d0, d3
34    fadd d5, d5, d9 // adding one to the iteration counter
35    sub x0, x0, #1 // subtracting one from the terms
36    bl compute
37 done:
38    ldr x30, [sp, #8]
39    br x30
40 store:
41    str x30, [sp, #8]
42    add x10, x10, #1
43    bl compute
44 .data
45 terms:
46     .word 6
47 x:
48     .double 5
49 prt_str1:
50     .ascii "The approximation is %f \n\0"
51
52 .end
--
```

I tried to keep my approach to taylor as simple as possible, while also managing the link register and function calling using bl. In my main function I load the 2 "inputs" into x0 and d1. X0 is the terms and d1 is the x value. I then initialized many registers to keep track of what is going on in compute. I also initialized d0 to 1 because the function would return 1 in its base case where  $n = 0$ . I could not do this in compute because it is running in a loop and it would've reinitialized the values every time giving me the wrong answer.

In compute a problem I ran into was not being able to return to the main after my calculations were done. It would only return correctly to main if the terms were 0. So, I figured out that I needed to manage the link register and load it on to the stack before running compute in a loop. To accomplish this I initialized x10 to 1 and if it was 1 when compute ran I would store x30 on to the stack then increment x10 by 1 so it would never be run again then, call compute. In compute on line 26-27 I make sure the amount of terms left are not 0. My general thinking in compute was to go bottom up instead of top down because that would be faster and arguably easier to code. Since, every iteration you would not have to call a factorial loop or function and a power loop or function. On line 30 I multiply the whatever is in the register d2 by d1 because my d2 register is there for keeping the value of the current exponent value before that iteration. So for example in the first iteration if x was 2 it would be  $1x2$ , next it would be  $2x2$ ,  $4x2$  and so on so I would not have to compute what I already computed before in terms of exponents. On line 31 I multiply the previous factorial needed by the iteration of the loop. I start the iteration variable at 1. And add one to it every time compute runs. This is for the same reason as before in where I would have the previous factorial answer so all I would have to do to get the new factorial needed for this iteration is to multiply it by the next number in line. On line 32 I now do the division of  $x^n/n!$ . Then, I add this to the running count of the answer in d0. After the calculation is done in each iteration I increment the iteration counter by 1 and subtract 1 from the amount of terms left. When terms finally equals 0 it will branch out to done and return to main.

After I return to the main I load x30 from the stack then load the string I want to print in x0. I then add the spots back on the stack. Then, I execute bl printf. Finally on line 24 I do br x30 and I am finished.

Pictures of the taylor.s printing the correct value to console and link register changing are below if needed.

## Before change in x registers

The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows the project structure with 'example-configuration connected' and 'ARMv8-A #1 stopped on breakpoint (EL3h)'.
- Source Editor:** Displays the assembly code for 'taylors.c'. The current line is line 6: `SUB sp, sp, #16`.
- Registers Window:** Shows the state of registers. The 'X' register (X0) is highlighted, showing a value of `0x0000000000000000`.
- Console:** Displays the output of the program, including the copyright notice and the start of the factorial calculation.

The registers window shows the following values for the 'X' register (X0):

Register	Value	Size	Access
X0	0x0000000000000000	64	R/W
X1	0x0000000000000000	64	R/W
X2	0x0000000000000000	64	R/W
X3	0x0000000000000000	64	R/W
X4	0x0000000000000000	64	R/W
X5	0x0000000000000000	64	R/W
X6	0x0000000000000000	64	R/W
X7	0x0000000000000000	64	R/W
X8	0x0000000000000000	64	R/W
X9	0x0000000000000000	64	R/W
X10	0x0000000000000000	64	R/W
X11	0x0000000000000000	64	R/W
X12	0x0000000000000000	64	R/W
X13	0x0000000000000000	64	R/W
X14	0x0000000000000000	64	R/W
X15	0x0000000000000000	64	R/W
X16	0x0000000000000000	64	R/W
X17	0x0000000000000000	64	R/W
X18	0x0000000000000000	64	R/W
X19	0x0000000000000000	64	R/W
X20	0x0000000000000000	64	R/W

## After change in x registers

The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows the project structure with 'example-configuration connected' and 'ARMv8-A #1 stopped on step1 (EL3h)'.
- Source Editor:** Displays the assembly code for 'taylors.c'. The current line is line 10: `ldr x0, [x0] // terms`.
- Registers Window:** Shows the state of registers. The 'X' register (X0) is highlighted, showing a value of `0x0000000000000000`.
- Console:** Displays the output of the program, including the copyright notice and the start of the factorial calculation.

The registers window shows the following values for the 'X' register (X0):

Register	Value	Size	Access
X0	0x0000000000000000	64	R/W
X1	0x0000000000000000	64	R/W
X2	0x0000000000000000	64	R/W
X3	0x0000000000000000	64	R/W
X4	0x0000000000000000	64	R/W
X5	0x0000000000000000	64	R/W
X6	0x0000000000000000	64	R/W
X7	0x0000000000000000	64	R/W
X8	0x0000000000000000	64	R/W
X9	0x0000000000000000	64	R/W
X10	0x0000000000000000	64	R/W
X11	0x0000000000000000	64	R/W
X12	0x0000000000000000	64	R/W
X13	0x0000000000000000	64	R/W
X14	0x0000000000000000	64	R/W
X15	0x0000000000000000	64	R/W
X16	0x0000000000000000	64	R/W
X17	0x0000000000000000	64	R/W
X18	0x0000000000000000	64	R/W
X19	0x0000000000000000	64	R/W
X20	0x0000000000000000	64	R/W

## Before change in d registers

The screenshot shows the Eclipse IDE interface for a DS-5 workspace. The main editor displays the assembly code for a program named 'taylor'. The code is as follows:

```
1 .text
2 .global main
3 .extern printf
4
5 main:
6 SUB sp, sp, #16
7 STR x30, [sp]
8 ldr x8, =terms
9 ldr x9, =x
10 ldr x0, [x8] // terms
11 ldr d1, [x9] // x
12 fmov d0, #1.0 // keeps the running count of answer
13 fmov d2, #1.0 // initializes the x keeper with 1 because of multiplication
14 fmov d4, #1.0 // factorial keeper
15 fmov d5, #1.0 // iteration counter
16 fmov d6, #1.0 // keeper for each iteration
17 fmov d9, #1.0 // a 1 to use
18 mov x10, #1
19 bl compute
20 ldr x30, [sp]
21 ldr x0, =prt_str1
22 add sp, sp, #16
23 bl printf
24 br x30
25 compute:
26 cmp x0, #0 // if terms is 0 then done
27 beq done
```

The registers window on the right shows the state of the registers. The 'All registers' tab is selected, and the 'Registers' list is expanded. The registers are organized by type: AArch64 (699 of 699 registers), Core (64 of 64 registers), SIMD (160 of 160 registers), Vec... (32 of 32 registers), Quad (32 of 32 registers), and Do... (32 of 32 registers). The registers are listed with their names, values, sizes, and access types. The registers D0 and D1 are highlighted in yellow, showing values 0.0 and 5.0 respectively.

The console window at the bottom shows the output of the program, including the copyright notice for ARM Limited and the status of the CADI server.

## After change in d registers

The screenshot shows the Eclipse IDE interface for a DS-5 workspace. The main editor displays the assembly code for a program named 'taylor'. The code is as follows:

```
1 .text
2 .global main
3 .extern printf
4
5 main:
6 SUB sp, sp, #16
7 STR x30, [sp]
8 ldr x8, =terms
9 ldr x9, =x
10 ldr x0, [x8] // terms
11 ldr d1, [x9] // x
12 fmov d0, #1.0 // keeps the running count of answer
13 fmov d2, #1.0 // initializes the x keeper with 1 because of multiplication
14 fmov d4, #1.0 // factorial keeper
15 fmov d5, #1.0 // iteration counter
16 fmov d6, #1.0 // keeper for each iteration
17 fmov d9, #1.0 // a 1 to use
18 mov x10, #1
19 bl compute
20 ldr x30, [sp]
21 ldr x0, =prt_str1
22 add sp, sp, #16
23 bl printf
24 br x30
25 compute:
26 cmp x0, #0 // if terms is 0 then done
27 beq done
```

The registers window on the right shows the state of the registers. The 'All registers' tab is selected, and the 'Registers' list is expanded. The registers are organized by type: AArch64 (699 of 699 registers), Core (64 of 64 registers), SIMD (160 of 160 registers), Vec... (32 of 32 registers), Quad (32 of 32 registers), and Do... (32 of 32 registers). The registers are listed with their names, values, sizes, and access types. The registers D0 and D1 are highlighted in yellow, showing values 1.0 and 5.0 respectively.

The console window at the bottom shows the output of the program, including the copyright notice for ARM Limited and the status of the CADI server.

Printing the correct value to console. Values for x and n shown in the picture

The screenshot displays the DS-5 Workspace interface with the following components:

- Project Explorer:** Shows the project structure with 'example-configuration connected' and 'ARMv8-A #1 running'.
- Source Code (taylor.s):**

```
27 beq done
28 cmp x10, #1
29 beq store
30 fmul d2, d2, d1 // for the exponent so multiplies x by the previous values
31 fmul d4, d4, d5 // multiplies the iteration by the previous factorial
32 fdiv d3, d2, d4
33 fadd d0, d0, d3
34 fadd d5, d5, d9 // adding one to the iteration counter
35 sub x0, x0, #1 // subtracting one from the terms
36 bl compute
37 done:
38 ldr x30, [sp, #8]
39 br x30
40 store:
41 str x30, [sp, #8]
42 add x10, x10, #1
43 bl compute
44 .data
45 terms:
46 .word 6
47 x:
48 .double 5
49 prt_str:
50 .ascii "The approximation is %f \n\0"
51
52 .end
```
- Registers:** A table showing the state of various registers. The 'D0' register contains the value 113.11805555555556.
- Console:** Displays the output of the program, including the approximation result: 113.118056.
- Source Not Found:** A message indicating that the source file for the 'printf.c' library was not found.

Name	Value	Size	Access
AArch64	699 of 699 registers		
Core	64 of 64 registers		
SIMD	160 of 160 registers		
Vec...	32 of 32 registers		
Quad	32 of 32 registers		
Do...	32 of 32 registers		
D0	113.11805555555556	64	R/W
D1	5.0	64	R/W
D2	15625.0	64	R/W
D3	21.70138888888889	64	R/W
D4	720.0	64	R/W
D5	7.0	64	R/W
D6	1.0	64	R/W
D7	-1.193550482098823...	64	R/W
D8	-1.193550482098823...	64	R/W
D9	1.0	64	R/W
D10	-1.193550482098823...	64	R/W
D11	-1.193550482098823...	64	R/W
D12	-1.193550482098823...	64	R/W
D13	-1.193550482098823...	64	R/W
D14	-1.193550482098823...	64	R/W
D15	-1.193550482098823...	64	R/W
D16	-1.193550482098823...	64	R/W
D17	-1.193550482098823...	64	R/W
D18	-1.193550482098823...	64	R/W
D19	-1.193550482098823...	64	R/W
D20	-1.193550482098823...	64	R/W
D21	-1.193550482098823...	64	R/W
D22	-1.193550482098823...	64	R/W
D23	-1.193550482098823...	64	R/W
D24	-1.193550482098823...	64	R/W
D25	-1.193550482098823...	64	R/W
D26	-1.193550482098823...	64	R/W
D27	-1.193550482098823...	64	R/W
D28	-1.193550482098823...	64	R/W
D29	-1.193550482098823...	64	R/W
D30	-1.193550482098823...	64	R/W
D31	-1.193550482098823...	64	R/W
D32	-1.193550482098823...	64	R/W
D33	-1.193550482098823...	64	R/W
D34	-1.193550482098823...	64	R/W
D35	-1.193550482098823...	64	R/W
D36	-1.193550482098823...	64	R/W
D37	-1.193550482098823...	64	R/W
D38	-1.193550482098823...	64	R/W
D39	-1.193550482098823...	64	R/W
D40	-1.193550482098823...	64	R/W
D41	-1.193550482098823...	64	R/W
D42	-1.193550482098823...	64	R/W
D43	-1.193550482098823...	64	R/W
D44	-1.193550482098823...	64	R/W
D45	-1.193550482098823...	64	R/W
D46	-1.193550482098823...	64	R/W
D47	-1.193550482098823...	64	R/W
D48	-1.193550482098823...	64	R/W
D49	-1.193550482098823...	64	R/W
D50	-1.193550482098823...	64	R/W
D51	-1.193550482098823...	64	R/W
D52	-1.193550482098823...	64	R/W
D53	-1.193550482098823...	64	R/W
D54	-1.193550482098823...	64	R/W
D55	-1.193550482098823...	64	R/W
D56	-1.193550482098823...	64	R/W
D57	-1.193550482098823...	64	R/W
D58	-1.193550482098823...	64	R/W
D59	-1.193550482098823...	64	R/W
D60	-1.193550482098823...	64	R/W
D61	-1.193550482098823...	64	R/W
D62	-1.193550482098823...	64	R/W
D63	-1.193550482098823...	64	R/W
D64	-1.193550482098823...	64	R/W
D65	-1.193550482098823...	64	R/W
D66	-1.193550482098823...	64	R/W
D67	-1.193550482098823...	64	R/W
D68	-1.193550482098823...	64	R/W
D69	-1.193550482098823...	64	R/W
D70	-1.193550482098823...	64	R/W
D71	-1.193550482098823...	64	R/W
D72	-1.193550482098823...	64	R/W
D73	-1.193550482098823...	64	R/W
D74	-1.193550482098823...	64	R/W
D75	-1.193550482098823...	64	R/W
D76	-1.193550482098823...	64	R/W
D77	-1.193550482098823...	64	R/W
D78	-1.193550482098823...	64	R/W
D79	-1.193550482098823...	64	R/W
D80	-1.193550482098823...	64	R/W
D81	-1.193550482098823...	64	R/W
D82	-1.193550482098823...	64	R/W
D83	-1.193550482098823...	64	R/W
D84	-1.193550482098823...	64	R/W
D85	-1.193550482098823...	64	R/W
D86	-1.193550482098823...	64	R/W
D87	-1.193550482098823...	64	R/W
D88	-1.193550482098823...	64	R/W
D89	-1.193550482098823...	64	R/W
D90	-1.193550482098823...	64	R/W
D91	-1.193550482098823...	64	R/W
D92	-1.193550482098823...	64	R/W
D93	-1.193550482098823...	64	R/W
D94	-1.193550482098823...	64	R/W
D95	-1.193550482098823...	64	R/W
D96	-1.193550482098823...	64	R/W
D97	-1.193550482098823...	64	R/W
D98	-1.193550482098823...	64	R/W
D99	-1.193550482098823...	64	R/W
D100	-1.193550482098823...	64	R/W
D101	-1.193550482098823...	64	R/W
D102	-1.193550482098823...	64	R/W
D103	-1.193550482098823...	64	R/W
D104	-1.193550482098823...	64	R/W
D105	-1.193550482098823...	64	R/W
D106	-1.193550482098823...	64	R/W
D107	-1.193550482098823...	64	R/W
D108	-1.193550482098823...	64	R/W
D109	-1.193550482098823...	64	R/W
D110	-1.193550482098823...	64	R/W
D111	-1.193550482098823...	64	R/W
D112	-1.193550482098823...	64	R/W
D113	-1.193550482098823...	64	R/W
D114	-1.193550482098823...	64	R/W
D115	-1.193550482098823...	64	R/W
D116	-1.193550482098823...	64	R/W
D117	-1.193550482098823...	64	R/W
D118	-1.193550482098823...	64	R/W
D119	-1.193550482098823...	64	R/W
D120	-1.193550482098823...	64	R/W
D121	-1.193550482098823...	64	R/W
D122	-1.193550482098823...	64	R/W
D123	-1.193550482098823...	64	R/W
D124	-1.193550482098823...	64	R/W
D125	-1.193550482098823...	64	R/W
D126	-1.193550482098823...	64	R/W
D127	-1.193550482098823...	64	R/W
D128	-1.193550482098823...	64	R/W
D129	-1.193550482098823...	64	R/W
D130	-1.193550482098823...	64	R/W
D131	-1.193550482098823...	64	R/W
D132	-1.193550482098823...	64	R/W
D133	-1.193550482098823...	64	R/W
D134	-1.193550482098823...	64	R/W
D135	-1.193550482098823...	64	R/W
D136	-1.193550482098823...	64	R/W
D137	-1.193550482098823...	64	R/W
D138	-1.193550482098823...	64	R/W
D139	-1.193550482098823...	64	R/W
D140	-1.193550482098823...	64	R/W
D141	-1.193550482098823...	64	R/W
D142	-1.193550482098823...	64	R/W
D143	-1.193550482098823...	64	R/W
D144	-1.193550482098823...	64	R/W
D145	-1.193550482098823...	64	R/W
D146	-1.193550482098823...	64	R/W
D147	-1.193550482098823...	64	R/W
D148	-1.193550482098823...	64	R/W
D149	-1.193550482098823...	64	R/W
D150	-1.193550482098823...	64	R/W
D151	-1.193550482098823...	64	R/W
D152	-1.193550482098823...	64	R/W
D153	-1.193550482098823...	64	R/W
D154	-1.193550482098823...	64	R/W
D155	-1.193550482098823...	64	R/W
D156	-1.193550482098823...	64	R/W
D157	-1.193550482098823...	64	R/W
D158	-1.193550482098823...	64	R/W
D159	-1.193550482098823...	64	R/W
D160	-1.193550482098823...	64	R/W
D161	-1.193550482098823...	64	R/W
D162	-1.193550482098823...	64	R/W
D163	-1.193550482098823...	64	R/W
D164	-1.193550482098823...	64	R/W
D165	-1.193550482098823...	64	R/W
D166	-1.193550482098823...	64	R/W
D167	-1.193550482098823...	64	R/W
D168	-1.193550482098823...	64	R/W
D169	-1.193550482098823...	64	R/W
D170	-1.193550482098823...	64	R/W
D171	-1.193550482098823...	64	R/W
D172	-1.193550482098823...	64	R/W
D173	-1.193550482098823...	64	R/W
D174	-1.193550482098823...	64	R/W
D175	-1.193550482098823...	64	R/W
D176	-1.193550482098823...	64	R/W
D177	-1.193550482098823...	64	R/W
D178	-1.193550482098823...	64	R/W
D179	-1.193550482098823...	64	R/W
D180	-1.193550482098823...	64	R/W
D181	-1.193550482098823...	64	R/W
D182	-1.193550482098823...	64	R/W
D183	-1.193550482098823...	64	R/W
D184	-1.193550482098823...	64	R/W
D185	-1.193550482098823...	64	R/W
D186	-1.193550482098823...	64	R/W
D187	-1.193550482098823...	64	R/W
D188	-1.193550482098823...	64	R/W
D189	-1.193550482098823...	64	R/W
D190	-1.193550482098823...	64	R/W
D191	-1.193550482098823...	64	R/W
D192	-1.193550482098823...	64	R/W
D193	-1.193550482098823...	64	R/W
D194	-1.193550482098823...	64	R/W
D195	-1.193550482098823...	64	R/W
D196	-1.193550482098823...	64	R/W
D197	-1.193550482098823...	64	R/W
D198	-1.193550482098823...	64	R/W
D199	-1.193550482098823...	64	R/W
D200	-1.193550482098823...	64	R/W
D201	-1.193550482098823...	64	R/W
D202	-1.193550482098823...	64	R/W
D203	-1.193550482098823...	64	R/W
D204	-1.193550482098823...	64	R/W
D205	-1.193550482098823...	64	R/W
D206	-1.193550482098823...	64	R/W
D207	-1.193550482098823...	64	R/W
D208	-1.193550482098823...	64	R/W
D209	-1.193550482098823...	64	R/W
D210	-1.193550482098823...	64	R/W
D211	-1.193550482098823...	64	R/W
D212	-1.193550482098823...	64	R/W
D213	-1.193550482098823...	64	R/W
D214	-1.193550482098823...	64	R/W
D215	-1.193550482098823...	64	R/W
D216	-1.193550482098823...	64	R/W
D217	-1.193550482098823...	64	R/W
D218	-1.193550482098823...	64	R/W
D219	-1.193550482098823...	64	R/W
D220	-1.193550482098823...	64	R/W
D221	-1.193550482098823...	64	R/W
D222	-1.193550482098823...	64	R/W
D223	-1.193550482098823...	64	R/W
D224	-1.193550482098823...	64	R/W
D225	-1.193550482098823...	64	R/W
D226	-1.193550482098823...	64	R/W
D227	-1.193550482098823...	64	R/W
D228	-1.193550482098823...	64	R/W
D229	-1.193550482098823...	64	R/W
D230	-1.193550482098823...	64	R/W
D231	-1.193550482098823...	64	R/W
D232	-1.193550482098823...	64	R/W
D233	-1.193550482098823...	64	R/W
D234	-1.193550482098823...	64	R/W
D235	-1.193550482098823...	64	R/W
D236	-1.193550482098823...	64	R/W
D237	-1.193550482098823...	64	R/W
D238	-1.193550482098823...	64	R/W
D239	-1.193550482098823...	64	R/W
D240	-1.193550482098823...	64	R/W
D241	-1.193550482098823...	64	R/W
D242	-1.193550482098823...	64	R/W
D243	-1.193550482098823...	64	R/W
D244	-1.193550482098823...	64	R/W
D245	-1.193550482098823...	64	R/W
D246	-1.193550482098823...	64	R/W
D247	-1.193550482098823...	64	R/W
D248	-1.193550482098823...	64	R/W
D249	-1.193550482098823...	64	R/W
D250	-1.193550482098823...	64	R/W