Лабораторная работа № 2

Команды пересылки данных

Цель работы:

Практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение команд передачи данных. Практическое освоение основных функций отладчика TD.

Порядок выполнения работы:

1. Создать рабочую папку для текстов программ на ассемблере и записать в нее файлы tasm.exe, tlink.exe, rtm.exe и td.exe из пакета tasm, а также файл с исходным текстом программы на ассемблере, который сохранить с именем prog_3.asm, prog_4.asm,
2. Создать загрузочный модуль, загрузить его в отладчик и выполнить программу в пошаговом режиме.

Содержание отчета:

1. Цель работы.
2. Постановка задачи.
3. Листинг программы Prog_3,
4. Таблица состояния регистров процессора, в ходе выполнения программы Prog_3.
5. Листинг программы Prog_4,
6. Таблица состояния программы Prog_4 в ходе выполнения программы.
7. Вывод.

Постановка задачи:

1. Изучить методические указания и рекомендованную литературу.
2. Написать программу Prog_3 с помощью шаблона, приведённого ниже.
3. Задать начальные значения переменных A, B, C, D в сегменте данных в соответствии с вариантом (Таблица 1).
4. Проследить за работой в Турбоотладчике, заполнить Таблицу 2 для строк программы с 11 по 35.
5. Написать программу Prog_4, согласно условию (Приложение 2)

Теоретическая часть

КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

Команды передачи данных (команды пересылок) предназначены для организации пересылки данных между регистрами, регистрами и памятью, памятью и регистрами, а также для загрузки регистров или ячеек памяти данными. При выполнении команд передачи данных флаги не устанавливаются.

Наиболее часто используемой командой передачи данных является команда MOV.

MOV (MOVe operand)

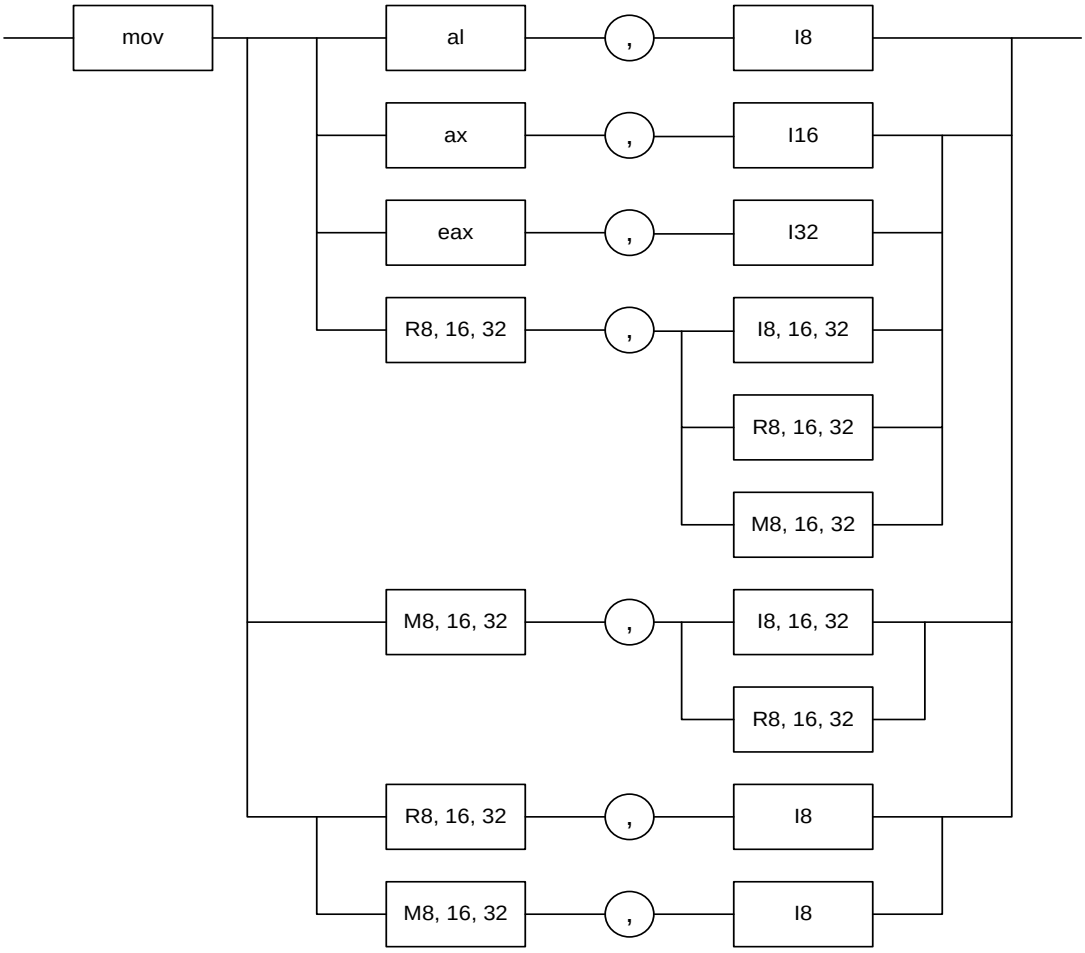
Пересылка операнда

```
--------------------------------------------------------
          mov     приемник,    источник
--------------------------------------------------------
```

Назначение:

Пересылка данных между регистрами или регистрами и памятью.

Синтаксис:



Синтаксическое описание команды **mov**

Алгоритм работы:

Копирование второго операнда в первый операнд.

Применение:

Команда **mov** применяется для различного рода пересылок данных, при этом, несмотря на всю простоту этого действия, необходимо помнить о некоторых ограничениях и особенностях выполнения данной операции:

- направление пересылки в команде **mov** всегда справа налево, то есть из второго операнда в первый;
- значение источника не изменяется;
- оба операнда не могут быть из памяти (при необходимости можно использовать цепочечную команду **movs**);
- лишь один из операндов может быть сегментным регистром;
- желательно использовать в качестве одного из операндов регистр **al/ax/eax**, так как в том случае **TASM** генерирует более быструю форму команды **mov**;

Пример использования:

```
mov   al, 5  ;в регистр al помещается число 5
mov   bl, al ;в регистр bl помещается значение регистра al
mov   cx, [A];в регистр cx помещается значение переменной A
mov   [A], ax ;переменная A получает значение регистра ax
mov   eax, 5 ;в регистр eax помещается число 5
mov   ebx, eax ;в регистр ebx помещается значение регистра eax
```

XCHG (eXCHanGE)

Обмен

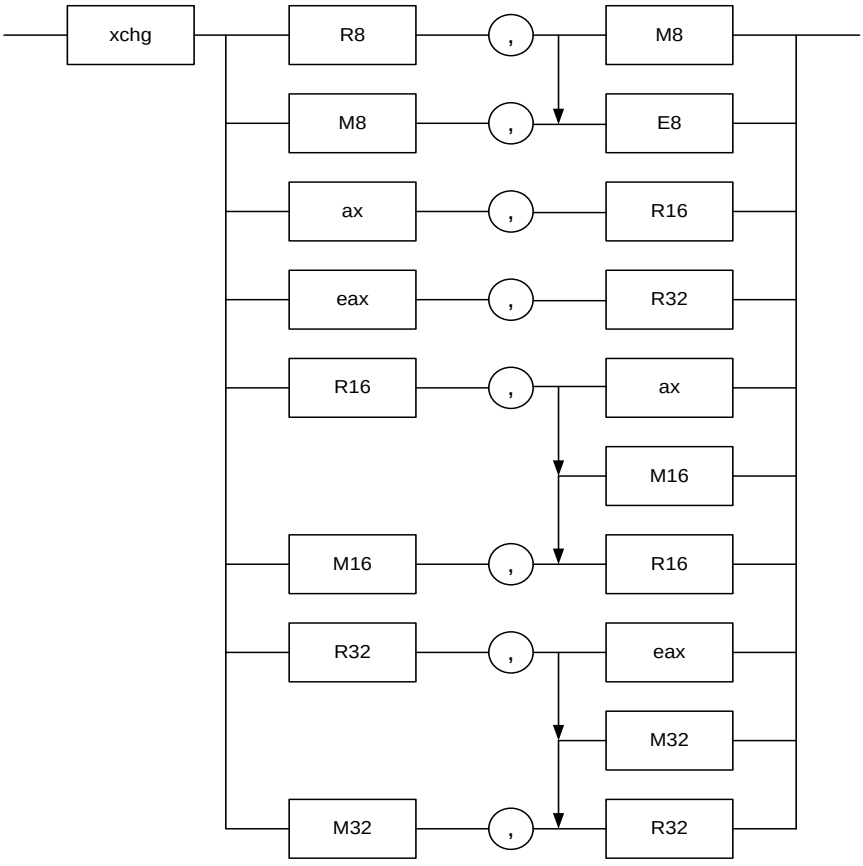
---

xchg операнд_1, операнд_2

Назначение:

Обмен двух значений между регистрами или между регистрами и памятью.

Синтаксис:



Синтаксическое описание команды **xchg**

Алгоритм работы:

Обмен содержимого операнд_1 и операнд_2.

Применение:

Команду **xchg** можно использовать для выполнения операции обмена двух операндов с целью изменения порядка следования байт, слов, двойных слов или их временного сохранения в регистре или памяти. Альтернативой является использование для этой цели стека.

Пример использования:

```
; поменять порядок следования байт в слове
ch   label  byte
dw   0f85ch
.  .  .
mov  al, ch1
xchg ch1, al
mov  ch1, al
```

КОМАНДА ЗАГРУЗКИ ИСПОЛНИТЕЛЬНОГО АДРЕСА

LEA reg, mem;

Применение:

Загружает в регистр reg, указанный в качестве первого операнда, относительный адрес второго операнда, который находится в памяти

КОМАНДЫ РАБОТЫ СО СТЕКОМ

PUSH – поместить в стек.

PUSH src;

Применение:

Используются для занесения данных в стек и извлечения данных из стека. Для адресации к вершине стека используется **регистр указателя стека SP**, который при выполнении стековых команд автоматически модифицируется.

В качестве операнда может использоваться любой 16 разрядный регистр или двухбайтовая ячейка памяти. Все стековые команды манипулируют только двухбайтовыми данными – словами.

Алгоритм работы

Это команда Она уменьшает на 2 содержимое указателя стека **SP** и заносит на вершину стека по этому адресу двухбайтовый операнд, указанный в команде.

SP:= (SP) – 2 , [(SS):(SP)]:= (src).

POP – извлечь из стека

POP dst;;

Алгоритм работы

Команда извлекает 16-ти разрядные данные из ячеек стека, на которые указывает указатель **SP** и помещает их в получатель, указанный в команде. Содержимое **SP** при этом автоматически **увеличивается на 2**.

Пример программного кода

Lab2_1.ASM

```
Title 'LAB2_1 NESTERENKO AA'

.model small
.stack 100h
.data
A DB ?
B DB ?
C DB ?
D DB ?

.code
start:
mov ax, @data
mov ds, ax
mov A,5Ah
mov B,2
mov C, 42
mov D, 9
mov al, A
mov ah, B
xchg al, ah
mov bx, 3E10h
mov cx, bx
push bx
push cx
push ax
lea si, C
mov ax, si
lea di ,D
mov bx, di
pop ax
pop cx
pop bx
mov bx, ax
mov A, al
mov B, ah
mov C,0
mov ax, 4c00h
int  21h
end start
end
```

ПРИЛОЖЕНИЕ 1

Таблица 1 Начальные значения переменных для программы

№ варианта	Значения переменных				№ варианта	Значения переменных			
	A	B	C	D		A	B	C	D
1	3	9	2Eh	AAh	9	32	6	9h	eh
2	5Ah	2	42	9	10	22h	32	25	10h
3	B5h	55h	15	8	11	32	C1h	6	21
4	22h	7	8	12	12	3Bh	10	12h	9
5	15	1Ah	1Fh	6	13	3Bh	1Fh	11	12
6	3	1Eh	12	22h	14	5	8	10h	0Fh
7	7h	12	1Dh	9	15	12h	12	05h	9
8	5	2Eh	18h	11	16	9	1Ch	8	10h

Таблица 2 Результат выполнения программы

Вариант…			
№ строки	**Команда Ассемблера**	**Машинный код**	**Состояние регистров[1]**
11	**mov ax, @data**	**B8FF00**	**ax=00FF[2], ip=0103**
…			

[1] в таблице отражаются только регистры, значимые на данном шаге.
[2] значения, хранящиеся в регистрах, могут отличаться

ПРИЛОЖЕНИЕ 2

Задание 2. ***Исходные данные:***

- Дата рождения студента в формате **ДД-ММ-ГГГГ**.
  Считать ДД и ММ десятичными числами, ГГГГ – 16-ричным числом.
- Минимальная и максимальная отметки, полученные студентом в экзаменационную сессию – **min, max** (десятичные числа)
- Возраст студента (количество полных лет) – **vozrast** (десятичное число).

Задание:

1. Составить программу, которая содержит:
   - сегмент стека длиной 512 байт,
   - сегмент данных, включающий:

ИМЯ	***ДЛИНА (ТИП)***	***СОДЕРЖИМОЕ***
den	байт	**ДД**
qwer	слово	4321h
mes	слово	**ММ**
Mas_1	одномерный байтовый массив длиной **ММ** элементов	**Vozrast**-значение каждого элемента
Mas_2	Двумерный массив, тип элементов – слово. Количество строк - **min**, количество столбцов – **max**	не определено
stroka	строка символов	**Фамилия** студента

   - сегмент кода, содержащий:
     1.1. Инициализацию сегмента данных;
     1.2. Команды, выполняющие следующие действия:
       1.2.1. загрузку 16-ричного числа **ГГГГ** в регистр **DI**
       1.2.2. загрузку переменных:
         - **den** в регистр **AL**
         - **mes** в регистры **CX** и **ES**
       1.2.3. сохранить в стеке значения переменных **den, mes**
       1.2.4. обмен содержимого переменных **qwer** и **mes**
       1.2.5. поместить в регистр **SI** адрес переменной **den**
       1.2.6. копирование регистра **DS** в **CS**
       1.2.7. извлечь из стека и поместить в регистры **CX, DX** значения переменных **den, mes**
       1.2.8. вывод на экран содержимого переменной **stroka**
       1.2.9. команды, заканчивающие программу.

2. Получить объектный код программы, листинг и исполняемый файл.

3. Выполнить программу по шагам. При этом:
   3.1. убедиться в правильности загрузки исходных данных
   3.2. объяснить изменения содержимого регистров и памяти, наблюдая в окне CPU отладчика результаты выполнения каждой команды программы

Примечания:

1) в начале программы должны быть указаны[3]: номер лабораторной работы, фамилия студента, группа, перечислены исходные данные
2) все ключевые моменты программы должны быть снабжены комментариями; отметить особые случаи пересылки данных командой **mov.**

[3] Д и р е к т и в а TITLE. Для того, чтобы вверху каждой страницы листинга печатался заголовок (титул) программы, используется директива TITLE в следующем формате:

TITLE текст

Рекомендуется в качестве текста использовать имя программы, подкоторым она находится в каталоге на диске. Например, если программа называется ASMSORT, то можно использовать это имя и описательный комментарий общей длиной до 60 символов:

TITLE ASMSORT - Ассемблерная программа сортировки имен