Национальный исследовательский университет “Высшая Школа Экономики”,
Факультет компьютерных наук
Департамент программной инженерии

«Программа по нахождению наименьшего числа с плавающей точкой»

Пояснительная записка к разработке консольного приложения

Исполнитель:
Студент группы БПИ199
Волохов Никита Алексеевич

Оглавление

1. Текст задания

Формулировка задания: «Разработать программу, определяющую наименьшее число с плавающей точкой методом дихотомии и сравнения с единицей» [1].

2. Применяемые расчетные методы

a. Считывание количества вводимых чисел с плавающей точкой

При запуске программы выводится сообщение ***strNumOfFPUs*** с просьбой ввести число вводимых чисел с плавающей точкой.

Число вводимых чисел с плавающей точкой, сохраняемое в переменную ***num_of_fpus***, считывается из консоли и должно быть целым числом в диапазоне [1; 100] (включая границы диапазона). Для считывания числа используется функция **scanf** строка ‘%d’ (переменная ***strScanInt***).

При некорректном вводе данного числа в консоль выводится сообщение об ошибке ***strIncorNum***.

b. Считывание чисел с плавающей точкой и мгновенная проверка на минимальность введенного числа

Для работы с числами с плавающими точками был использован модуль операций с плавающей точкой FPU [2].

Число с плавающей точкой, сохраняемое в переменную ***temp_fval***, считывается из консоли и должно быть формата *«{целая_часть}.{дробная_часть}»* или *«{целая_часть}»*. Для считывания числа используется функция **scanf** строка **‘%lf’** (переменная ***strScanFloat***). Используется не строка **‘%f’**, чтобы функция **scanf** могла работать с 64-разрядными числами.

Первое считанное число с плавающей точкой сразу задвигается в вершину стека регистров FPU [2] st0, чтобы в последствии брать это самое первое значение стека и сравнивать с ним остальные числа с плавающей точкой.

Если ***num_of_fpus*** не равно 1, то остальные числа таким же образом считываются в цикле длины (***num_of_fpus – 1)***. Сразу же после считывания выполняется проверка, меньше ли значение ***num_of_fpus*** значения на вершине стека FPU [2] st0. Если это так, то значение ***num_of_fpus*** записывается в вершину стека FPU [2] st0. Иначе цикл продолжает выполняться.

По завершении цикла берется значение из вершины стека FPU [2] st0 и сохраняется в переменную ***min_fval***.

c. Вывод результата

Для вывода используется функция **printf** и строка ***strMinFloat***, куда подставляется значение переменной ***min_fval***.

3. Тестовые примеры

Программа корректно работает на корректных входных данных (см. Рисунок 1.1, 1.2)

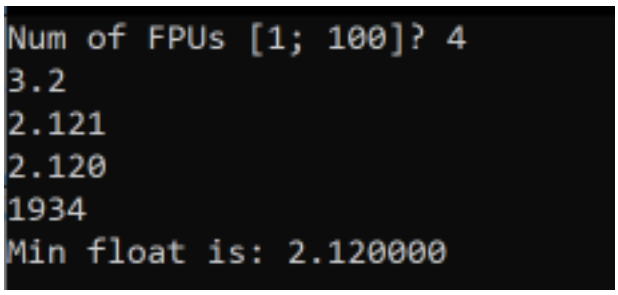

Рисунок 1. Корректные входные данные. Пример 1

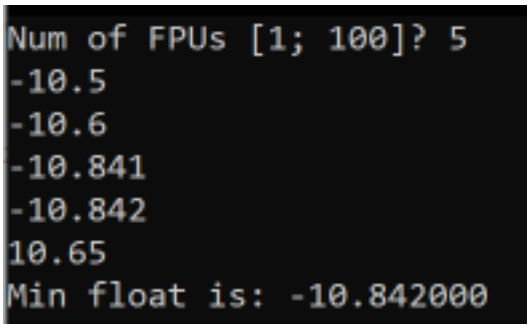

Рисунок 1.2. Корректные входные данные. Пример 2

Выводит сообщение об ошибке при некорректном вводе количества чисел с плавающей точкой (см. Рисунок 2.1, 2.2)

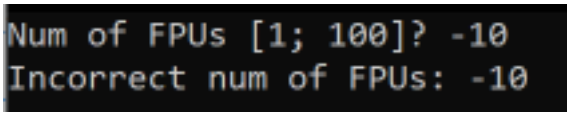

Рисунок 2.1. Количество чисел с плавающей точкой меньше разрешенного диапазона

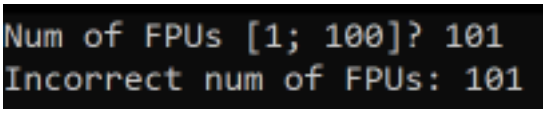

Рисунок 2.2. Количество чисел с плавающей точкой больше разрешенного диапазона

4. Список используемых источников

[1] Программирование на языке ассемблера. Микропроект. Требования к оформлению. 2020-2021 уч.г. [Электронный ресурс]. // URL: http://softcraft.ru/edu/comparch/tasks/mp01/ (Дата обращения: 31.10.2020, режим доступа: свободный)

[2] FLAT ASSEMBLER 1.64 - МАНУАЛ ПРОГРАММЕРА [Электронный ресурс]. // URL: http://flatassembler.narod.ru/fasm.htm#2-1-13 (Дата обращения: 30.10.2020, режим доступа: свободный)

5. Текст программы

```
; БПИ199
; Волохов Никита Алексеевич
; Вариант 5
; "Разработать программу, определяющую наименьшее число с плавающей
точкой
; методом дихотомии и сравнения с единицей"
format PE console
entry start

include 'win32a.inc'

section '.data' data readable writable

        strNumOfFPNs  db 'Num of FPNs [1; 100]? ', 0
        strMinFloat   db 'Min float is: %lf', 10, 0
        strIncorNum   db 'Incorrect num of FPNs: %d', 10, 0
        strScanInt    db '%d', 0
        strScanFloat  db '%lf', 0

        num_of_fpns   dd 0
        temp_fval     dq ?
        min_fval      dq ?
        i             dd ?

section '.code' code readable executable

start:
; 1) num of FPNs input
        push strNumOfFPNs
        call [printf]

        push num_of_fpns
        push strScanInt
        call [scanf]

        ; check if num entered is correct
        cmp [num_of_fpns], 1
        jl incorrNumOfFPNs

        cmp [num_of_fpns], 100
        jg incorrNumOfFPNs

        jmp firstFPN

incorrNumOfFPNs:
        push [num_of_fpns]
        push strIncorNum
        call [printf]
        jmp finish
```

```
; 2) save first FPN to st0 to compare it with other FPNs
firstFPN:
        FINIT ; coprocessor init

        invoke scanf, strScanFloat, temp_fval

        fld [temp_fval] ; push temp_fval to stack (st0)

; 3) input other (num_of_fpns - 1) FPNs and compare it with st1 FPN
preFPNLoop:
        cmp [num_of_fpns], 1
        je endFPNLoop ; we already have 1 num. It is the answer

        sub [num_of_fpns], 1
        xor ecx, ecx ; ecx = 0

FPNLoop:
        mov [i], ecx

        cmp ecx, [num_of_fpns]
        je endFPNLoop ; to end loop

        invoke scanf, strScanFloat, temp_fval

        fcom [temp_fval] ; compare with FPN in st0
        fstsw ax
        sahf
        ja newStVal

        inc [i]
        mov ecx, [i]
        jmp FPNLoop

newStVal:
        fld [temp_fval] ; push temp_fval to stack (st0)

        inc [i]
        mov ecx, [i]
        jmp FPNLoop

endFPNLoop:
        fst [min_fval] ; move st0 FPN to min_fval var

        invoke printf, strMinFloat, dword[min_fval], dword[min_fval +
4]

; 4) programm end
finish:
        call [getch]
```

```
        push 0
        call [ExitProcess]


section '.idata' import data readable
library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll'

include 'api\kernel32.inc'
    import kernel,\
           ExitProcess, 'ExitProcess'

include 'api\kernel32.inc'
    import msvcrt,\
           printf, 'printf',\
           scanf, 'scanf',\
           getch, '_getch'
```