# Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра прикладной математики

### Лабораторная работа № 3 по дисциплине «Языки программирования и методы трансляции»

**Программирование сопроцессора**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |  |  |
| Группа: | ПМ-63 |  |  |
| Студент: | Шепрут И.И. |  |  |
| Вариант: | 8 |  |  |
| Преподаватели: | Еланцева И.Л.  Петров Р.В. |  |  |

Новосибирск

2018

1. Задание
2. Изучить архитектуру и средства программирования сопроцессора на языке ассемблера. Изучить численный метод решения нелинейного уравнения, заданный в варианте.
3. Написать программу, реализующую метод **Ньютона с подвижным полюсом для функции ln((x-1)/3)/3 = 0, x0 = 1.5.**
4. Программа должна состоять из модулей на С++ и ассемблере, причем в модуле на С++ осуществляется ввод-вывод, а все вычисления – в модуле на ассемблере.
5. Следует оформить отдельными процедурами на ассемблере:
   * вычисление функции, задающей левую часть уравнения;
   * вычисление (если требуется) производных функции;
   * вычисления, связанные с методом решения.
6. Входными данными является точность решения. Выходными данными являются решение и число итераций.
7. Для функции и (если требуется) ее производных необходимо использовать алгоритм вычисления выражений в постфиксной записи (любые отклонения от него должны быть обоснованы повышением эффективности). Необходимо обращать особое внимание на грамотное использование стека сопроцессора.
8. Отладить программу, убедиться в правильности ее работы на тестовых примерах.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. Алгоритм   Функция:    **Метод Ньютона с подвижным полюсом:**    Если подставить значения, то получится:    Если заменить обращения к функциям на буквы, то получится итоговая формула:     1. Текст программы  |  | | --- | | .386  .MODEL FLAT  .DATA  x REAL4 ?  d REAL4 ?  dp REAL4 ?  c3 REAL4 3.  c2 REAL4 2.  ;------------------------------------------------------------------------------  ;------------------------------------------------------------------------------  ;------------------------------------------------------------------------------  .CODE  ;------------------------------------------------------------------------------  f PROC ; d = f(x)  ; log2((x-1)/3)/log2(e)/3 = 1 1 x - 3 / log2 log2(e) / 3 /  FLD1 ; Stack: 1  FLD1 ; Stack: 1, 1  FLD x ; Stack: 1, 1, x  FSUBRP ; Stack: 1, x-1  FLD c3 ; Stack: 1, x-1, 3  FDIVP ; Stack: 1, (x-1)/3  FYL2X ; Stack: log2((x-1)/3)  FLDL2E ; Stack: log2((x-1)/3), log2(e)  FDIVP ; Stack: ln((x-1)/3)  FLD c3 ; Stack: ln((x-1)/3), 3  FDIVP ; Stack: ln((x-1)/3)/3  FSTP d  RET  f ENDP  ;------------------------------------------------------------------------------  fp PROC ; dp = f'(x)  ; 1/(x-1)/3 = 1 x 1 - / 3 /  FLD1 ; Stack: 1  FLD x ; Stack: 1, x  FLD1 ; Stack: 1, x, 1  FSUBP ; Stack: 1, x-1  FDIVP ; Stack: 1/(x-1)  FLD c3 ; Stack: 1/(x-1), 3  FDIVP ; Stack: 1/(x-1)/3  FSTP dp  RET  fp ENDP  ;------------------------------------------------------------------------------  @newton\_movable\_pole@12 PROC  PUSH EBP  MOV EBP, [ESP+8]  ; ECX - float eps address  ; EDX - float x address  ; EBP - int iterations address  ; iterations = 0  MOV dword ptr [EBP], 0  ; x = x0  MOV EAX, [EDX]  MOV x, EAX  FINIT  START:  ; d = f(x)  CALL f  ; while (fabs(f(x)) > eps)  FLD dword ptr [ECX] ; push eps  FLD d ; push fabs(f(x))  FABS  FCOMPP ; fabs(f(x)) > eps  FSTSW AX  SAHF  JBE END\_OF\_CYCLE  ; dp = f'(x)  CALL fp  ; x - d/dp/(1 - d/(d - 2)) = x d dp / 1 d d 2 - / - / -  FLD x ; Stack: x  FLD d ; Stack: x, d  FLD dp ; Stack: x, d, dp  FDIVP ; Stack: x, d/dp  FLD1 ; Stack: x, d/dp, 1  FLD d ; Stack: x, d/dp, 1, d  FLD d ; Stack: x, d/dp, 1, d, d  FLD c2 ; Stack: x, d/dp, 1, d, d, 2  FSUBP ; Stack: x, d/dp, 1, d, d - 2  FDIVP ; Stack: x, d/dp, 1, d/(d - 2)  FSUBP ; Stack: x, d/dp, 1 - d/(d - 2)  FDIVP ; Stack: x, d/dp/(1 - d/(d - 2))  FSUBP ; Stack: x - d/dp/(1 - d/(d - 2))  ; x = x - d/(dp - d\*dp/(d-2))  FSTP x  ; iterations++  INC dword ptr [EBP]  JMP START  END\_OF\_CYCLE:  ; внешний x = внутренний x  MOV EAX, x  MOV [EDX], EAX  POP EBP  RET 4  @newton\_movable\_pole@12 ENDP  END | |  | | #include <cstdlib>  #include <iostream>  #include <iomanip>  void \_\_fastcall newton\_movable\_pole(float\* eps, float\* x, int\* iterations);  int main() {  float eps;  std::cout << "This program solves equation ln((x-1)/3)/3=0 from point x0 = 1.5" << std::endl;  std::cout << "Enter epsilon: ";  std::cin >> eps;  float x = 1.5;  int iterations = 0;  newton\_movable\_pole(&eps, &x, &iterations);  std::cout << std::fixed << std::setprecision(8);  std::cout << "Solution: " << x << std::endl;  std::cout << "Iterations count: " << iterations << std::endl;  std::system("pause");  } |  1. Тестирование |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | eps | x | iterations | x\* | |x-x\*| |
| 1 | 10 | 1.5 | 0 | 4 | 2.5 |
| 2 | 1 | 1.5 | 0 | 4 | 2.5 |
| 3 | 0.1 | 3.74081659 | 2 | 4 | 0.25918341 |
| 4 | 0.01 | 3.99219632 | 3 | 4 | 0.00780368 |
| 5 | 0.001 | 3.99219632 | 3 | 4 | 0.00780368 |
| 6 | 0.0001 | 3.99999332 | 4 | 4 | 0.00000668 |
| 7 | 0.0000001 | 4 | 5 | 4 | 0 |