Лабораторная работа № 2_1

Команды пересылки данных

Цель работы:

Практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение команд передачи данных и использования различных способов адресации операндов. 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 prog4.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 prog3_1.
4. Таблица состояния программы в ходе выполнения программы.
5. Листинг программы prog3_2.
6. Окно DUMP памяти отладчика до и после выполнения варианта.
7. Ответы на контрольные вопросы.
8. 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 19 по 35.
5. Выполнить вариант задания (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)

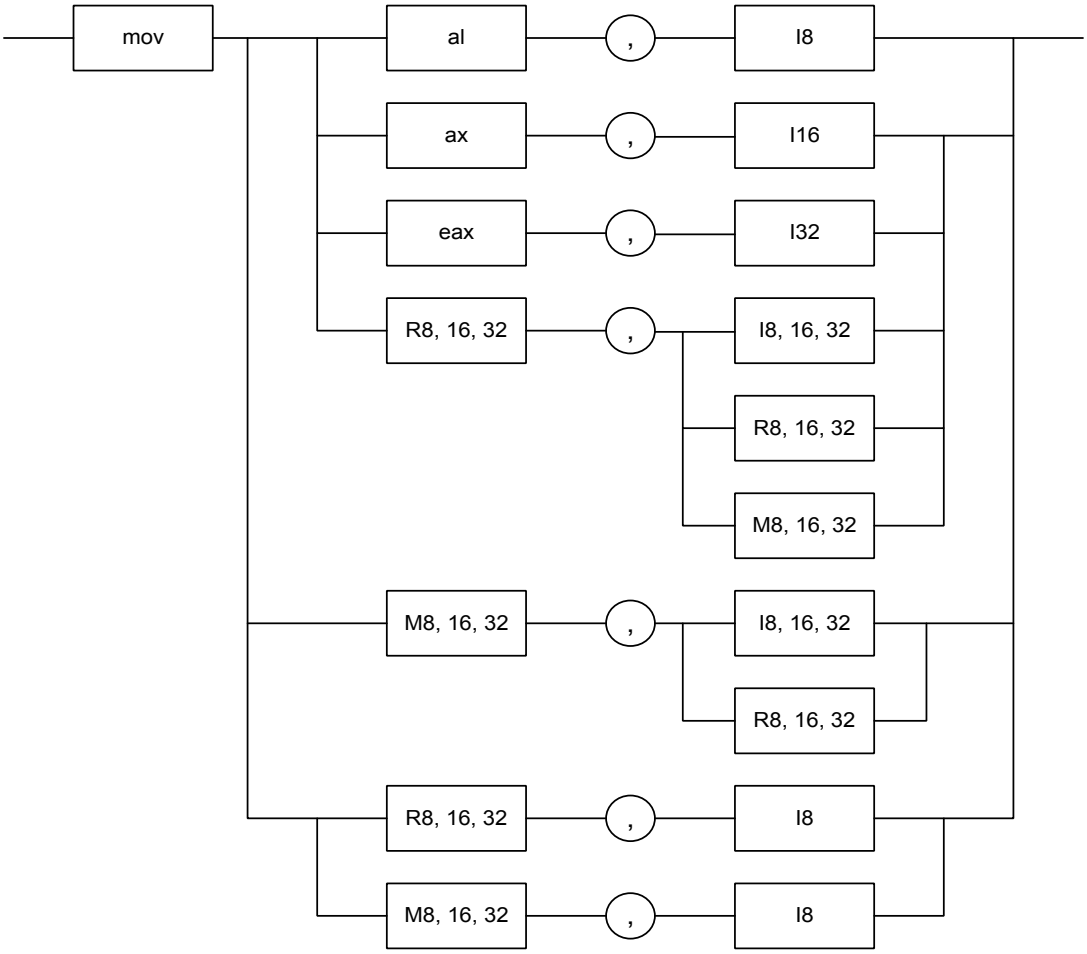
Пересылка операнда

```
--------------------------------------------------------
          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

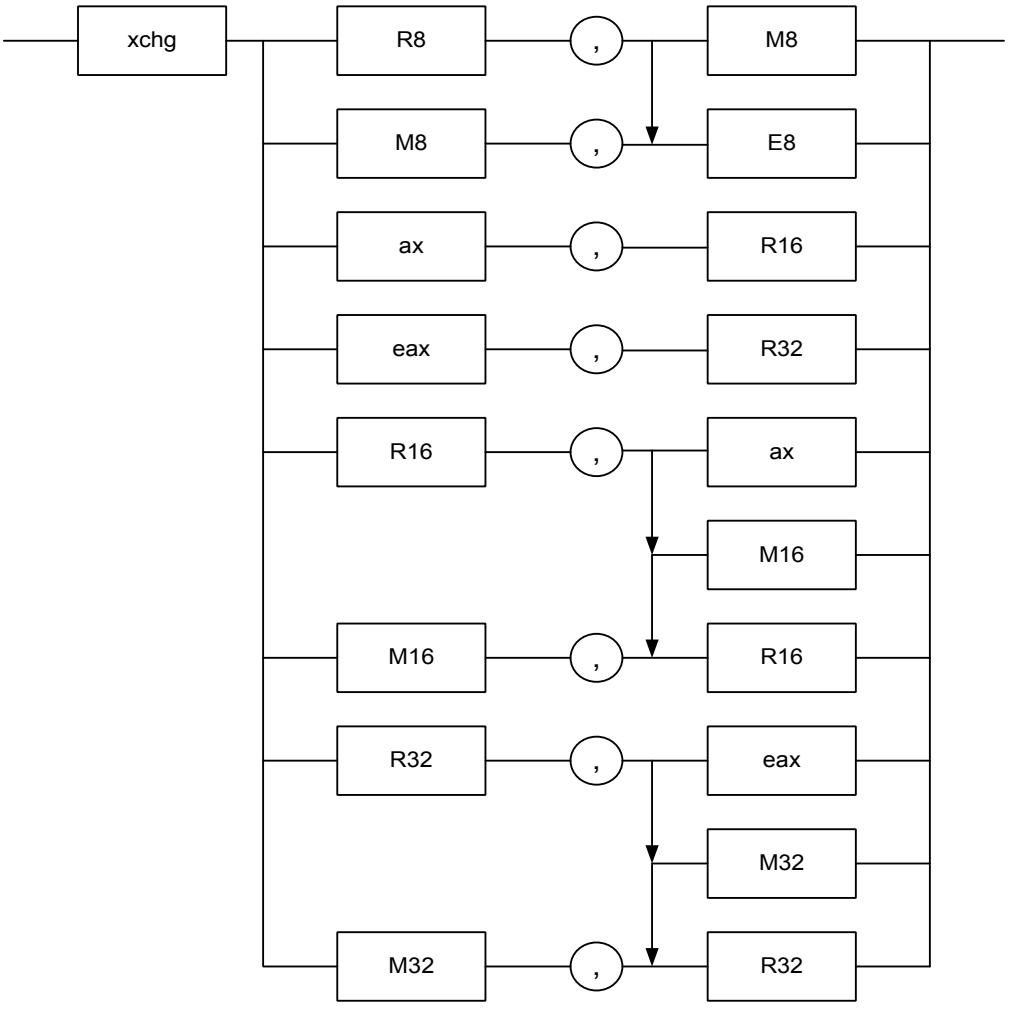
---

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 ко-манде. Содержимое **SP** при этом автоматически **увеличивается на 2**.

Контрольные вопросы

1. Как записываются общие команды передачи данных на Ассемб-лере? Что может использоваться в качестве операндов команды?
2. Для чего предназначена команда LEA и что является ее операн-дами?
3. Поясните выполнение команд работы со стеком.
4. Поясните выполнение команды обмена данными.

Приложение 1

Пример программного кода

```
     PPG_4. asm        {Пример использования команд передачи данных}
1    .model small
2    .stack 100h
3    .data
4    A db ?            ; зарезервировать
5    B db ?            ; в памяти
6    C db ?            ; место для
7    D db ?            ; переменных
8
9    .code             ; открыть сегмент кода
10   Start:
11   mov AX, @Data     ; инициализировать сегмент
12   mov DS, AX        ; данных
13   mov A,  9         ; инициализировать
14   mov B,  1Ch       ; переменные A, B, C, D
15   mov C,  8         ; значениями Вашего
16   mov D,  10h       ; варианта
17   mov AL, A
18   mov AH, B
19   xchg AL,Ah
20   mov BX, 3E10h
21   mov
22   push BX
23   push CX
24   push AX
25   lea SI, C
26   mov AX, SI
27   lea DI, D
28   mov BX, DI
29   pop AX
30   pop CX
31   pop BX
32   mov BX, AX
33   mov A, AL
34   mov B, AH
35   mov C, 0
36   mov AX, 4C00h     ; завершить программу
37   int 21            ; с помощью DOS
38   end Start         ;закрыть процедуру
39   END               ; закрыть программу
```

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 …						
Файл и № строки	Команда Ассемблера	Машинный код	Длина машинного кода, байт	Логический адрес в памяти	Физический адрес в памяти	Состояние регистров и флагов
Prog_31						AX=…, BX=…, CX=…, DX=…, SP=…, BP=…, SI=…, DI=…, IP=…, DS=…, SS=…, CS=…, ES=…; CF=…, ZF=…, SF=…, OF=…, PF=…, AF=…
Prog_32						AX=…, BX=…, CX=…, DX=…, SP=…, BP=…, SI=…, DI=…, IP=…, DS=…, SS=…, CS=…, ES=…; CF=…, ZF=…, SF=…, OF=…, PF=…, AF=…

Приложение 2

Варианты заданий:

Вариант 1. Задать одномерный массив, состоящий из X элементов (X задается преподавателем из диапазона [1..8]). Заполнить массив константами. Переместить заданный массив в другую область памяти, поменяв местами элементы с четными и нечетными номерами (поставив каждый элемент с четным номером на место нечетного элемента и каждый элемент с нечетным номером – на место четного)

а) элементы массива – однобайтовые (vec1);

б) элементы массива – двухбайтовые (vec2);

Вариант 2. Задать одномерный массив, состоящий из X элементов (X задается преподавателем из диапазона [1..8]). Заполнить массив константами. Переместить в другую область памяти элементы с нечетными номерами

а) элементы массива – однобайтовые (vec1);

б) элементы массива – двухбайтовые (vec2);

Вариант 3. Задать одномерный массив, состоящий из X элементов (X задается преподавателем из диапазона [1..8). Заполнить массив константами. Переместить в другую область памяти элементы с четными номерами

а) элементы массива – однобайтовые (vec1);

б) элементы массива – двухбайтовые (vec2);

Вариант 4. Задать одномерный массив, состоящий из X элементов (X задается преподавателем из диапазона [1..8]). Заполнить массив константами. Создать новый одномерный массив, поместив в него на место элементов с четными номерами элементы заданного массива с нечетными номерами и обнулив элементы нового массива с нечетными номерами

а) элементы массива – однобайтовые (vec1);

б) элементы массива – двухбайтовые (vec2);

Вариант 5. Задать одномерный массив, состоящий из X элементов (X задается преподавателем из диапазона [1..8]). Заполнить массив константами. Создать новый одномерный массив, поместив в него на место элементов с нечетными номерами элементы заданного массива с нечетными номерами и заполнив элементы нового массива с четными номерами максимальными значениями констант

а) элементы массива – однобайтовые (vec1);

б) элементы массива – двухбайтовые (vec2).

Таблица индивидуальных заданий

Вариант	Массивы	Значения элементов
1	vec1	1,2,3,4,8,7,6,5
	vec2	-10,-20,10,20,-30,-40,30,40
2	vec1	5,6,7,8,12,11,10,9
	vec2	-20,-30,20,30,-40,-50,40,50
3	vec1	8,7,6,5,1,2,3,4
	vec2	-30,-40,30,40,-10,-20,10,20
4	vec1	12,11,10,9,5,6,7,8
	vec2	-40,-50,40,50,-20,-30,20,30

5	vec1	11,12,13,14,18,17,16,15
	vec2	10,20,-10,-20,30,40,-30,-40
6	vec1	18,17,16,15,11,12,13,14
	vec2	30,40,-30,-40,10,20,-10,-20
7	vec1	21,22,23,24,28,27,26,25
	vec2	40,50,-40,-50,20,30,-20,-30
8	vec1	28,27,26,25,21,22,23,24
	vec2	20,30,-20,-30,40,50,-40,-50
9	vec1	31,32,33,34,38,37,36,35
	vec2	50,60,-50,-60,70,80,-70,-80
10	vec1	38,37,36,35,31,32,33,34
	vec2	70,80,-70,-80,50,60,-50,-60
11	vec1	1,3,5,7,9,11,13,15
	vec2	20,30,40,50,-50,-40,-30,-20
12	vec1	9,11,13,15,1,3,5,7,
	vec2	-50,-40,-30,-20,20,30,40,50
13	vec1	28,27,26,25,21,22,23,24
	vec2	20,-20,30,-30,40,-40,50,-50,
14	vec1	31,32,33,34,35,36,37,38
	vec2	50,-50,60,-60,70,-70,80,-80
15	vec1	1,2,3,4,8,7,6,5
	vec2	-1,-2,-3,-4,-8,-7,-6,-5
16	vec1	18,17,16,15,11,12,13,14
	vec2	-10,-20,10,20,-30,40,-30,40
17	vec1	8,7,6,5,1,2,3,4
	vec2	30,40,-30,-40,10,20,-10,-20