

1. Романчукевич Марина Александровна
2. БПИ216
3. Вариант 26 Разработать программу вычисления корня пятой степени согласно быстро сходящемуся итерационному алгоритму определения корня n -той степени с точностью не хуже 0,1%.
4. Тест 1: 1

```

1 #include <stdio.h>
2
3 double getX(double a);
4
5 double mabs(double x){
6     return (x < 0)? -x : x;
7 }
8
9 int main() {
10     double a;
11     scanf("%lf", &a);
12     int b = (a < 0)? 0 : 1;
13     a = mabs(x: a);
14     double x = getX(a);
15     if(!b){
16         x = -x;
17     }
18     printf("%lf", x);
19
20     return 0;
21 }

```

Run: untitled23

```

/Users/marinaromanchukovich/CLionProjects/untitled23/cmake-build-debug/untitled23
1.000116
Process finished with exit code 0

```

OnlineGDB beta

code, compile, run, debug, share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

189K

GOT AN OPINION? SHARE AND GET REWARDED. ORakuten AIP Have fun taking surveys and get paid! ADS VIA CARBON

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy © 2016 - 2022 GDB Online

```

1 # Эквивалентное представление переменных в программе на C
2 # B main:
3 # rbp[-24] -- `double a`
4 # rbp[-12] -- `int b`
5 # rbp[-40] -- `a` (параметр в mabs)
6 # rbp[-8] -- `double x`
7 # B mabs:
8 # rbp[-8] -- `double x` -- формальный параметр
9 # B L9:
10 # rbp[-40] -- `x`
11 # B getX:
12 # rbp[-24] -- `double a`
13 # rbp[-8] -- `double x0`
14 .intel_syntax noprefix
15 .text
16 .globl mabs
17 .type mabs, @function
18 mabs:
19     push    rbp
20     mov     rbp, rsp
21     movsd   QWORD PTR [-8]rbp, xmm0
22     pxor    xmm0, xmm0
23     ucomisd xmm0, QWORD PTR -8[rbp]
24     jbe     .L7
25     movsd   xmm1, QWORD PTR -8[rbp]
26     movq    xmm0, QWORD PTR .LC1[rip]
27     xorpd   xmm0, xmm1

```

Синтаксис в стиле Intel

Начало секции

Функция mabs

/ Сохраняем rbp на стек

| Вместо rbp записали rsp (rbp := rsp)

`double x`

обнуление xmm0

сравнение 0 и `x`

если 0 меньше или равен, переход к .L7

xmm1 := `x`

копирование 64 бита .LC1 в xmm0

xmm0 ^= xmm1 (теперь xmm0 = -x)

Input

1.000116

...Program finished with exit code 0

Press ENTER to exit console.

Тест 2: 0

The screenshot displays a development environment with three main panels. The top panel shows the source code for `main.c`, which includes a function `getX` and a `main` function that reads a double value, calculates its absolute value, and prints it. The middle panel shows the execution output, indicating the program finished with exit code 0. The bottom panel shows the assembly code generated for the program, with comments in Russian explaining the instructions. The assembly code is for the `main` function and the `mabs` function. The `main` function starts by pushing `rbp` onto the stack, saving it in `rsp`, then pushes `QWORD PTR -8[rbp]` and `xmm0` onto the stack. It then calls `pxor xmm0, xmm0` to clear `xmm0`, followed by `ucomisd xmm0, QWORD PTR -8[rbp]` to compare `xmm0` with the value at `-8[rbp]`. If the value is less than or equal to zero, it jumps to `.L7`. Otherwise, it moves the value from `-8[rbp]` into `xmm1`, shifts it right by 64 bits into `xmm0`, and XORs `xmm0` with `xmm1` to calculate the absolute value. Finally, it prints the result and returns.

```
#include <stdio.h>

double getX(double a);

double mabs(double x){
    return (x < 0)? -x : x;
}

int main() {
    double a;
    scanf("%lf", &a);
    int b = (a < 0)? 0 : 1;
    a = mabs(x: a);
    double x = getX(a);
    if(!b){
        x = -x;
    }
    printf("%lf", x);

    return 0;
}
```

```
Run: untitled23
/Users/marinaronamchukovich/ClionProjects/untitled23/cmake-build-debug/untitled23
0.000000
Process finished with exit code 0
```

```
main.S
1: # Эквивалентное представление переменных в программе на C
2: # B main:
3: # rbp[-24] -- `double a`
4: # rbp[-12] -- `int b`
5: # rbp[-40] -- `a` (параметр в mabs)
6: # rbp[-8] -- `double x`
7: # B mabs:
8: # rbp[-8] -- `double x` -- формальный параметр
9: # B L9:
10: # rbp[-40] -- `x`
11: # B getX:
12: # rbp[-24] -- `double a`
13: # rbp[-8] -- `double x0`
14: .intel_syntax noprefix
15: .text
16: .globl mabs
17: .type mabs, @function
18: mabs:
19: push rbp
20: mov rbp, rsp
21: movsd QWORD PTR -8[rbp], xmm0
22: pxor xmm0, xmm0
23: ucomisd xmm0, QWORD PTR -8[rbp]
24: jbe .L7
25: movsd xmm1, QWORD PTR -8[rbp]
26: movq xmm0, QWORD PTR .LC1[rip]
27: xorpd xmm0, xmm1
input
0
0.000000
...Program finished with exit code 0
Press ENTER to exit console.
```

Тест 3: 5

Project

main.c

1 #include <stdio.h>

2

3 double getX(double a);

4

5 double mabs(double x){

6 return (x < 0)? -x : x;

7 }

8 int main() {

9 double a;

10 scanf("%lf", &a);

11 int b = (a < 0)? 0 : 1;

12 a = mabs(x: a);

13 double x = getX(a);

14 if(!b){

15 x = -x;

16 }

17 printf("%lf", x);

18

19 return 0;

20 }

main

Run: untitled23

/Users/marinanoromanchukovich/CLionProjects/untitled23/cmake-build-debug/untitled23

1.379732

Process finished with exit code 0

Version Control

Run

TODO

Problems

Terminal

Python Packages

CMake

Messages

Event Log

Process finished with exit code 0

5:1 LF UTF-8 4 spaces C:untitled23 | Debug

onlinegdb.com

Run Debug Stop Share Save Beauty

Language Assembly

main.S

1 # Эквивалентное представление переменных в программе на C

2 # B main:

3 # rbp[-24] -- `double a`

4 # rbp[-12] -- `int b`

5 # rbp[-40] -- `a` (параметр в mabs)

6 # rbp[-8] -- `double x`

7 # B mabs:

8 # rbp[-8] -- `double x` -- формальный параметр

9 # B L9:

10 # rbp[-40] -- `x`

11 # B getX:

12 # rbp[-24] -- `double a`

13 # rbp[-8] -- `double x0`

14 .intel_syntax noprefix

15 .text

16 .globl mabs

17 .type mabs, @function

18 mabs:

19 push rbp

20 mov rbp, rsp

21 movsd QWORD PTR [-8]rbp, xmm0

22 pxor xmm0, xmm0

23 ucomisd xmm0, QWORD PTR -8[rbp]

24 jbe .L7

25 movsd xmm1, QWORD PTR -8[rbp]

26 movq xmm0, QWORD PTR .LC1[rip]

27 xorpd xmm0, xmm1

Синтаксис в стиле Intel

Начало секции

Функция mabs

/ Сохраняем rbp на стек

| Вместо rbp записали rsp (rbp := rsp)

`double x`

обнуление xmm0

сравнение 0 и `x`

если 0 меньше или равен, переход к .L7

xmm1 := `x`

копирование 64 бита .LC1 в xmm0

xmm0 ^= xmm1 (теперь xmm0 = -x)

input

5

1.379732

...Program finished with exit code 0

Press ENTER to exit console.

About

FAQ

Blog

Terms of Use

Contact Us

GDB Tutorial

Credits

Privacy

© 2016 - 2022 GDB Online

Тест 4: 122

The image shows a development environment with three main panels. The top panel displays C code for a program named 'main.c'. The code defines a function 'mabs' that calculates the absolute value of a double 'x' and a function 'getX' that iteratively refines the absolute value of a double 'a' using the Newton-Raphson method. The bottom-left panel shows the execution output of the program, which prints the value '2.613801' and indicates that the process finished with exit code 0. The bottom-right panel shows the assembly code generated for the C program, with comments explaining the instructions in Russian. The assembly code is for the 'main' function and includes instructions for stack frame setup, parameter passing, and the iterative calculation in 'getX'.

```
11 int b = (a < 0)? 0 : 1;
12 a = mabs( x: a);
13 double x = getX(a);
14 if(!b){
15     x = -x;
16 }
17 printf("%lf", x);
18
19 return 0;
20
21
22 double getX(double a) {
23     double x0 = a/5;
24     while(mabs( x: a - x0 * x0 * x0 * x0 * x0 ) >= 0.001){
25         x0 = 0.2 * (4 * x0 + a / (x0 * x0 * x0 * x0));
26     }
27     return x0;
28 }
29
```

Run: untitled23
/Users/marinaronamchukovich/ClionProjects/untitled23/cmake-build-debug/untitled23
2.613801
Process finished with exit code 0

main.S

```
1 # Эквивалентное представление переменных в программе на C
2 # B main:
3 # rbp[-24] -- `double a`
4 # rbp[-12] -- `int b`
5 # rbp[-40] -- `a` (параметр в mabs)
6 # rbp[-8] -- `double x`
7 # B mabs:
8 # rbp[-8] -- `double x` -- формальный параметр
9 # B L9:
10 # rbp[-40] -- `x`
11 # B getX:
12 # rbp[-24] -- `double a`
13 # rbp[-8] -- `double x0`
14 .intel_syntax noprefix
15 .text
16 .globl mabs
17 .type mabs, @function
18 mabs:
19     push    rbp
20     mov     rbp, rsp
21     movsd   QWORD PTR [-8[rbp], xmm0
22     pxor    xmm0, xmm0
23     ucomisd xmm0, QWORD PTR -8[rbp]
24     jbe     .L7
25     movsd   xmm1, QWORD PTR -8[rbp]
26     movq    xmm0, QWORD PTR .LC1[rip]
27     xorpd   xmm0, xmm1
28     input
29
```

122
2.613801
...Program finished with exit code 0
Press ENTER to exit console.

Тест 5: -32

The screenshot displays a development environment with three main panels. The top panel shows the C source code for a program named 'main.c'. The code defines a function 'mabs' that returns the absolute value of a double 'x', and a 'main' function that reads a double from standard input, applies 'mabs', and prints the result. The middle panel shows the program's execution output, where the input '-2.000012' is read and the output '-2.000012' is printed. The bottom panel shows the assembly code generated for the C program, with comments explaining the instructions in Russian. The assembly includes stack frame setup, argument passing, function calls, and arithmetic operations to calculate the absolute value.

```
double getX(double a);
double mabs(double x){
    return (x < 0)? -x : x;
}
int main() {
    double a;
    scanf("%lf", &a);
    int b = (a < 0)? 0 : 1;
    a = mabs(x: a);
    double x = getX(a);
    if(!b){
        x = -x;
    }
    printf("%lf", x);
    return 0;
}
double getX(double a) {
    main
}
```

```
Run: untitled23
/Users/marinaronamchukovich/ClionProjects/untitled23/cmake-build-debug/untitled23
-2.000012
Process finished with exit code 0
```

```
main.S
1 # Эквивалентное представление переменных в программе на C
2 # B main:
3 # rbp[-24] -- `double a`
4 # rbp[-12] -- `int b`
5 # rbp[-40] -- `a` (параметр в mabs)
6 # rbp[-8] -- `double x`
7 # B mabs:
8 # rbp[-8] -- `double x` -- формальный параметр
9 # B L9:
10 # rbp[-40] -- `x`
11 # B getX:
12 # rbp[-24] -- `double a`
13 # rbp[-8] -- `double x0`
14 .intel_syntax noprefix
15 .text
16 .globl mabs
17 .type mabs, @function
18 mabs:
19     push    rbp
20     mov     rbp, rsp
21     movsd   QWORD PTR [-8[rbp]], xmm0
22     pxor    xmm0, xmm0
23     ucomisd xmm0, QWORD PTR -8[rbp]
24     jbe     .L7
25     movsd   xmm1, QWORD PTR -8[rbp]
26     movq    xmm0, QWORD PTR .LC1[rip]
27     xorpd   xmm0, xmm1
    input
-32
-2.000012
...Program finished with exit code 0
Press ENTER to exit console.
```

Выполнено в соответствии с критериями на 5 баллов

- Приведено решение задачи на C на планируемую оценку. Ввод данных осуществляется с клавиатуры. Вывод данных осуществляется на дисплей. Файл idz3.c на гитхаб
- В полученную ассемблерную программу, откомпилированную без оптимизирующих и отладочных опций, добавлены комментарии, поясняющие эквивалентное представление переменных в программе на C. То есть, для всех

ссылок на память, включая и относительные адреса и регистры, указать имя переменной на языке С исходной программы.

7. • Из ассемблерной программы убраны лишние макросы за счет использования при компиляции из С соответствующих аргументов командной строки и/или за счет ручного редактирования исходного текста ассемблерной программы.
`gcc -masm=intel -fno-asynchronous-unwind-tables -fno-jump-tables -fno-stack-protector -fno-exceptions -S 8.c`
8. • Модифицированная ассемблерная программа отдельно откомпилирована и скомпонована без использования опций отладки.
9. • Представлено полное тестовое покрытие, дающее одинаковый результат на обеих программах. Приведены результаты тестовых прогонов для обеих программ, демонстрирующие эквивалентность функционирования.
 - Для сопоставления с полученной ассемблерной программой необходимо также приложить исходные тексты на ассемблере, сформированные компилятором языка С. Файл `idz3noComm.s` на гитхаб
10. • Сформирован отчет с результатами тестовых прогонов и описанием используемых опций компиляции, проведенных модификаций ассемблерной программы.
11. • В программе на языке С необходимо использовать функции с передачей данных через формальные параметры.
Функция `getX` и `mabs`
12. • Внутри функций необходимо использовать локальные переменные, которые при компиляции отображаются на стек.
Функция `getX`
13. • В ассемблерную программу в местах вызова функции добавить комментарии, описывающие передачу фактических параметров и перенос возвращаемого результата. При этом необходимо отметить, какая переменная или результат какого выражения соответствует тому или иному фактическому параметру.
Файл `idz3.s` на гитхаб
14. • В ассемблерных функциях для каждого формального параметра необходимо добавить комментарии, описывающие связь между именами формальных параметров на языке С и регистрами (или значением на стеке), через которые эти параметры передаются.
15. • Для сопоставления с полученной ассемблерной программой необходимо также приложить исходные тексты на ассемблере, сформированные компилятором языка С.
16. • Информацию о проведенных изменениях отобразить в отчете наряду с информацией, необходимой на предыдущую оценку.