Федеральное государственное бюджетное образовательное учреждение высшего образования «Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа №5

по дисциплине ЭВМ и периферийные устройства

Непривилегированные команды процессоров Intel х86

Вариант №16

Выполнил: Кузнецов Даниил Андреевич студент группы ИВТ-42-23

Проверила: доцент Андреева А.А.

Чебоксары, 2025

**Цель работы**

Получить практические навыки по использованию непривилегированных команд процессоров Intel х86.

**Задание к лабораторной работе**

Составить подпрограмму, реализующую действия, соответствующие варианту задания, приведенному ниже. Номер варианта выбирается по порядковому номеру студента в списке группы. Для отладки подпрограммы составляется основная программа, определяющая необходимые структуры данных и вызывающая рабочую процедуру.

Написать процедуру, производящую поиск заданной цепочки бит в массиве. Адрес массива задается парой ES:BX, длина (в байтах) - регистром CX. Битовая строка записывается в регистр EAX, начиная с младшего бита, количество бит в строке указывается в регистре DL. Процедура возвращает в регистре DI значение 0FFFFh, если строка не найдена, или номер первого бита строки.

**Тексты программ**

; func.asm

code segment

assume cs: code

public findBits

.386

findBits proc

; eax: битовая строка

; bx: массив байт

; cx: длина массива (в байтах)

; dl: длина битовой строки

; dh: значение бита из массива

; bp: индекс проверяемого бита в массиве

; esi: индекс проверяемого бита в битовой строке

; di: индекс начального бита в массиве

xor di, di ; индекс начального бита = 0

shl cx, 3 ; находим последний индекс

xor dh, dh ; начального бита в массиве:

sub cx, dx ; cx \* 8 - dl + 1

inc cx

startBitLoop: mov bp, di ; индекс проверяемого бита в массиве = индексу начального бита

xor esi, esi ; индекс проверяемого бита в строке = 0

checkBitLoop: xor dh, dh ; бит массива = 0

bt es:[bx], bp ; считать бит массива в CF

jnc arr0 ; если CF = 0, то переход

arr1: mov dh, 1 ; иначе бит массива = 1

arr0: bt eax, esi ; считать бит строки в CF

jnc bits0 ; если CF = 0, то переход

bits1: xor dh, 1 ; иначе инвертируем бит массива

bits0: test dh, dh ; т.о. если биты равны, то DH = 0

jnz nextBit ; если биты не равны, то переход

inc bp ; увеличить индекс бита в массиве

inc esi ; увеличить индекс бита в битовой строке

xor dh, dh ; DH = 0, для сравнения

cmp si, dx ; если достигнут конец битовой строки

jae found ; то проверены все биты и можно вернуть ответ

jmp checkBitLoop ; иначе рассматриваем следующий бит

nextBit: inc di

cmp cx, di

jae startBitLoop

notFound: mov di, -1

found: ret

findBits endp

code ends

end

; main.asm

sseg segment stack 'stack'

dw 256 dup(?)

sseg ends

data segment

; 11000000 10000111 10011111 10110000

arr1 db 00000011b, 11100001b, 11111001b, 00001101b

size1 dw $-arr1

; 0000001000

str1 dd 0001000000b

len1 db 10

; 10110000

str2 dd 00001101b

len2 db 8

; 01010101

str3 dd 10101010b

len3 db 8

data ends

code segment

assume ds: data, cs: code, ss: sseg

extrn findBits: near

.386

callFindBits macro arr, size, string, len

lea bx, arr1

mov cx, size

mov eax, string

mov dl, len

call findBits

endm

\_start: mov ax, data

mov ds, ax

mov es, ax

callFindBits arr1, size1, str1, len1 ; DI = 00002h

callFindBits arr1, size1, str2, len2 ; DI = 0018h

callFindBits arr1, size1, str3, len3 ; DI = FFFFh

mov ax, 4c00h

int 21h

code ends

end \_start

**Результаты работы программы**

После первого вызова DI = 0002h, после второго – DI = 0018h, после третьего – DI = 0FFFFh.

**Выводы**

Получены практические навыки по использованию непривилегированных команд процессоров Intel х86.