МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Отчет   
по лабораторной работе №2  
на тему: «Организация межсегментных переходов»  
по дисциплине «Архитектура ЭВМ и систем»

Выполнили: Марочкин М.А. Шифр: 170584  
 Яшин М. О. Шифр: 170133  
 Шорин В.Д. Шифр: 171406  
ИПАИТ  
Направление: 09.03.04 «Программная инженерия»  
Группа: 71-ПГ  
Проверил(а):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Отметка о зачете:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Дата «\_\_\_\_» \_\_\_\_\_\_ 2017 г.

Орел, 2017г.

**Цель работы.**

Цели лабораторной работы:

1. Изучение принципов функционирования памяти и микропроцессора компьютера при выполнении межсегментных переходов.

2. Приобретение навыков использования команд сдвига при написании ассемблерных программ.

3. Получение представления об особенностях обработки данных и режимах доступа к данным при выполнении операций сдвига над данными.

**Контрольные вопросы:**

1. Выделяют следующие режимы адресации команд:

1. Внутрисегментный прямой. Команда, к которой осуществляется переход, находится в том же сегменте кода, что и текущая команда перехода, т.е. при выполнении перехода содержимое регистра CS не изменяется. Эффективный адрес перехода (смещение команды в сегменте кода) вычисляется как сумма текущего содержимого указателя команд IP и 8- или 16- битного относительного смещения (длины пропускаемых команд). Данный режим допустим в условных и безусловных переходах.
2. Межсегментный прямой. Команда, к которой осуществляется переход, находится в другом сегменте кода по отношению к текущей команде перехода, т.е. при выполнении перехода изменяется содержимое регистра CS и регистра IP. В команде указывается пара: сегмент и смещение. Начальный адрес нового сегмента кода загружается в сегментный регистр CS, а смещение – в регистр IP. Данный режим допустим только в командах безусловного перехода.
3. Внутрисегментный косвенный. В этом случае двухбайтовый адрес перехода размещается в ячейках памяти по некоторому адресу (смещению) в сегменте данных. В команде перехода это смещение указывается в регистре процессора или ячейке памяти с помощью любого режима адресации данных, кроме непосредственного. Содержимое указателя команд IP заменяется соответствующим содержимым регистра или ячейки памяти. Данный способ допустим только в командах безусловного перехода.
4. Межсегментный косвенный. В этом режиме четырёхбайтовый адрес перехода размещается в смежных ячейках памяти по некоторому адресу (смещению) в сегменте данных. В команде перехода это смещение указывается в регистре процессора или ячейке памяти с помощью любого режима адресации данных, кроме непосредственного и регистрового. Содержимое регистров CS и IP заменяется содержимым двух смежных слов памяти, хранящихся по этому смещению в сегменте данных. Младшее слово загружается в регистр IP, старшее – в регистр CS. Данный режим допустим только в командах безусловного перехода.

2. Условием, на основании которого осуществляется переход, чаще всего выступают признаки результата выполнения предшествующей арифметической или логической команды (без флага AF). Каждый из признаков фиксируется в своём разряде регистра флагов PSW.

Флаги условий:

Флаг знака SF. Равен старшему биту результата: ноль – если результат положительный, и единица – если результат отрицательный.

Флаг нуля ZF. Устанавливается в единицу при получении нулевого результата и сбрасывается в ноль, если результат отличается от нуля.

Флаг паритета PF. Устанавливается в единицу, если младшие 8 бит результата содержат четное число единиц, в противном случае он сбрасывается в ноль.

Флаг переноса CF. При сложении (вычитании) устанавливается в единицу, если возникает перенос из младшего бита или заем из старшего бита.

Флаг вспомогательного переноса AF. Устанавливается в единицу, если при сложении (вычитании) возникает перенос (заем) из бита 3. Только для двоично-десятичной арифметики.

Флаг переполнения OF. Устанавливается в единицу, если возникает переполнение, т.е. получение результата вне допустимого диапазона. При сложении флаг устанавливается, если имеется перенос в старший бит и нет переноса из старшего бита и наоборот.

3. Эти команды также обеспечивают манипуляции над отдельными битами, перемещая биты операнда влево или вправо на определенное число битов, в зависимости от кода операции.

В командах сдвига влево с правой стороны операнда «вдвигаются» нули, а старшие биты «выдвигаются» с левой стороны и теряются, но последний из них сохраняется во флаге CF. Команды сдвига вправо аналогичным образом сдвигают биты вправо.

Команды арифметического линейного сдвига отличаются от команд логического сдвига тем, что они воспринимают сдвигаемые значения как числа со знаком и особым образом работают со знаковым битом (седьмым) числа. Но арифметический сдвиг вправо не помещает слева нули, а дублирует в старшие биты знак операнда.

Команды арифметического сдвига позволяют выполнить «быстрое» умножение и деление операнда на степени двойки. Например, сдвиг числа влево на один разряд аналогичен его умножению на 2, а сдвиг числа вправо на один разряд аналогичен делению его на 2. Преимущество этих команд по сравнению с традиционными командами умножения и деления заключается в скорости исполнения: команды сдвига выполняются быстрее.

4. Команды циклического сдвига отличаются от команд сдвига тем, что операнд считается «кольцом», в котором выдвигаемые с одной стороны биты вдвигаются с другой стороны. В командах простого циклического сдвига сдвигаемый бит одновременно и вдвигается в операнд с другого конца, и становится значением флага переноса CF.

5. ASSUME - псевдооператор, определяющий, каким сегментным регистрам соответствуют назначенные метками адреса начала сегментов.

**Текст программы.**

s\_s segment stack "stack"

dw 10 dup (?) ; зарезервировать 20 байт в сегменте стека

s\_s ends

d\_s1 segment

arg1 db 00101010b ; определить однобайтовое двоичное число

d\_s1 ends

d\_s2 segment

adr dd c\_s1:met1 ; определить адрес перехода на метку met1

d\_s2 ends

d\_s3 segment

arg2 db 00110011b ; определить однобайтовое двоичное число

d\_s3 ends

c\_s1 segment

assume ss:s\_s, cs:c\_s1, ds:d\_s1

met1: mov ax, d\_s1 ; метка перехода в первый сегмент кода met1

mov ds, ax

mov bl, arg1

ror bl, 3 ; циклически сдвинуть биты значения регистра BL вправо на 3

and bl, 00000001b ; получение в регистре BL значения бита, ставшего нулевым

jmp far ptr c\_s2:met2 ; прямой межсегментный переход во второй сегмент кода   
 по метке met2

c\_s1 ends

c\_s2 segment

assume ss:s\_s, cs:c\_s2, ds:d\_s3

met2: mov ax, d\_s3 ; метка перехода во второй сегмент кода met2

mov ds, ax

mov bh, arg2

rol bh, 3 ; циклически сдвинуть биты значения регистра BH влево на 3

and bh, 00000001b

jmp far ptr c\_s3:met3 ; прямой межсегментный переход в третий сегмент кода   
 по метке met3

c\_s2 ends

c\_s3 segment

assume ss:s\_s, cs: c\_s3, ds:d\_s1

begin:

assume ds:d\_s1

mov ax, d\_s1

mov ds, ax

mov al, arg1

shl al, 1 ; умножение значения в регистре AL на 2 с помощью сдвига влево

assume ds:d\_s3

mov ax, d\_s3

mov ds, ax

mov al, arg2

shr al, 2 ; деление значения в регистре AL на 4 с помощью сдвига вправа

assume ds:d\_s2

mov ax, d\_s2

mov ds, ax

lea bx, adr ; в регистр помещается адрес смещения adr

jmp dword ptr [bx] ; косвенный межсегментный переход по адресу в регистре   
 BX

met3: mov ah, 4ch ; метка перехода в третий сегмент кода met3

int 21h

c\_s3 ends

end begin