Задание

- Изучить архитектуру и средства программирования сопроцессора на языке ассемблера. Изучить численный метод решения нелинейного уравнения, заданный в варианте
- Написать программу, реализующую **метод хорд** для функции $x + ln(x + 0.5) - 0.5 = 0,\ [0,\ 2]$. Программа должна состоять из модулей на C++ и ассемблере, причем в модуле на C++ осуществляется ввод-вывод, а все вычисления – в модуле на ассемблере. Входными данными является точность решения, выходными являются решение и число итераций.
- Для функции необходимо использовать алгоритм вычисления выражений в постфиксной записи.
- Отладить программу, убедиться в правильности ее работы на тестовых примерах.

Алгоритм

Пусть x_1и x_2– абсциссы концов хорды, $f(x) = 0$ – уравнение функции, решаемое методом хорд. Найдем коэффициенты k и b из системы уравнений.

$$\begin{cases} f(x_1) = kx_1 + b, \\ f(x_2) = kx_2 + b. \end{cases}$$

Вычтем из второго уравнения первое:

$$f(x_2) - f(x_1) = k(x_2 - x_1)$$

Затем найдем коэффициенты:

$$k = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

$$b = f(x_1) - x_1 \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

Уравнение принимает вид:

$$y = f(x_1) + (x - x_1) \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

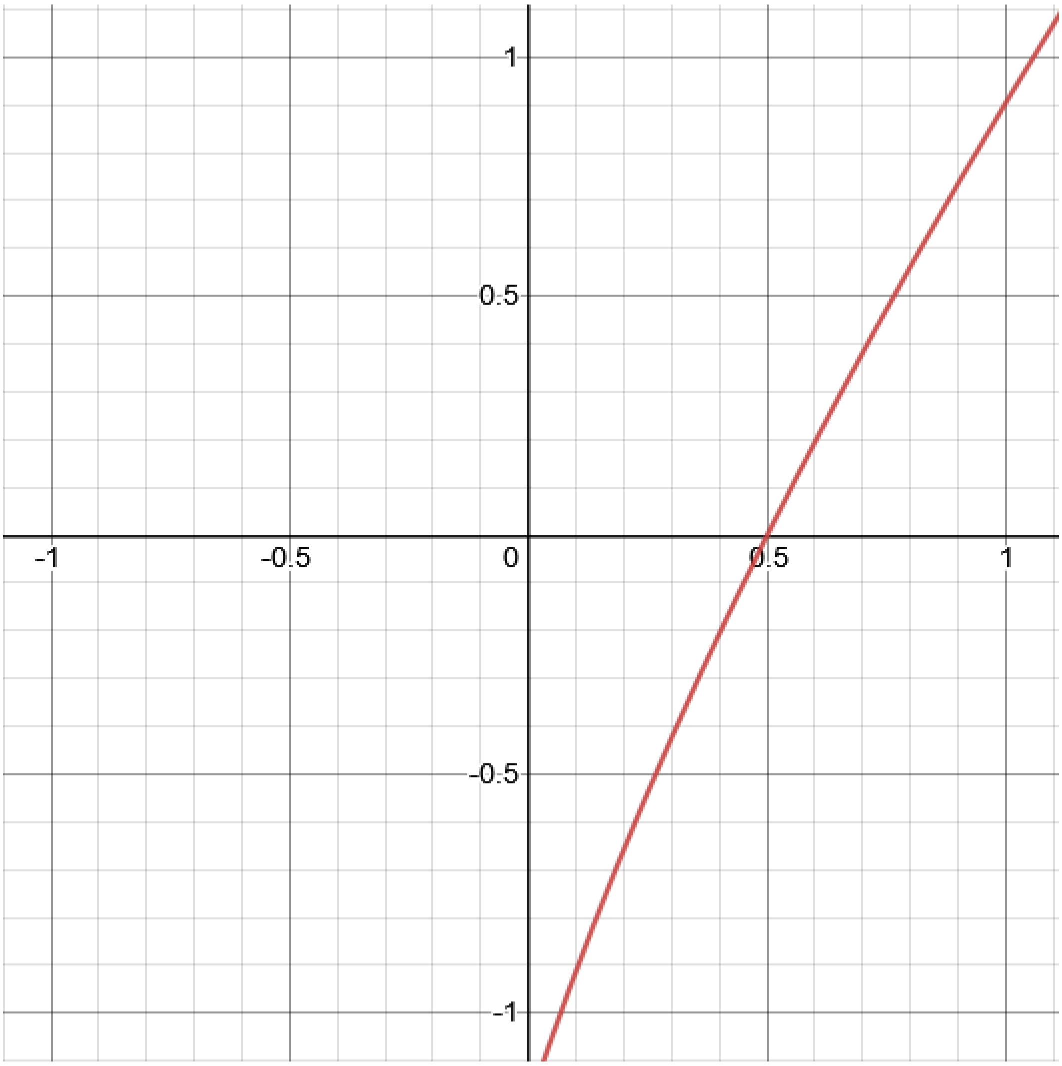
Таким образом можно найти первое приближение к корню:

$$x_3 = x_1 - \frac{(x_2 - x_1) f(x_1)}{f(x_2) - f(x_1)}$$

Проводя аналогичные операции над x_2и x_3найдем новое приближение к корню. Таким образом получаем итерационную формулу:

$$x_{i+1} = x_i - \frac{(x_{i+1} - x_i)\, f(x_i)}{f(x_{i+1}) - f(x_i)}$$

Исходная функция $x + ln(x + 0.5) - 0.5 = 0$ имеет следующий график:



Текст программы

```
.386
.MODEL FLAT
.STACK 4096

.DATA
      CNST DQ    0.5
      PREV DQ    0.0
      CURR DQ    2.0
      EPS  DQ    ?

.CODE

LN PROC

      FLD1
      FXCH  ST(1)
      FYL2X
      FLDL2E
      FDIVP

RET
LN ENDP

FUNC PROC

LNX:
      FST   ST(1)
      FADD  CNST
      CALL  LN

FX:
      FADD  ST, ST(1)
      FSUB  CNST

RET
FUNC ENDP

X_K1 PROC
      FINIT

FXCURR:
      FLD         PREV
      CALL  FUNC; f(prev) -> ST(0) prev -> ST(1)

FXPREV:
      FLDZ
      FLD         CURR
      CALL  FUNC; f(prev) -> ST(2) curr -> ST(1) f(curr) -> ST(0)
```

```
_F_SUB:
      FSUB  ST(2), ST; (f(prev) - f(curr)) -> ST(2)

_SUB:
      FLD         PREV
      FSUB  ST, ST(2); (prev - curr) -> ST(0)

_DIV:
      FDIV  ST, ST(3); (prev - curr)/(f(prev) - f(curr)) -> ST(0)

_MUL:
      FMUL  ST, ST(1)

_RES:
      FSUBR ST, ST(2); next -> ST(0)

RET
X_K1 ENDP

ERROR PROC

      FINIT
      FLD         CURR
      FLD         PREV
      FSUB  ST, ST(1)
      FABS

RET
ERROR ENDP

PUBLIC _SolveByChord@12
_SolveByChord@12 PROC

PREP:
      FINIT
      PUSH  EBP
      MOV   EBP, ESP
      MOV   ECX, [EBP + 8]
      FLD   QWORD PTR [EBP + 12]
      FSTP  EPS
```

```
MAINLOOP:
      CALL  X_K1
      FLD   CURR
      FSTP  PREV
      FST   CURR

      INC   DWORD PTR [ECX]

      CALL  ERROR
      FCOM  EPS
      FSTSW AX
      SAHF
      JNC         MAINLOOP

      FLD   CURR
      POP   EBP

RET 12
_SolveByChord@12 ENDP

END
```

```
#include <stdio.h>

extern double __stdcall SolveByChord(int* out, double epsilon);

double InitializeEpsilon()
{
      double epsilon;

      printf_s("let epsilon is ");
      scanf_s("%lf", &epsilon);

      return epsilon;
}

void main()
{
      const char* equation = "'x + ln(x + 0.5) - 0.5 = 0'";
      printf_s("Solving the %s by the chord method\n", equation);

      double epsilon = InitializeEpsilon();

      int iterations = 0;

      double result = SolveByChord(&iterations, epsilon);
      printf_s("\nResult: %.10lf, epsilon: %.10lf\nIterations:
%d", result, epsilon, iterations);
}
```

Тестирование

№	epsilon	x(double)	iterations	x*	|x - x*|
1	1	0,4636964	2	0,5000000	0,0363036
2	0,1	0,5013936	3	0,5000000	0,0013936
3	0,01	0,5000128	4	0,5000000	0,0000128
4	0,001	0,5000000	5	0,5000000	0,0000000
5	0,0001	0,5000000	5	0,5000000	0,0000000
6	0,00001	0,5000000	6	0,5000000	0,0000000
7	0,000001	0,5000000	6	0,5000000	0,0000000