# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №4**

# по дисциплине «Организация систем и ЭВМ»

**Тема:**

«**Написание собственного прерывания и работа со строками**»

Студент гр. 3388 Снигирёв А.А.

Преподаватель Фирсов М.А.

Санкт-Петербург 2024

# Цель работы.

Изучить работу с прерываниями и обработку строк в ассемблере 8086. Написать программу, реализующую собственное прерывание 9h с обработкой строки.

# Задание.

Состоит из двух основных задач:

1) Реализация сценария работы с прерываниями в соответствии с вариантом.

2) Реализация преобразования строки с использованием команд работы со строками в соответствии с вариантом.

Вариант 19: 3с

|  |  |
| --- | --- |
| Действия основной программы | Действия 9h |
| 2) замена 9h;  3) ожидание ввода строки | вып. работу |

с) Найти во входной строке самую длинную последовательность повторяющихся символов. В выходной строке вывести первую такую последовательность, а затем - число последовательностей с этой максимальной длиной. Например: "dkBbRRRTTTBBRRR888"→"RRR4".

Замечания:

1) В сценариях опущены:

– Вывод приветственного сообщения и ввод строки пользователем. Это 1-ый пункт основной программы каждого сценария (поэтому основная программа во всех сценариях начинается с пункта 2). Слова́ «Ожидание ввода строки», которые встречаются в таблице сценариев, – это всего лишь организация задержки с ожиданием нажатия клавиш, для ввода данных не используется.

– Действия по восстановлению изменённых прерываний, если они не требуются по сценарию специально. После завершения программы все изменённые прерывания всегда должны быть восстановлены (независимо от сценария). Перед завершением программы следует восстановить в том числе и те вектора прерываний, которые могли бы восстановиться по таймеру. После завершения программы DOS освобождает память, занимаемую программой, что может помешать корректному восстановлению векторов. Поэтому рассчитывать на восстановление по таймеру после завершения программы будет неправильно.

2) «Ожидание ввода строки» следует делать так, чтобы пользователь мог нажать множество клавиш, прежде чем решит завершить ввод нажатием Enter.

4) «вып. работу» означает, что надо на основе введённой строки (п. 1 основной программы) создать модифицированную строку и вывести её на экран. При выполнении преобразования нельзя портить исходную строку, результат преобразования должен записываться в выходную строку.

5) Перед запуском ожидания нажатия клавиши («ожидание ввода строки» в таблице) вывести сообщение об этом.

6) Перед заменой 9h следует сделать небольшую задержку (см. 3.2 «Задержка во времени»), чтобы предшествующая активность пользователя была обработана до того, как 9h будет изменён;

7) Для исключения возможного взаимного влияния системных и пользовательских прерываний рекомендуется отвести в программе под стек не менее 1Кбайт.

# Основные теоретические положения.

1. Краткие сведения о прерываниях.

Прерывание – это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств. Когда возникает прерывание, процессор прекращает выполнение текущей программы (если её приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата (CS:IP) – ме́ста, с которого будет продолжена прерванная программа. Затем в CS:IP загружается адрес программы обработки прерывания и ей передаётся управление.

Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP, во вторых – CS) и хранятся в младших 1024 байтах памяти.

Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

В лабораторной работе среди действий по обработке прерывания могут быть:

– преобразование строки;

– вывод сообщения на экран;

– замена или восстановление вектора прерывания (другого или самого́ себя);

Программа, использующая новые программы обработки прерываний, при своём завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX.

Для задания адреса собственного прерывания с заданным номером в таблицу векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес нового обработчика.

При использовании функции 25H прерывания 21H DOS знает, что вы делаете, и гарантирует, что в момент записи прерывания будут заблокированы.

Прерывания бывают аппаратные (вызываемые в результате сигналов от оборудования) и программные (вызываемые в коде). В лабораторной работе предлагаются к замене вектора́ следующих прерываний:

– 09H – аппаратное прерывание, возникающее в результате нажатия клавиш клавиатуры;

– 16H – программное прерывание для ожидания ввода символа с клавиатуры;

– 21H – программное прерывание для вызова сервисов DOS.

Заменённое тело 9h следует завершать не выходом из прерывания (iret), а переходом к выполнению старого тела 9h (использовать команду jmp dword ptr), иначе обработка сигналов клавиатуры будет нарушена. То же самое касается прерываний 16h и 21h.

2. Краткие сведения о командах обработки строк.

Для обработки строковых данных ассемблер имеет пять групп команд обработки строк:

— MOVS — переслать один байт или одно слово из одной области памяти в другую;

Каждая команда имеет модификации, указывающие размер операнда: байт (B), слово (W), двойное слово (D). Например: MOVSB, MOVSW, MOVSD.

Эти команды предполагают, что регистры DI и SI содержат относительные адреса, указывающие на необходимые области памяти (для загрузки можно использовать команду LEA). Регистр SI обычно связан с регистром сегмента данных – DS:SI. Регистр DI всегда связан с регистром дополнительного сегмента – ES:DI. Следовательно, команды MOVS, STOS, CMPS и SCAS требуют инициализации регистра ES (обычно адресом в регистре DS).

Префиксы REP/REPE/REPZ/REPNE/REPNZ позволяют этим командам обрабатывать строки любой длины.

# Выполнение работы

# В основной процедуре Main происходит вывод вспомогательных сообщений и ввод строки пользователем.

# После этого задается небольшая задержка, дабы программа успела все обработать.

# Далее с помощью кода, данного в теоретических положениях,

# происходит сохранение оригинального прерывания в переменные IP\_9H и SEG\_9H, также происходит замена прерывания 9h на пользовательское, реализованное в функции ProcessString и соответствующее условию варианта.

# Поскольку 9h – это аппаратное прерывание, реагирующее на нажатие клавиш, то после ввода пользователю предлагается нажать любую клавишу, чтобы запустить получившееся прерывание.

# После нажатия запускается обработчик прерывания ProcessString.

# Он будет отрабатывать пока происходит нажатие клавиш, а клавиша не является enter. После нажатия enter происходит восстановление оригинального вектора прерывания 9h.

# В обработчике прерывания ProcessString происходит обработка строки, введенной пользователем. В регистры ES и DS сохраняется адрес сегмента данных, это необходимо, так как регистры SI и DI, необходимые для использования MOVSB, связаны с ними.

# Также организован условный оператор, гарантирующий, что обработка произойдет только один раз.

# Дальше реализован цикл, проходящий по всей строке и находящий указатель на первую последовательность одинаковых символов, длины MAX\_LEN, а также количество последовательностей такой длины.

# В следующем шаге с помощью команды movsb происходит копирование части входной строки длины MAX\_LEN, и на которую указывает bx в буфер выходной строки. Еще в конец добавляется содержимое счетчика COUNT, приведенное к символьному типу, и завершающий символ.

# В конце функции происходит вывод новой строки и переход к оригинальному обработчику прерывания 9h.

# Тестирование прерывания и обработки строки. 9H реагирует на нажатие клавиш, поэтому для продолжения работы программы пользователь должен нажать клавишу, в реализованной программе — любую. Если клавишу не нажать, то обработка строки не стартует. Для завершения работы нужно нажать Enter.

# Рисунок 1 — Обработка множественного нажатия клавиш

# Рисунок 2 — Успешное продолжение и завершение выполнения после нажатия Enter

# Рисунок 3 — Отработка при нажатии enter сразу после ввода

# Рисунок 4 — работа при зажимании клавиши(не enter)

# 4. Тестирование различных входных данных

|  |  |
| --- | --- |
| Input | Output |
| If we'd go agaain All the way from the staart | aa3 |
| MMMMOOOORRRRDDDDOOOORRRR | MMMM6 |
| MmOoRRDdOoRr | RR1 |
| dkBbRRRTTTBBRRR888 | RRR4 |
| Neeear, far, fooreeewer you aaare | eee3 |
| And so he spoooke the lord of Caaastameeere | ooo3 |
| One Ring to rule them all, One Ring to find them | ll1 |

**Вывод**

В ходе лабораторной работы были изучены прерывания и работа со строками на языке Ассемблер Intel 8086. Разработан собственный обработчик прерывания 9h и логика изменения строки.

# ПРИЛОЖЕНИЕ

# ИСХОДНЫЙ КОД ПРОГРАММЫ

1. laba.asm

­

ASSUME CS:CODE, SS:AStack, DS:DATA

AStack SEGMENT STACK

DW 400h DUP(0) ; 1 КБ стека

AStack ENDS

DATA SEGMENT

EOFLine EQU '$' ;Определение символьной константы "Конец строки"

STR1HEAD DB 50h,0

STR1 DB 80 DUP('\*'),0AH, 0DH, EOFLine ; Буфер памяти для введённых символов плюс байты для корректного завершения вывода

START\_MESSAGE DB 'Здравствуйте! Введите строку: $'

HELP\_MSG DB 'Для завершения программы нажмите Enter. Для вызова измененного 9h нажмите любую клавишу',13,10,'$'

NEWLINE DB 0Dh, 0Ah, '$'

STR2 DB 80 DUP(' '),0AH, 0DH, EOFLine

MAX\_LEN DW 0

COUNT DW 0

IP\_9H DW 0 ; для хранения смещения

SEG\_9H DW 0 ; для хранения сегмента

COUNTER DW 0

DATA ENDS

CODE SEGMENT

StopTime PROC NEAR

push cx

push dx

push ax

mov cx, 0eh

mov dx, 0ffffh

mov ah, 86h

int 15h

pop ax

pop dx

pop cx

ret

StopTime ENDP

WriteMsg PROC NEAR

push ax

mov AH,9

int 21h ; Вызов функции DOS по прерыванию

pop ax

ret

WriteMsg ENDP

StringRead PROC FAR

mov AH,0ah ; функция ввода строки

push dx ; смещение заголовка строки...

int 21h ; вызов функции DOS ввода строки

pop bp ; ...поместить в bp

xor bx,bx

mov bl,ds:[bp+1] ; теперь в bx количество введённых символов

add bx,bp ; теперь bx указывает на конечный введённый символ

add bx,2 ; теперь bx указывает на байт, следующий за финальным 0dh

mov word ptr [bx+1],240ah ; добавить в конец 0ah и '$'

ret

StringRead ENDP

ProcessString PROC FAR

; Сохранение всех регистров, которые будут изменены в обработчике

push ax

push bx

push cx

push dx

push si

push di

push es

push ds

; сохранение в DS, ES адрес начала сегмента даных

MOV AX, DATA

MOV DS, AX

MOV ES, AX

; Счетчик, чтобы обработка строки происходила только один раз

MOV bx, COUNTER

cmp bx, 0

jnz Output

inc COUNTER

; Пользовательская обработка строки (обработчик прерывания)

lea si, STR1 ; Указатель на начало строки STR1

mov dx, 0 ; Текущая длина подпоследовательности

mov bx, 0 ; Указатель на первую максимальную последовательность

mov al, [si] ; сохраняем в al первый элемент строки str1

inc dx ; Увеличиваем текущую длину

inc si

ITERATION:

mov ah, [si] ; загружаем текущий символ

inc si ; Увеличиваем индекс для следующей итерации

cmp ah, '$' ; Проверка на конец строки

je END\_PROCESS ; Завершение, если достигнут конец строки

cmp ah, al ; Сравнение текущего символа с предыдущим

mov al, ah ; Обновляем предыдущий символ для следующей итерации

jne RESET\_LEN ; Если символы не равны, переходим к сбросу длины

inc dx ; Увеличение длины последовательности, если символы равны

jmp ITERATION ; Переход к следующей итерации

RESET\_LEN:

cmp dx, MAX\_LEN ; Сравнение текущей длины с максимальной

ja CHANGE\_MAX ; Если текущая длина больше максимальной, обновляем MAX\_LEN

je INCREASE\_COUNT ; Если текущая длина равна максимальной, увеличиваем COUNT

mov dx, 1 ; Сброс длины текущей последовательности

jmp ITERATION

CHANGE\_MAX:

mov MAX\_LEN, dx ; Обновление MAX\_LEN

mov dx, 1 ; Сброс длины

mov bx, si ; Обновление указателя на первую максимальную последовательность

dec bx ;

sub bx, MAX\_LEN ;

mov COUNT, 1 ; Сброс счётчика

jmp ITERATION

INCREASE\_COUNT:

inc COUNT ; Увеличение счётчика

mov dx, 1 ; Сброс длины

jmp ITERATION

END\_PROCESS:

MOV CX, MAX\_LEN ; Длина последовательности

cld

mov SI, bx ; Указатель на начало нужной последовательности

lea DI, STR2 ; Указатель на начало STR2

rep movsb ; Копируем CX байт из STR1 в STR2

MOV AX, COUNT ; Приведение к строковому типу данных

ADD AL, '0' ;

MOV [DI], AL ; Записываем количество последовательностей

inc DI

MOV BYTE PTR [DI], '$' ; Добавление символа конца строки

mov dx, OFFSET STR2 ; Вывод обработанного сообщения и предложения завершить программу

call WriteMsg

Output:

; Восстановление содержимого регистров

pop ds

pop es

pop di

pop si

pop dx

pop cx

pop bx

pop ax

; Перход к оригинальному обработчику прерывания 9h

jmp dword ptr [IP\_9H] ; Переход к сохраненному обработчику 9h

ProcessString ENDP

Main PROC FAR

push ds

sub ax, ax

push ax

mov ax, DATA

mov ds, ax

; Вывод стартового сообщения

mov dx, OFFSET START\_MESSAGE

call WriteMsg

; Ввод строки

mov dx, OFFSET STR1HEAD

call StringRead

; Вывод инструкции

mov dx, OFFSET NEWLINE

call WriteMsg

mov dx, offset HELP\_MSG

call WriteMsg

; Задержка перед заменой прерывания

call StopTime

; Сохранение оригинального обработчика прерывания 9h

mov ah, 35h

mov al, 9h

int 21h

mov SEG\_9H, es

mov IP\_9H, bx

; Замена обработчика прерывания 9h на ProcessString

push ds

mov dx, OFFSET ProcessString

mov ax, SEG ProcessString

mov ds, ax

mov ah, 25h

mov al, 9h

int 21h

pop ds

wait\_for\_click:

mov ah, 0h ; функция "проверка клавиши"

int 16h ; вызываем прерывание для чтения клавиши

; cmp ah, 1Ch ; проверяем, что нажата клавиша Enter (код 0x0D)

;jne wait\_for\_enter ; если нет, продолжаем ожидание;

push ds

mov dx, IP\_9H ; смещение для оригинального обработчика 9h

mov ax, SEG\_9H ; сегмент оригинального обработчика

mov ds, ax ; помещаем в DS

mov ah, 25h ; функция установки вектора прерывания

mov al, 9h ; номер вектора

int 21h ; замена прерывания

pop ds

; Программа завершает выполнение и ждет нажатия клавиши, чтобы завершить работу.

ret

Main ENDP

CODE ENDS

END Main