Микропроект №1.

Определение взаимной простоты

Ястребов Игорь Андреевич

НИУ ВШЭ

Факультет Компьютерных наук

Группа БПИ197

Текст задания

Вариант 27

Разработать программу, решающую вопрос – являются ли четыре заданных числа взаимно простыми. (числа задать машинного словами без знака)

Применяемые расчётные методы

В качестве проверки двух чисел на взаимную простоту был использован Алгоритм Евклида.

Используемые источники

Алгоритм Евклида - https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%B0

Исходный код

mp1.ASM :

```
format PE console

entry start

include 'win32a.inc'
include 'macro.inc'


section '.data' data readable writable

        strEnterPrompt  db 'Enter num #%d: ', 0
        positiveResStr  db 'The four numbers are coprime', 10, 0
        negativeResStr  db 'The four numbers are not coprime', 10, 0
        badNumStr       db 'Numbers have to be greater than 0', 0
        digitIn         db '%d', 0

        firstNum        dd ?    ; first number
        secondNum       dd ?    ; second number
        thirdNum        dd ?    ; third number
        fourthNum       dd ?    ; fourth number
        tmpStack        dd ?    ; the temporary value of the stack (for
procedures)
```

```
    result        dd 0    ; result value (the sum of all calls)
    NULL = 0              ; null value (for ExitProcess)


section '.code' code readable executable
    start:
        enterNum 1, firstNum    ; read the first num
        enterNum 2, secondNum   ; read the second num
        enterNum 3, thirdNum    ; read the third num
        enterNum 4, fourthNum   ; read the fourth num

        ; check that all numbers are > 0
        cmp [firstNum], 0
        jle terminateProg
        cmp [secondNum], 0
        jle terminateProg
        cmp [thirdNum], 0
        jle terminateProg
        cmp [fourthNum], 0
        jle terminateProg



        primeCompare firstNum, secondNum ; 1 & 2
        add [result], ebx
```

```
primeCompare firstNum, thirdNum ; 1 & 3
add [result], ebx
primeCompare firstNum, fourthNum ; 1 & 4
add [result], ebx
primeCompare secondNum, thirdNum ; 2 & 3
add [result], ebx
primeCompare secondNum, fourthNum ; 2 & 4
add [result], ebx
primeCompare thirdNum, fourthNum ; 3 & 4
add [result], ebx

cmp [result], 0
je resultPositive
jmp resultNegative

resultPositive:
        push positiveResStr
        call [printf]
        jmp endProg

resultNegative:
        push negativeResStr
        call [printf]
        jmp endProg
        endProg:
                call [getch]
                push NULL
                call [ExitProcess]

        terminateProg:
                push badNumStr
                call [printf]

                call [getch]
                push NULL
                call [ExitProcess]

;--------------------------------------------------
coprimeProc:                    ; compares two numbers, if they are
coprime sets ebx to 0, else sets ebx to 1
        mov [tmpStack], esp
        pop ebx                 ; pop the 2 values from the stack
        pop eax
        primeLoop:




        modLoop1:
```

```
        cmp eax, ebx          ; compare eax and ebx
        jl endModLoop1         ; if eax < ebx, end first subtraction loop
        sub eax, ebx          ; eax -= ebx
        jmp modLoop1          ; jump back to the start of the
subtraction loop

    endModLoop1:
        cmp eax, 0            ; if eax == 0 then end the loop
        je endPrimeLoop1

    modLoop2:
        cmp ebx, eax          ; compare ebx and eax
        jl endModLoop2         ; if eax > ebx, end second subtraction
loop
        sub ebx, eax          ; ebx -= eax
        jmp modLoop2          ; jump back to the start of the
subtraction loop

    endModLoop2:
        cmp ebx, 0            ; if ebx == 0 then end the loop
        je endPrimeLoop2


    jmp primeLoop              ; end of the main loop

   endPrimeLoop1:
    cmp ebx, 1              ; if ebx == 1 then return 1, else return 0
    je endPrimeLoopNo
    jmp endPrimeLoopYes

   endPrimeLoop2:           ; if eax == 1 then return 1, else return 0
    cmp eax, 1
    je endPrimeLoopNo
    jmp endPrimeLoopYes

   endPrimeLoopYes:
    mov ebx, 0
    jmp endPrime
   endPrimeLoopNo:
    mov ebx, 1
    jmp endPrime
   endPrime:
    mov esp, [tmpStack]
    ret

;---------------------------------------------------------------------------




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

import msvcrt, \
       printf, 'printf', \
       scanf, 'scanf', \
       getch, '_getch'
```

macro.inc:

```
macro enterNum numberNum, number{
    push numberNum          ; print the user prompt
    push strEnterPrompt
    call [printf]

    push number
    push digitIn
    call [scanf]
}
;------------------------------------------------------------
macro primeCompare num1, num2{
    push [num1]             ; push the 2 values to the stack, then figures
out if they are coprime
    push [num2]
    call coprimeProc
}
```

Тест программы

Тест с не взаимно простыми числами:

test1.txt

Тест с взаимно простыми числами:

test2.txt