

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра электронных вычислительных машин

Отчет по предмету
Проектирование на языке ассемблера
Вариант 9

Лабораторная работа №2
«Обработка символьных данных»

Выполнил:
Студент группы 150503
Шарай П.Ю.

Проверил:
Туровец Н.О.

Минск 2022

Цель работы: ознакомиться с директивами определения данных, изучить команды пересылки данных и передачи управления, изучить строчные операции и прерывания консольного ввода-вывода высокого уровня

Теоретические сведения:

1. Директивы определения данных.

Директивы определения данных указывают ассемблеру, что в соответствующем месте программы располагается переменная, определяют тип переменной (байт, слово и т.д.), задают ее начальное значение и ставят в соответствие переменной метку, которая будет использоваться для обращения к этим данным. Определения данных записываются в общем виде следующим образом:

метка D*:

- DB – байт,
- DW – слово (2 байта),
- DD – двойное слово (4 байта),
- DF – 6 байт (для представления адреса (FAR указатель)),
- DQ – 8 байт,
- DT – 10 байт (80-битные данные для FPU).

2. Команды пересылки данных и способы адресации.

Базовой командой пересылки данных является команда MOV:

MOV приемник, источник

Эта команда копирует содержимое источника в приемник, источник при этом не изменяется.

3. Команды передачи управления.

Команды передачи управления служат для организации ветвления вычислительного процесса. Предлагается использовать следующие команды этой группы:

-- безусловный переход (JMP метка) – переход на метку без возврата (от текущего положения до 32768 байт).

-- условный переход (Jxx метка, где xx – условие перехода, обычно используется после команды CMP(источник, приемник), которая сравнивает два числа, вычитая второе из первого, но не сохраняет результат, а лишь устанавливает в соответствии с результатом флаги состояния) – переход в зависимости от состояния флагов, которые обычно устанавливаются предыдущей арифметической или логической операцией.

-- цикл (LOOPxx метка) – организация циклов в программах, используя регистр CX в качестве счетчика цикла.

4. Прерывания ввода-вывода.

Для организации ввода данных с клавиатуры предлагается использовать одну из ниже приведенных функций DOS:

-- Функция DOS 01h (INT 21h) – считывает (ожидает) символ со стандартного входного устройства. Отображает этот символ на стандартное выходное устройство (эхо). при распознавании Ctrl-Break выполняется INT 23H

-- Функция DOS 06h (INT 21h) – считать символ из STDIN без эха, без ожидания и без проверки на Ctrl-Break

-- Функция DOS 07h (INT 21h) – считать символ из STDIN без эха, с 15 ожиданием и без проверки на Ctrl-Break:

-- Функция DOS 08h (INT 21h) – считать символ из STDIN без эха, с ожиданием и проверкой на Ctrl-Break

-- Функция DOS 0Ah (INT 21h) – считать строку символов из STDIN в буфер: Ввод: AH = 0Ah, DS:DX = адрес буфера.

Для вывода данных на консоль предлагается использовать одну из ниже приведенных функций DOS:

-- Функция DOS 02h (INT 21h) – записать символ в STDOUT с проверкой на Ctrl-Break

-- Функция DOS 06h (INT 21h) – записать символ в STDOUT без проверки на Ctrl-Break

-- Функция DOS 09h (INT 21h) – записать строку в STDOUT с проверкой на Ctrl-Break

-- Функция DOS 40h (INT 21h) – записать строку в файл или устройство

5. Макросы.

Макросом называется фрагмент программы, который подставляется в код программы всякий раз, когда ассемблер встречает его имя в тексте программы. Макрос начинается именем и директивой MACRO, а заканчивается директивой ENDM.

Код программы

```
.model small
```

```
.stack 100h
```

```
.data
```

```
message1 db "Enter string: $" ;input string
```

```
message2 db 0Ah, 0Dh, "Enter substring: $";searched string
```

```
message3 db 0Ah, 0Dh, "Result string: $" ;string without searched
```

```
enter db 0Ah, 0Dh, "$"
```

```
length equ 200
```

```
Strb db '$'
```

```
Strl db length
```

```
Str db length dup('$') ;string
```

```
SubStrb db '$'
SubStrl db length
SubStr  db length dup('$')
```

```
.code
```

```
start:
```

```
    mov ax, @data
    mov ds, ax
```

```
    lea dx, message1      ; смещение 'Enter string'
    call outputString      ; вывод строки
    lea dx, enter
    call outputString      ; вывод строки
```

```
    lea dx, Strb           ; смещение MainString
    call inputString       ; ввод строки
    lea dx, enter          ; смещение enter
    call outputString      ; вывод введенной строки
```

```
    lea dx, message2      ; адрес 'Enter substring'
    call outputString      ; вывод
    lea dx, enter          ; смещение enter
    call outputString      ; вывод
```

```
    lea dx, SubStrb        ; смещение SubString
    call inputString       ; ввод
    lea dx, enter
    call outputString
```

```
    mov al, [Strl]         ; проверка на равенство
    cmp al, [SubStrl]
    jb exit
```

```
    xor cx, cx             ; обнуление регистра
    lea si, Str            ; начало строки
    dec si
    jmp start_find
```

```
find:
    inc si
    cmp [si], ' '          ; сравнение с пробелом
    je start_find
    cmp [si], '$'          ; сравнение с концом строки
    je exit
```

```

jmp find
start_find:
inc si
cmp [si], ''
je start_find
lea di, SubStr          ; указатель на начало подстроки
call searchSubString
jmp find

```

```

searchSubString proc
    push ax              ;сохранение значений регистров
    push cx
    push di
    push si

    xor cx, cx           ;обнуление регистра
    mov cl, [SubStrl]    ; cl имеет длину SubStr
    comparestr:         ;сравнение строк
    mov ah,[si]
    dec cx
    cmp ah,[di]
    je compare          ;если символы равны
    jne NotEqual        ;если символы не совпали
    compare:
    inc si              ;переход к следующему символу
    inc di
    cmp cx,0            ;конец слова
    je check            ;проверка на конец слова
    jne comparestr      ;дальнейший поиск слова

    check:
    cmp [si], ''        ;проверка конца слова
    je Equal            ;конец слова достигнут
    jne NotEqual        ;конец слова не достигнут

    Equal:              ;найден совпадающий символ
    call length
    call shift
    call searchSubString

    NotEqual:           ;найден несовпадающий символ
    pop si              ;восстановление регистров
    pop di

```

```
pop cx
pop ax
ret
```

```
searchSubString endp      ;конец процедуры поиска
```

```
shift proc                ;процедура сдвига
    push cx                ;сохранение регистров
    push di
    push bx
```

```
    lea ax, Str            ;смещение строки
    add al, [Strl]
    sub ax, si
    mov cx, ax             ;длина слова
    add cx, 2
```

```
    ;shifting the word
    sdvigg_vlevo:
        mov ah, [si]       ; сохраняет текущий элемент
        sub si, bx         ; сдвиг влево
        mov [si], ah
        add si, bx
        inc si
    loop sdvigg_vlevo
    xor bh, bh             ;обнуление регистра
```

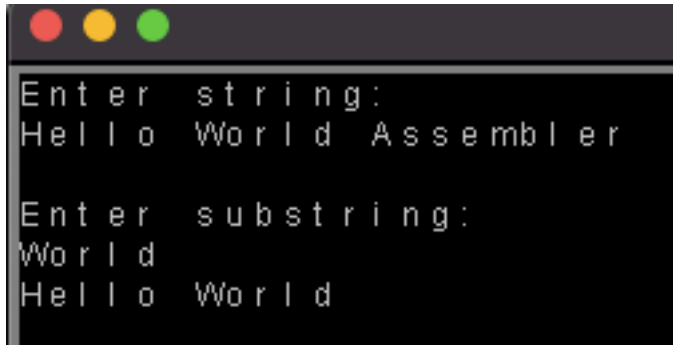
```
    pop bx                ;восстановление регистров
    pop di
    pop cx
    ret
shift endp                ;конец процедуры
```

```
length proc               ;процедура поиска длины слова
    push ax                ;сохранение регистра
    skip:
        inc si
        cmp [si], ' '
        je skip
    mov ax, si             ; сравнения элемента строки с ' '
```

```
word:
    mov dh, [si]
    inc si
```

cmp [si], ' '	; сравнение с пробелом
je continue	
cmp [si], '\$'	;сравнение с концом строки
je continue	
jmp word	
continue:	
push si	
sub si,ax	
mov bx,si	
pop si	;восстановление регистров
pop ax	
ret	
length endp	;конец процедуры
inputString proc	;процедура ввода строки
push ax	;сохранение регистра
mov ah, 0Ah	;прерывание
int 21h	
pop ax	;восстановление регистра
ret	
inputString endp	;конец процедуры
outputString proc	;процедура вывода строки
push ax	;сохранение регистра
mov ah, 09h	;перывание
int 21h	
pop ax	;восстановление регистра
ret	
outputString endp	;конец процедуры
exit:	;процедура конца программы
lea dx, Str	
call OutputString	;вывод строки
mov ax,4c00h	;прерывание
int 21h	
end start	

Результат работы программы:



```
Enter string:
Hello World Assembler

Enter substring:
World
Hello World
```

Вывод

В ходе выполнения лабораторной работы была написана программа для удаления слова в строке, стоящего перед заданным словом. Мы ознакомились с директивами определения данных, изучили команды пересылки данных и передачи управления, изучили строчные операции и прерывания консольного ввода-вывода высокого уровня.