Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 2

по дисциплине «Программирование на языке Ассемблера»

Вариант 2

Выполнил студент гр. 150502: Альхимович Н.Г.

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.

Вариант задания:

Выполнить реверс слов строки.

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\* значение

Здесь D\* является определением типа и может быть задано как:

• 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).

Для работы с символами и строками символов в данной работе достаточно типа DB. Для работы с числовыми данными (например, индексами символов в строке) лучше использовать тип DW, т.к. длина строки может превышать размер в 255 символов.

Поле значения в определении переменной может содержать одно или несколько чисел, строк символов (взятых в одиночные («'») или двойные кавычки («"»)), операторов ? и DUP, разделенных запятыми.

Имя переменной будет соответствовать адресу первого из указанных значений. Если вместо точного значения указан символ «?», то такая переменная считается неинициализированной и ее значение на момент запуска программы может оказаться любым. Если нужно заполнить участок памяти повторяющимися данными, то вместо указания точного значения используется специальный оператор DUP, имеющий формат

счетчик DUP (значение)

где счетчик – число задающее число элементов.

Директива EQU предназначена для описания констант:

имя EQU операнд

Директива требует замены указанного имени на заданный операнд. Директиву EQU можно ставить в любое место программы.

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. Команда MOV действует аналогично операторам присваивания из языков высокого уровня. В качестве источника для MOV могут использоваться: число (непосредственный операнд), регистр общего назначения, сегментный регистр или переменная (операнд, находящийся в памяти). В качестве приемника для MOV могут использоваться: регистр общего назначения, сегментный регистр (кроме CS) или переменная.

Оба операнда должны быть одного и того же размера – байт, слово или двойное слово. Нельзя выполнять пересылку данных с помощью MOV из одной переменной в другую, из одного сегментного регистра в другой и нельзя помещать в сегментный регистр непосредственный операнд – эти операции выполняют только двумя командами MOV.

Также в данной работе можно использовать стек – это специальным образом организованный участок памяти, используемый для временного хранения переменных, для передачи параметров вызываемым подпрограммам и для сохранения адреса возврата при вызове процедур и прерываний. Данные можно записывать и считывать только с вершины стека. Таким образом, если записать в стек числа 1, 2, 3, то при чтении они будут получаться в обратном порядке — 3, 2, 1.

Для работы со стеком используются команды:

• PUSH источник – поместить данные в стек,

• POP приемник – считать данные из стека.

Способы адресации определяют формирование адреса памяти для доступа к данным. Для правильной адресации по умолчанию (без явного указания сегментного регистра) требуется следующее:

• регистр CS должен указывать на начало сегмента кода – команды переходов всегда используют этот сегментный регистр;

• регистр SS должен указывать на начало сегмента стека – если для косвенной адресации используется регистр BP, то это адресация к стеку.

• регистр DS должен указывать на начало сегмента данных – адресация к данным по умолчанию (кроме BP) использует этот сегментный регистр.

• Операторы адресации:

• SEG выражение – сегментный адрес;

• OFFSET выражение – смещение;

• THIS тип – текущий адрес (TASM и MASM);

• тип PTR выражение – переопределение типа;

• SMALL выражение – 16-ти битное смещение (TASM);

• SHORT выражение – 8-ми битное смещение.

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). Для перехода в диапазоне 128 байт от текущего места можно использовать команду JMP SHORT метка.

• условный переход (Jcc метка, где cc – условие перехода, обычно используется после команды CMP) – переход в зависимости от состояния флагов, которые обычно устанавливаются предыдущей арифметической или логической операцией. Флаги, проверяемые командой, кодируются в ее мнемонике (например, JC – переход, если установлен флаг CF). Сокращения «L» (less – меньше) и «G» (greater – больше) применяются для сравнения целых чисел со знаком, а «A» (above – над) и «B» (below – под) для сравнения целых чисел без знака.

• переход, если CX = 0 (JCXZ метка).

Для организации условных переходов достаточно часто используется команда сравнения (CMP источник, приемник), которая сравнивает два числа, вычитая второе из первого, но не сохраняет результат, а лишь устанавливает в соответствии с результатом флаги состояния.

4. Строковые операции.

Кроме перечисленных выше базовых команд пересылки данных, для обработки строк символов можно использовать специальные строковые операции.

Каждая строковая операция, представлена в процессоре двумя видами команд, различающихся по последнему символу мнемоники команды:

• B (byte) – для обработки строк, состоящих из символов-байтов (как в данной лабораторной работе),

• W (word) – для обработки строк, состоящих из символов-слов.

Если флаг направления DF перед выполнением команды строковой обработки установлен в 0 (выполнена команда CLD), то значение в индексном регистре автоматически увеличивается, если в 1 (выполнена команда STD) – уменьшается. Индексные регистры уменьшаются или увеличиваются на 1, если команды работают с байтами, или на 2 – при работе со словами.

Строковые операции обеспечивают выполнение следующих операций:

• сравнение строк (CMPS) – команда сравнивает значение элемента одной строки (DS:SI) со значением элемента второй строки (ES:DI) и устанавливает индексных значения регистров на следующие элементы строк. Сравнение происходит так же, как и по команде сравнения CMP. Результатом операции является установка флагов.

• сканирование строки (SCAS) – команда производит сравнение содержимого аккумулятора (AL или AX) с байтом или словом памяти, абсолютный адрес которого определяется парой ES:DI, после чего регистр DI устанавливается на следующий символ. Команда SCAS используется обычно для поиска в строке (ES:DI) элемента заданного в аккумуляторе.

• пересылка строки (MOVS) – пересылает поэлементно строку DS:SI в строку ES:DI и устанавливает значения индексных регистров на следующий элемент строки.

• запись в строку (STOS) – заполняет строку, содержащуюся по адресу ES:DI, элементом из аккумулятора (AL или AX), не влияет на флаги.

• чтение из строки (LODS) – записывает в аккумулятор (AL или AX) содержимое ячейки памяти, адрес которой задается регистрами DS:SI, не влияет на флаги.

Команды строковой обработки чаще всего используются с однобайтными префиксами (префиксами повторения), которые обеспечивают многократное автоматическое повторение выполнения команды:

• повторять, пока равно (REPE),

• повторять, пока ноль (REPZ),

• повторять (REP),

• повторять, пока не равно (REPNE),

• повторять, пока не ноль (REPNZ).

Префиксы повторения ставятся перед строковыми командами обязательно в той же строке. Префикс использует регистр CX как счетчик циклов. На каждом этапе цикла выполняются следующие действия:

1) если CX=0, то выход из цикла и переход к следующей команде;

2) выполнение заданной строковой операции;

3) уменьшение CX на единицу, флаги при этом не изменяются;

4) выход из цикла, если:

a) условие сравнения не выполняется для SCAS или CMPS;

b) префикс REPE и ZF=0 (последнее сравнение не совпало);

c) префикс REPNE и ZF=1 (последнее сравнение совпало);

5) изменение значения индексных регистров в соответствии со значением флага направления DF и переход на начало цикла.

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. После директивы MACRO могут быть перечислены через запятую идентификаторы параметров, используемых в макросе, что делает макрос гибким средством оформления кода.

Код программы:

.model small

.stack 100h

.data

input db "Your string: $"

string db "This string needs to be reversed$"

result db 0ah, 0dh, "Reversed string: $"

.code

mov ax, @data

mov ds, ax

start:

lea dx, input

mov ah, 09h

int 21h

lea dx, string

mov ah, 09h

int 21h

lea bx, string

mov si, bx ;moves address of the string in bx

next\_byte:

cmp [si], '$' ;unless the end of the string is reached

je the\_end

inc si

jmp next\_byte

the\_end:

dec si ;corrects the last position of the string

reverse:

cmp bx, si ;bx points to the beginning and si points to the end of string

jae each\_word

;swapping symbols and moving to the middle of the string

mov al, [bx]

mov ah, [si]

mov [si], al

mov [bx], ah

inc bx

dec si

jmp reverse

each\_word:

lea bx, string

mov si, bx

jmp next\_byte\_of\_the\_word

shift:

mov bx, si ;beginning of the next word

next\_byte\_of\_the\_word:

cmp cx, 1h ;if the last word is reversed

je stop

cmp [si], ' ' ;another word is identified

je the\_end\_of\_the\_word

cmp [si], '$' ;the last word is reached

je the\_end\_of\_the\_string

inc si

jmp next\_byte\_of\_the\_word

the\_end\_of\_the\_string: ;1h in cx indicates, that the whole string is processed

mov cx, 1h

the\_end\_of\_the\_word:

mov di, si ;so as not to "spoil" si

dec di ;correcting position of the end

inc si ;points to the next word

reverse\_of\_the\_word:

cmp bx, di

jae shift

mov al, [bx]

mov ah, [di]

mov [di], al

mov [bx], ah

inc bx

dec di

jmp reverse\_of\_the\_word

stop:

lea dx, result

mov ah, 09h

int 21h

lea dx, string

mov ah, 09h

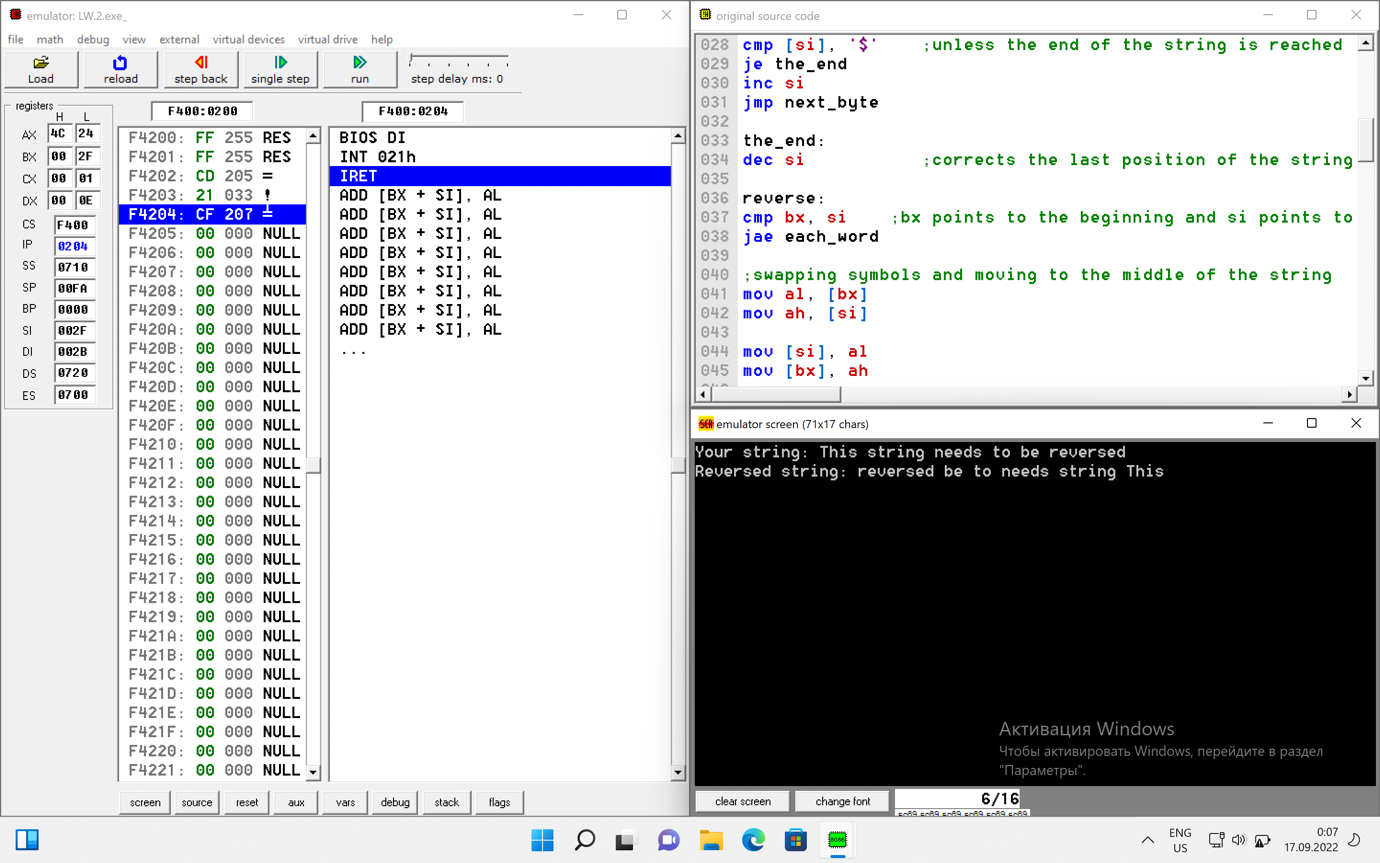
int 21h

mov ah, 4ch ;the program ends

int 21h

ret

Примеры работы программы:



Вывод:

В ходе лабораторной работы было написано приложение, выводящее на экран строку символов, для которой выполнен реверс ее слов. Я ознакомилась с директивами определения данных, изучила команды пересылки данных и передачи управления, а также строчные операции и прерывания консольного ввода-вывода высокого уровня.