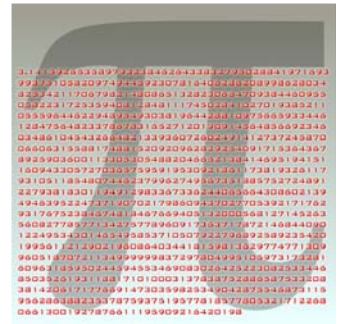
Float Assignment

Print and use the specification document on the class web site.

See syllabus for assignment type

individual

Salamin-Brent algorithm for pi announced independently in 1976 by both mathematicians.



Salamin-Brent algorithm for pi announced independently in 1976 by both mathematicians.

Give you

- the formula
- the pseudocode
- the code for most of the program

Salamin-Brent algorithm for pi announced independently in 1976 by both mathematicians.

$$pi = \frac{4 * (a_n)^2}{1 - \sum_{k=0}^{n} 2^k * (a_k - b_k)^2}$$

$n = 0, 1, 2, \ldots$

$a_0 = 1$

$b_0 = 1 / \text{sqrt}(2)$

$a_{n+1} = (a_n + b_n)/2$

$b_{n+1} = \text{sqrt}(a_n * b_n)$


```
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                    // sum in denominator
t   = 1.0                    // 2^0 first value of 2^k
old = 0.0                    // value of pi calculated

while (true)                 // loop forever
  {
  s  = s - t * (a - b) * (a - b) // sub val 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 current value of pi
  if (pi == old) break       // exit if pi not changing
  old = pi                   // save current value of pi
  }
```


You code these 3 lines

≈ 26 instructions

push, pop, arithmetic

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

a   =
b   =
s   =                                    or
t   =
old =                                    ed

while
  {
  s  = s - t * (a - b) * (a - b) // sub val 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 current value of pi
  if (pi == old) break       // exit if pi not changing
  old = pi                   // save current value of pi
  }
```


We provide

square root, compare,

output code

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

a   =
b   =
s   =                                    or
t   =
old =                                    ed

while
  {
  s  = s - t * (a - b) * (a - b) // sub val 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 current value of pi
  if (pi == old) break       // exit if pi not changing
  old = pi                   // save current value of pi
  }
```

Step 1. Create a design

Use the documents provided as reference

Decide how to code the 3 statements

Step 2. Code your assembler solution

Retrieve *unpack.exe* from *float* locker

Type *unpack* to build grading system

One file is .... *float.m*
Rename it to be .... *float.asm*
Add your code to .... *float.asm*

Two files are .... *sqroot.obj* *output.obj*

Link *float.obj* + *sqroot.obj* + *output.obj*

Step 3. Test and debug your solution

To test type: *float*

Should output:

4.37534
3.18879
3.14168
3.14159
3.14159

Step 4. Grading

To grade type: *gradfl*

The final grade will be based on:
- **100% for the correct answers**

Step 5. Submit your assignment

float.ans

(The only acceptable file)