FLOAT

<u>Summary of the FLOAT assignment. The FLOAT assignment is worth 1% of the course grade.</u>

There is 1 submission:

1. *Float.ans* is worth 1% of the course grade. This file is created by the grading system when you grade FLOAT. The grade is based on correct answers. See calendar for the due date.
2. If you need help from the staff:
   - submit your assembler source code to the ASK4HELP locker
   - send a note to the staff stating you have submitted code to ASK4HELP
   - explain the problem to be addressed
   - Assure your code is fully documented with header blocks and line comments. The staff may request to see your design.

You and I will calculate pi using the Salamin-Brent algorithm[1]. This algorithm was announced independently in 1976 by both mathematicians.

I will provide the algorithm, the pseudo code for the algorithm, a full C program that mirrors the code needed for the assignment, and the assembler code for all the required supporting functions. You will write the floating point code for the first three statements in the algorithm. This is estimated at about 26 instructions.

This is the algorithm.

$$\pi = \frac{4 * (a_n)^2}{1 - \sum_{k=0}^{n} 2^k * (a_k - b_k)^2} \qquad \begin{aligned} & n=0,1,2, \ldots \\ & a_0 = 1 \\ & b_0 = 1 / \mathrm{sqrt}(2) \\ & a_{n+1} = (a_n + b_n)/2 \\ & b_{n+1} = \mathrm{sqrt}(a_n * b_n) \end{aligned}$$

This pseudo code for the algorithm loops until it reaches the limit of the precision of the variables and value of pi being calculated stops changing.

```
float a, b, c, d, s, t, pi, old;

a   = 1.0                          // a0 = 1
b   = 1.0 / sqrt(2.0)              // b0  = 1 / sqrt(2)
s   = 1.0                          // s is the sum in the denominator
t   = 1.0                          // 2^0 the first value of 2^k when k = 0
old = 0.0                          // last value of pi calculated

while (true)                       // loop forever
   {
   s  = s - t * (a - b) * (a - b)  // subtract next value of ∑ from sum
   pi = 4 * a * a / s              // calc new value of pi
   c  = (a + b) / 2.0              // calc an+1 = (an + bn)/2

   d  = sqrt (a * b)               // calc bn+1 = sqrt(an * bn)
   a  = c                          // set an+1
   b  = d                          // set bn+1
   t  = 2 * t                      // calc next value of 2^k
   output(pi)                      // print the current value of pi
   if (pi == old) break            // exit if pi is not changing
   old = pi                        // save the current value of pi
   }
```

You code these 3 instructions (the three lines beginning `s  = s - t * (a - b) * (a - b)`, `pi = 4 * a * a / s` and `c  = (a + b) / 2.0`)

1. Microcomputers and Mathematics by Bruce, Giblin, Rippon. Cambridge University Press 1990. ISBN 0-521-31238

Below are the values calculated in EXCEL for the first four terms using five fractional digits.

	n=0		n=1		n=2		n=3	
	a0	b0	a1	b1	a2	b2	a2	b2
Initial values	1.00000	0.7071						
$a_{n+1} = (a_n + b_n) / 2$			0.85355		0.84722		0.84721	
$b_{n+1} = sqrt (a_n * b_n)$				0.84090		0.84720		0.84721
$4 * (a_n)^2$	4.00000		2.91421		2.87116		2.87108	
$k0 = 2^0 * (a_0 - b_0)^2$	0.08579							
1-k0	0.91421							
$PI = 4 * (a_0)^2 / (1 - k0)$	4.37535							
$k1 = 2^1 * (a_1 - b_1)^2$			0.00032					
1-k0-k1			0.91389					
$PI = 4 * (a_1)^2 / (1 - k0-k1)$			3.18879					
$k2 = 2^2 * (a_2 - b_2)^2$					0.00000			
1-k0-k1-k2					0.91389			
$PI = 4 * (a_2)^2 / (1 - k0-k1-k2)$					3.14168			
$k3 = 2^3 * (a_3 - b_3)^3$							0.00000	
1-k0-k1-k2-k3							0.91389	
$PI = 4 * (a_3)^2 / (1 - k0-k1-k2-k3)$							3.14159	

Steps to complete this assignment

Step 1. Create a design.

Read the docuemts provided that discuss floating-point instructions.
Determine how to calculate the first three statements in the algorithm.

Step 2. Code your solution.

Retrieve the testing and grading files. All files are packed together in one self-extracting file named *unpack.exe*.
Go to the class homepage. Click on *Lockers With Program Grading System Files.* Click on the *FLOAT* assignment.
Download the *unpack.exe* file and save it on your hard drive in the \P23X\FLOAT directory.
Start DOSBox and execute unpack.exe to create the testing and grading files.

One of the files is a model for your code.

- The MASM version is *float.m.*
- Rename *float.m* to be *float.asm* and add your code to *float.asm*.

Two of the files are subroutines needed to complete the assignment.

- The sqroot subroutine calculates the square root of a number.
- The output subroutine displays your current value of pi.

Use the commands below to assemble and link your main float assignment with the two subroutines used to calculate square roots and output the current value of pi.

MASM
ml /c /Zi /Fl float.asm
link /CO float.obj sqroot.obj output.obj

Step 3. Test and debug your solution.

We are not logging your testing for this assignment.

Run the program by typing**:** ***float***

The output should look like this. Each line represents one iteration of the algorithm.
Using 32-bit floating-point values we can correctly calculate six digits.

```
4.37534
3.18879
3.14168
3.14159
3.14159
```

Step 4. Grading

Type the following DOS command: ***gradfl***

Your grade will be based 100% for getting the correct answer.

Step 5. Submit your solution

Submit the file ***float.ans*** to the submit assignment named ***float***

Incorrect electronic submissions will result in your assignment not being graded and considered late.

Additional Files in unpack.exe

When you unpack the grading files you will have the following source and executable code.

FLOATPI.C	The C source code for the whole assignment.
FLOATPI.EXE	The executable for the compiled C code.
SQROOT.ASM	The assembler source code for the routine that calculates the square root of a number.
SQROOT.OBJ	The object code you link with your float main program.
OUTPUT.ASM	The assembler source code for the routine that displays the current value of pi.
OUTPUT.OBJ	The object code you link with your float main program.