НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Архитектура вычислительных систем»

Программа для нахождения пар параллельных отрезков, заданных координатами концевых точек

Пояснительная записка

Вариант 9

Выполнил: Гарифуллин Руслан Ильфатович

студент БПИ191, ФКН Программная инженерия

Москва 2020

1. Текст задания

Разработать программу, которая решает вопрос о нахождении пар параллельных отрезков из общего числа N=5 отрезков, заданных координатами концевых точек.

Программа должна быть выполнения на языке Ассемблер и быть скомпилирована при помощи кроссплатформенного компилятора flat assembler.

2. Описание расчетных методов

Программа использует средства сопроцессора для работы с числами с плавающей точкой (FPU). Во время выполнения задачи потребовались следующие своиства из тригонометрии и векторной алгебры:

1. Формула нахождения направляющего вектора прямой:

$$\vec{a} = (x_2 - x_1,\ y_2 - y_1)$$

2. Формула нахождения тангенса угла наклона прямой:

$$tg(\vec{a}) = \frac{y_a}{x_a}$$

3. Свойство параллельности прямых:

$$tg(\vec{a}) = tg(\vec{b}) \Rightarrow a \| b$$

3. Описание входных данных

На вход подаются четыре числа для каждой прямой – координаты точек концов отрезка, составляющего прямую. Формат: **x1 y1 x2 y2**, где **x1 y1 –** координаты одного конца отрезка, **x2 y2** – координаты другого конца отрезка.

Если координаты двух точек одинаковы (вместо отрезка образуется точка), программа сообщает об этом и предлагает ввести другие значения.

```
Line 0: 3 4 4 5
Line 1: 4 4 4 4
You have given a point. Line expected.
Line 1: 4 5 5 6
Line 2: 7 8 1 4
Line 3: 10 -6 3 4
Line 4: 5 7 7 9
```

Пример входных данных

Диапазон допустимых значений: координаты точек обязаны помещаться в двойное слово, то есть принадлежать диапазону $[-2^{31}, 2^{31})$, в противном случае происходит переполнение. Допускается и ввод дробных чисел, однако точность расчётов не гарантируется (их использование условием задачи не предполагалось).

4. Описание выходных данных

Вывод программы представляет собой текст в консоли, где в каждой строке описывается, какие прямые параллельны между собой:

```
Line #1 is parallel to line #2.
Line #1 is parallel to line #3.
Line #2 is parallel to line #3.
```

Пример выходных данных

5. Описание ключевых переменных

Переменные **x1**, **y1**, **x2**, **y2** (тип DWORD) отвечают за временное хранение значений координат, введённых с консоли. Переменная **LINES** (тип DWORD, размер **N**) хранит в себе тангенсы углов наклона прямых. Возможно изменение значение константы **N** в коде.

6. Дополнительные библиотеки

Были использованы следующие функции и стандартной библиотеки Си (libc6) с соответствующими обёртками в виде макросов:

- printf
- scanf

Приложение 1

Код программы

Код программы представлен в репозитории на сервисе GitHub: https://github.com/ruslang02/HSE-FASM-Projects/tree/master/project01.

main.asm:

```
format ELF64
public main

N EQU 5; Number of lines to compare.
WORD_SIZE EQU 4; dd size

include 'io.asm'; I/O macros

section '.text' writable
main:
      print DESCRIPTION, N
      call input_lines
      call process_lines

      mov rax, 1
      int 80h

input_lines:
      mov rbx, LINES
      mov rcx, N
      .next:
      mov rdx, N
      sub rdx, rcx
      print LINE_INPUT, rdx; input lines data
      ; using scanf resets all registers
      read LINE_INPUT_FORMAT, x1, y1, x2, y2
      call count_tg
      cmp al, byte 1; NaN check
      jne .continue
      .error:
            call print_error
            jmp .next
      .continue:
            fstp dword [rbx + rdx * WORD_SIZE]; load angle tan into array
            loop .next
      ret
print_error:
      print POINT_PROVIDED_ERROR
      ret

; output: ax - status flags
```

```
count_tg:
      finit; reset FPU state
      ; <tg A = (y2 - y1) / (x2 - x1)>
      fld dword [x2]
      fsub dword [x1]
      fld dword [y2]
      fsub dword [y1]
      fdivrp; reverse divide
      ; </tg A>
      fstsw ax; load status flags into AX
      ret

process_lines:
      mov rbx, LINES; save array pointer into register, otherwise errors
      mov rcx, N
      dec rcx
      .next_line:
            mov rdx, rcx
            .find_parallel_line:
                  ; tg A = tg B => lines are parallel
                  ; i tried to do comparison by eplison but the values
                  ; can not load into the FPU (endianess) so i left it like this
                  ;
                  ; fld dword [rbx + (rcx + (-1)) * WORD_SIZE]
                  ; fsub dword [rbx + rdx * WORD_SIZE]
                  ; fcomp dword [EPSILON]
                  mov eax, [rbx + (rcx + (-1)) * WORD_SIZE]
                  cmp eax, [rbx + rdx * WORD_SIZE]
                  je .print
                  jmp .continue
            .print:
                  dec rcx
                  print LINE_OUTPUT, rdx, rcx
                  inc rcx
      .continue:
            loop .find_parallel_line
            mov rcx, rdx
            loop .next_line
            ret

section '.data' writable
DESCRIPTION db\
"Ruslan Garifullin (https://github.com/ruslang02)", 10,\
"From N=%d lines (set by the coords of two points) find parallel ones.", 10,\
"Input format: <x1> <y1> <x2> <y2>.", 10,\
10, 0
POINT_PROVIDED_ERROR db "You have given a point. Line expected.", 10, 0
LINES rd N; float[N]
LINE_INPUT db "Line %d: ", 0
```

```
LINE_INPUT_FORMAT db "%f %f %f %f", 0
LINE_OUTPUT db "Line #%d is parallel to line #%d.", 10, 0
x1 dd ?
y1 dd ?
x2 dd ?
y2 dd ?
```

io.asm:

```
extrn scanf
extrn printf
; printf macro
macro print format*, arg1, arg2, arg3 {
      push rax
      push rdx
      push rcx
      mov rdi, format
      if arg1 eq
            xor rsi, rsi
      else
            mov rsi, arg1
      end if
      if arg2 eq
            xor rdx, rdx
      else
            mov rdx, arg2
      end if
      if arg3 eq
            xor rcx, rcx
      else
            mov rcx, arg3
      end if
      xor rax, rax
      call printf
      pop rcx
      pop rdx
      pop rax
}

; scanf macro
macro read format*, arg1*, arg2*, arg3*, arg4* {
      push rcx
      push rdx
      lea rdi, [format]
      mov rsi, arg1
      mov rdx, arg2
      mov rcx, arg3
      mov r8, arg4
      xor rax, rax
      call scanf
      pop rdx
      pop rcx
}
```

Список использованных источников

- https://c9x.me/x86/
- https://softcraft.ru/
- https://cs.fit.edu/~mmahoney/cse3101/float.html
- http://av-assembler.ru/instructions/