ПРОГРАММА ДЛЯ ОПРЕДЕЛЕНИЯ НАИМЕНЬШЕГО ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ МЕТОДОМ ДИХОТОМИИ И СРАВНЕНИЯ С ЕДИНИЦЕЙ

Пояснительная записка

Исполнитель
студентка группы БПИ196
Д.В. Еремина
1 ноября 2020 г.

Москва 2020

СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ

Разработать программу, определяющую наименьшее число с плавающей точкой методом дихотомии и сравнения с единицей

2. МЕТОДЫ

2.1. Методы для произведения расчетов

2.1.1. Метод для получения середины текущего отрезка

```
macro GetNext {         ; finds midpoint of [left, right] and places result in tmp
    fld [left]        ; move left-value to st(0)
    fadd [right]      ; add right to value in st(0)
    fdiv [half]       ; same with divison
    fstp  [tmp]       ; moves result to tmp in cleares st(0)
}
```

2.1.2. Метод, сдвигающий границу поиска влево по принципу деления отрезка пополам

```
macro MoveRight {       ; moves upper limit of segment
    ; the same as with GetNext but result is in right
    fld [left]
    fadd [right]
    fdiv [half]
    fstp  [right]
}
```

2.1.3. Метод для сравнения с единицей

```
macro CompareToOne {    ; compares 1 + right to 1
    fld [one]
    fadd [right]
    fcomp [one]
    fstsw ax          ; copy the Status Word containing the result to AX
    fwait             ; insure the previous instruction is completed
    sahf              ; transfer the condition codes to the CPU's flag register
}
```

2.2. Методы вывода в консоль

2.2.1. Метод для вывода сообщения о завершении работы

```
macro PrintExit {       ; prints string before exit
    push strForExit
    call [printf]
}
```

3. МЕТКИ В ОСНОВНОЙ ПРОГРАММЕ

3.1. Метка цикла

```
lp:
        MoveRight       ; move upper limit

        CompareToOne    ; compare 1 + right to 1

        ja makePrev     ; if 1 + right is greater than 1, we just move upper limit
        jz compareToPrev  ; if 1 + right equals 1, we compare current right-value with previous and next to keep looking for exact epsilon
```

3.2. Метка, при переходе к которой происходит сохранение значения предыдущей итерации

```
; move value from right to prev and continue loop
makePrev:
    fld [right]
    fstp [prev]
    jmp lp
```

3.3. Метка, при переходе к которой происходит сравнение значений предыдущей, текущей и последующей итераций

```
; compare values we got on previous iteration, current value and next
compareToPrev:
    GetNext
    fld [tmp]
    fcomp [right]
    fstsw ax        ; copy the Status Word containing the result to AX
    fwait           ; insure the previous instruction is completed
    sahf            ; transfer the condition codes to the CPU's flag register

    ; if next value will be less, we continue
    jb makePrev
    ; if we reached epsilon and values dont change, we finish
    jz finish
```

4. ОПИСАНИЕ АЛГОРИТМА

Наименьшим вещественным числом является так называемый «машинный ноль», который предлагается искать методом дихотомии и сравнения с единицей. Так как очевидно, что данное число находится в диапазоне от 0 до 1, его поиск начинает производиться в этих границах. Каждый раз происходит попытка деления на два наибольшего значения (использование метода дихотомии). Необходимым условием машинного нуля можно считать выполнения равенства 1 + x = 1 при некотором достаточно малом x, поэтому после сдвига границы происходит подобная проверка (сравнение с единицей). Если полученное значение больше единицы, граница сразу сдвигается еще ближе к нулю. Если же значения уже расцениваются как равные, то происходит сравнение достигнутого результата на предыдущем шаге, текущего результата и возможного на следующем шаге. В том случае, если более сильного приближения достичь уже невозможно, программа завершает работу, выводя на экран полученное значение в научном формате. Также выводится погрешность относительно реального значения (оно указывается как константа в данных программы).

ПРИЛОЖЕНИЕ 1. КОД ОСНОВНОЙ ПРОГРАММЫ

```
format PE console
entry start

include 'win32a.inc'
include 'microLib.inc'  ; macros file
;--------------------------------------------------------------------------------------------
section '.data' data readable writable
        ; strings for output
        strIntro db "Searching for machine epsilon in range [%.1f, %.1f]", 10, 0
        strAns db "Result is: %.15e", 10, 0
        strDiff db "Difference between found value and actual machine epsilon is: %.15e", 10, 0

        strForExit  db '> Press any key to exit', 10, 0

        left  dq 0.0    ; initialising lower limit of segment
        right dq 1.0    ; initialising upper limit of segment

        one dq 1.0      ; const value
        half dq 2.0     ; const value
        prev dq 2.0     ; saving previous found value
        tmp dq ?        ; to save some temporary values

        actualEpsilon dq 4.94065645841247e-324  ; actual epsilon to check parity of calculations
;--------------------------------------------------------------------------------------------
section '.code' code readable executable
start:
        invoke printf, strIntro, dword[left], \
                dword[left+4], dword[right], dword[right+4]

        finit                   ; initialise  coprocessor
        lp:
                MoveRight       ; move upper limit

                CompareToOne    ; compare 1 + right to 1

                ja makePrev     ; if 1 + right is greater than 1, we just move upper limit
                jz compareToPrev    ; if 1 + right equals 1, we compare current right-value with
previous and next to keep looking for exact epsilon


finish:
        ; needed value is in prev
        invoke printf, strAns, dword[prev], \
                dword[prev+4]

        ; find diference between found epsilon and actual
        fld [prev]
        fsub [actualEpsilon]
        fstp [tmp]
```

```
        invoke printf, strDiff, dword[tmp], \
                dword[tmp+4]

        PrintExit
        call [getch]

        push 0
        call [ExitProcess]

; move value from right to prev and continue loop
makePrev:
        fld [right]
        fstp [prev]
        jmp lp

; compare values we got on previous iteration, current value and next
compareToPrev:
        GetNext
        fld [tmp]
        fcomp [right]
        fstsw ax         ; copy the Status Word containing the result to AX
        fwait            ; insure the previous instruction is completed
        sahf             ; transfer the condition codes to the CPU's flag register

        ; if next value will be less, we continue
        jb makePrev
        ; if we reached epsilon and values dont change, we finish
        jz finish
;--------------------------------------------------------------------------------------------------
section '.idata' import data readable
library kernel, 'kernel32.dll',\
          msvcrt, 'msvcrt.dll',\
          user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
   import kernel,\
         ExitProcess, 'ExitProcess',\
         HeapCreate,'HeapCreate',\
         HeapAlloc,'HeapAlloc'
   include 'api\kernel32.inc'
   import msvcrt,\
         printf, 'printf',\
         sprintf, 'sprintf',\
         scanf, 'scanf',\
         getch, '_getch'
```

ПРИЛОЖЕНИЕ 2. КОД ИМПОРТИРУЕМОГО ФАЙЛА microLib.inc

```
macro GetNext {         ; finds midpoint of [left, right] and places result in tmp
    fld [left]        ; move left-value to st(0)
    fadd [right]      ; add right to value in st(0)
    fdiv [half]       ; same with divison
    fstp  [tmp]       ; moves result to tmp in cleares st(0)
}

macro MoveRight {       ; moves upper limit of segment
    ; the same as with GetNext but result is in right
    fld [left]
    fadd [right]
    fdiv [half]
    fstp  [right]
}

macro CompareToOne {    ; compares 1 + right to 1
    fld [one]
    fadd [right]
    fcomp [one]
    fstsw ax          ; copy the Status Word containing the result to AX
    fwait             ; insure the previous instruction is completed
    sahf              ; transfer the condition codes to the CPU's flag register
}

macro PrintExit {       ; prints string before exit
    push strForExit
    call [printf]
}
```