Правительство Российской Федерации

Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук

Департамент программной инженерии

Минипроект

по теме «Программа интегрирования функции методом Симпсона»

Выполнил

Студент группы БПИ193

Минец Максим

mvminets@edu.hse.ru

Москва 2020

Оглавление

1 Задание

Разработать программу интегрирования функции $y = a + bx^{-2}$ (задается двумя числами a, b) в заданном диапазоне (задается так же) методом Симпсона (использовать FPU).

2 Применяемые расчетные методы

2.1 Расчет значения

Формулой Симпсона называется интеграл от интерполяционного многочлена второй степени на отрезке [a,b]:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$

где $f(x)$ – это исходная функция.

2.2 Программа и компиляция

По причине того, что использование fasm затруднительно, для налипания был выбран NASM.

Netwide Assembler -свободный ассемблер для архитектуры Intel x86, обладающий более широкими возможностями, чем fasm.

К программе (файл source.asm) прилагается отчет (report.pdf, данный файл), файл для компиляции (compile.bat) и файлы для запуска программы с некоторыми параметрами (run_test1.bat), а также шаблон для показа подсказки (show_help.bat), а также скриншоты выполнения самой программы. При этом в файле compile.bat используются вызовы программ nasm (получение объектного файла из 32-битной программы) и ld (линковка полученного объектного файла, стандартной библиотеки C в Windows msvrct.dll, библиотеки kernel32.dll для работы с командной строкой), а также файла user32.dll для вызова ASCII-версии функции MessageBox. Также, для того, чтобы основная функция могла называться main, линковщику также подается соответствующая команда. Последней командой происходит запуск полученного после линковки файла.

2.3 Документация

Согласно условию, программа может выводить подсказку. Если запустить программу с аргументами командной строки, которые заканчиваются на вопросительный знак «?», то программа выведет небольшую памятку о том, какие числа ожидаются на вход.

2.4 Общая структура

Программа использует декларацию cdecl для вызова функций. Таким образом, аргументы функций передаются через стек (первым аргументам соответствуют младшие адреса в стеке), при этом если функция возвращает число с плавающей точкой, оно возвращается через стек FPU. Функция simpson также использует 8 байт локальной памяти, которые последовательно используются как для хранения целых чисел (старшие 4 байта), так и для хранения чисел типа double (используется, как буфер при передаче значения из стека FPU и стандартный стек). В основной функции main также используются метки about и exit для функционирования подсказки.

Для чтения программы рекомендуется установить ширину табулирования равной 8

пробелами, так как комментарии выровнены при такой настройке текстового редактора.

В программе объявлены следующие макросы:

1. print_string - принимает на вход адрес строки, выводит ее в консоль

2. fcall – записывает в нижние 8 байт текущего стекового фрейма число из вершины стека FPU, после чего копирует это значение на вершину стека и вызывают функцию, таким образом применяя функцию в число, которое было на вершине стека FPU.

Использование данных макросов значительно упрощает код в программе.
Программа использует следующие функции:

1. func – принимает единственные значение, возвращает значение исходной функции в данной точке
2. simpson – не принимает аргументов, возвращает вычисленный методом Симпсона определенный интеграл
3. main – основная функция программы, в которой происходит ввод и вывод значений

Нужно отметить, что также используются глобальные переменные a, b (константы, подставляемые в функцию), x1, x2 (пределы интегрирования), command (адрес аргументов командной строки), а также answer (строка, в которую записывается ответ в конце работы программы). Эти переменные используются, так как при передаче всех этих значений по стеку согласно декларации cdecl и использование локальной стековой памяти значительно бы усложнило логику программы.

2.5 Формат ввода-вывода

Формат ввода:
Программа ожидает на вход 4 рациональных 8-байтовых числа: a,b (параметры функций), x1 и x2 (пределы интегрирования, при этом $x1 \leq x2$).

Формат вывода:
После вычислений программа выводит окно с результатом вычислений.

3 Текст программы

```
extern _printf                  ; Функция вывода в консоли
extern _sprintf, _sscanf        ; Функции ввода-вывода из строки
extern _strlen                  ; Функция подсчета длины строки
extern _MessageBoxA             ; Высплывающее окно
extern _GetCommandLineA         ; Функция для работы с аргументами командной строки

global main

%macro print_string 1           ; Макрос выводит строку по данному адресу в консоль
        push %1
        push printf_s
        call _printf
        add esp, 8
%endmacro

%macro fcall 0                  ; Использует первую локальную переменную в текущем фрейме,
        fstp qword [ebp - 8]    ; чтобы применить функцию к аргументу на вершине стека FPU
        push dword [ebp - 4]
        push dword [ebp - 8]
        call func
        add esp, 8
%endmacro

section .data
        args_f: db "%*s %lf %lf %lf %lf", 0
        scanf_d: db "%lf", 0    ; Форматирующая строка для ввода рационального числа
        printf_d_answer: db "The defined integral of %.2f + %.2f / (x^2), x from %.2f to %.2f is %.4f", 0
        printf_s: db "%s", 0    ; Форматирующая строка для вывода строки
        title: db "Answer", 0   ; Заголовок окна вывода
        about_message: db "Input four numbers to calculate integral:", 10, " a, b", 9, 9, "Function
parameters", 10, " x1, x2", 9, 9, "Limits of integration", 10, 0
        error_message: db "x1 should be less than x2; 0 is out of function range", 10, 0

section .bss                    ; Часто используемые значения вынесены в глобальные
                                ; переменные
        a: resb 8               ; Это позволяет не передавать их по стеку
        b: resb 8
        x1: resb 8
        x2: resb 8
        c: resb 8               ; Ответ
        command: resb 4         ; Адрес аргументов командной строки
        answer: resb 64         ; Буфер, в котором будет храниться ответ

section .text
func:                           ; Принимает единственный параметр -- аргумент функции
        enter 0, 0              ; Функция не использует локальных переменных
        fld qword [ebp + 8]     ; st0 = x
        fmul st0                ; st0 = x^2
        fdivr qword [b]         ; st0 = b / x^2
        fadd qword [a]          ; st0 = a + b / x^2
        leave                   ; Возвращаем регистр ebp
```

```
        ret                     ; Согласно cdecl, число типа double возвращается в стеке FPU

simpson:                        ; Функция вычисляет интеграл, не имеет аргументов
        enter 8, 0              ; Резервируем 8 байт (для типа double)
        fld qword [x1]          ; st0 = x1 (Здесь и далее описывается стек FPU)
        fadd qword [x2]         ; st0 = x1 + x2
        mov dword [ebp - 4], 2
        fidiv dword [ebp - 4]   ; st0 = (x1 + x2) / 2
        fcall                   ; st0 = f((x1 + x2) / 2)
        mov dword [ebp - 4], 4
        fimul dword [ebp - 4]   ; st0 = 4 * f((x1 + x2) / 2)
        fld qword [x1]          ; st0 = x1; st1 = 4 * f((x1 + x2) / 2)
        fcall                   ; st0 = f(x1); st1 = 4 * f((x1 + x2) / 2)
        fadd st1                ; st0 = f(x1) + 4 * f((x1 + x2) / 2)
        fld qword [x2]          ; st0 = x2; st1 = f(x1) + 4 * f((x1 + x2) / 2)
        fcall                   ; st0 = f(x2); st1 = f(x1) + 4 * f((x1 + x2) / 2)
        fadd st1                ; st0 = A = f(x1) + 4 * f((x1 + x2) / 2) + f(x2)
        fld qword [x2]          ; st0 = x2; st1 = A
        fsub qword [x1]         ; st0 = x2 - x1; st1 = A
        mov dword [ebp - 4], 6
        fidiv dword [ebp - 4]   ; st0 = (x2 - x1) / 6; st1 = A
        fmul st1                ; st0 = (x2 - x1) / 6 * A
        leave
        ret

main:
        call _GetCommandLineA       ; Адрес на аргументы записаны в eax
        mov [command], eax

        push eax
        call _strlen            ; Вычисляем длину строки аргументов
        add esp, 4

        mov ebx, [command]
        mov al, [eax + ebx - 1] ; Записываем последний символ аргументов
        cmp eax, 63             ; 63 - это код символа '?'
        je about

        push x2                 ; Считываем из консоли все данные
        push x1
        push b
        push a
        push args_f
        push dword [command]
        call _sscanf
        add esp, 24

        fld qword [x1]          ; x1 должен быть меньше x2
        fld qword [x2]
        fcomi st1
        jbe error

        fldz
```

```
        fld qword [x1]
        fcomi st1
        je error
        jb assert_less
        ja assert_greater

assert_less:
        fldz
        fld qword [x2]
        fcomi st1
        jae error
        jmp assert_end
assert_greater:
        fldz
        fld qword [x2]
        fcomi st1
        jbe error
        jmp assert_end
assert_end:

        finit
        call simpson            ; Решаем задачу, ответ находится на вершине стека FPU
        fstp qword [c]

        push dword [c + 4]      ; Заносим в стек верхние 4 байта ответа
        push dword [c]          ; Заносим в стек нижние 4 байта ответа
        push dword [x2 + 4]
        push dword [x2]
        push dword [x1 + 4]
        push dword [x1]
        push dword [b + 4]
        push dword [b]
        push dword [a + 4]
        push dword [a]
        push printf_d_answer    ; Заносим в стек формат для вывода ответа
        push answer             ; Заносим в стек адрес буфера, в который будет записан ответ
        call _sprintf           ; Вызываем функцию C для записи в буфер
        add esp, 48             ; Поправляем стек

        push 0                  ; Равно константе MB_OK из WinAPI, т.е. единственная кнопка OK
        push title              ; Заголовок окна
        push answer             ; Адрес строки с ответом
        push 0                  ; Нулевой указатель, у сообщения нет родительского окна
        call _MessageBoxA       ; Показывает окно с ответом
        add esp, 16             ; Поправляем стек

        jmp exit

about:
        print_string about_message
        jmp exit

error:
```

```
        print_string error_message
        jmp exit

exit:
        ret
```