Rashmika Batra

Programming Assignment #2

First, in these lines I initialized my output to 1 and made it so that if the power is 0, return 1.

```
fmov d1, d0
fmov d0, #1
cmp x0, #0
beq return
```

Then, in the inner loop I made a condition that if the power is 1, return the result. Otherwise, multiply the result by the original value and decrement the counter.

```
cmp x0, #0
beq return
fmul d0, d0, d1
sub x0, x0, #1
b exp_inner
```

Then, within my absolute value function, I made a condition that if the output is greater than 0, return input. Otherwise, just multiply it by negative 1.

```
fcmp d0, #0
bge return
fmov d1, #-1
fmul d0, d0, d1
br x30
```

Now, I have a calculating term equation in which I first have the address of the coefficient array, then I multiplied the index by 8 to find the offset.

```
 adr x1, coeff
 lsl x2, x0, #3
```

Then, I added the offset to the address of the array and loaded the array element into the register. Then I set d0 equal to x^index and multiplied x^index by the coefficient in these lines:

```
add x2, x2, x1
ldur d2, [x2, #0]
sub x28, x28, #8
stur x30, [x28, #0]
bl exp
ldur x30, [x28, #0]
add x28, x28, #8
fmul d0, d0, d2
```

```
br x30
```

Then, I move onto my find y loop. In this loop, I am comparing the counter to the highest power. Then, I set a condition that if my counter is higher than the highest power, I break out of the loop. Then, I move the counter to x0 for my procedure call. d0 is the value for my coefficient multiplied by x^counter.

```
cmp x5, x7
bhi return
mov x0, x5
fmov d0, d3
sub x28, x28, #8
stur x30, [x28, #0]
bl calculate_term
ldur x30, [x28, #0]
add x28, x28, #8
```

Then, I add the value of the term to sum and copy the sum into d0 in case this is the last iteration, and finally I increment the counter.

```
fadd d4, d4, d0
fmov d0, d4
add x5, x5, #1
b find_y_loop
```

Then, I enter my loop that finds the root. First, I find the sum of a and b. Then, I divide this sum by 2 to find the average value. Then, I move that value to d0 for the procedure call.

```
 fadd d8, d6, d5
fmov d9, #2
fdiv d8, d8, d9
fmov d0, d8
sub x28, x28, #8
stur x30, [x28, #0]
```

Then, I find the y value of the function where x is the average value of a and b. Then, I copy that value into d10 for storage. Then, I find the absolute value of f(avg(a,b)). Then, I compare the absolute value to the accuracy constant. If it is less than the accuracy constant, return the value. Then, I set a condition that if the y value is less than 0, i will not set a right bound, otherwise I do and go to the next iteration of the loop.

```
bl find_y
fmov d10, d0
bl absolute_value
fmov d11, d0
fmov d0, d8
ldur x30, [x28, #0]
add x28, x28, #8
fcmp d11, d7
ble return
fcmp d10, #0
ble less_than
fmov d6, d8
b find_root_loop
```

Lastly, I move into my less_than function where I simply create a condition in which if the y value is equal to zero, I will not set a new left bound. Otherwise, I set a new left bound and iterate the loop.

```
fcmp d10, #0
bge equal
fmov d5, d8
b find_root_loop
```