МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Лабораторная работа № 1

Программная модель процессора

Вариант № 8

ОТЧЕТ

по лабораторной работе

по дисциплине

Принципы и методы  
организации системных программных средств

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Викулова Е.Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Игнаков К. М.

19-В-2

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

**Цель работы**

* Освоить систему меню отладчика Turbo Debugger (td.exe)
* Изучить рамки окна CPU и локальные меню.
* Научиться управлять отображением и редактированием информации в рамках окна CPU
* Научится сохранять информацию, отображаемую в окне CPu в виде текстовых (в журнале отладчика Log) и бинарных файлов.
* Составить в кодах, ввести и выполнить в отладчике несколько команд перессылки с различными метолами адрессации, команды условного перехода и цикла.
* Написать программу. Ввести программу в отладчике, записать в память необходимые данные, выполнить программу в отладчике. Получить распечатки программы, дампов до и после выполнения программы.

**Задание**

Подготовить средствами отладчика в памяти данные для задачи: исходный массив кодов {a(1),a(2),...,a(n)}. Адрес массива и число элементов выбрать самостоятельно. Написать программу, выполняющую пересылку элементов исходного массива (или преобразование и пересылку), удовлетворяющих некоторому условию в другую область памяти.

1. **вариант:** Коды, не равные значениям *c*1, *c*2, *c*3.

**Алгоритм.**

НАЧАЛО

Ввести данные (массив)

Цикл ДЛЯ каждого элемента массива:

ЕСЛИ Элемент == С1:

ТО пропустить данный элемент

ИНАЧЕ переслать данный элемент

ЕСЛИ Элемент == С2:

ТО пропустить данный элемент

ИНАЧЕ переслать данный элемент

ЕСЛИ Элемент == С3:

ТО пропустить данный элемент

ИНАЧЕ переслать данный элемент

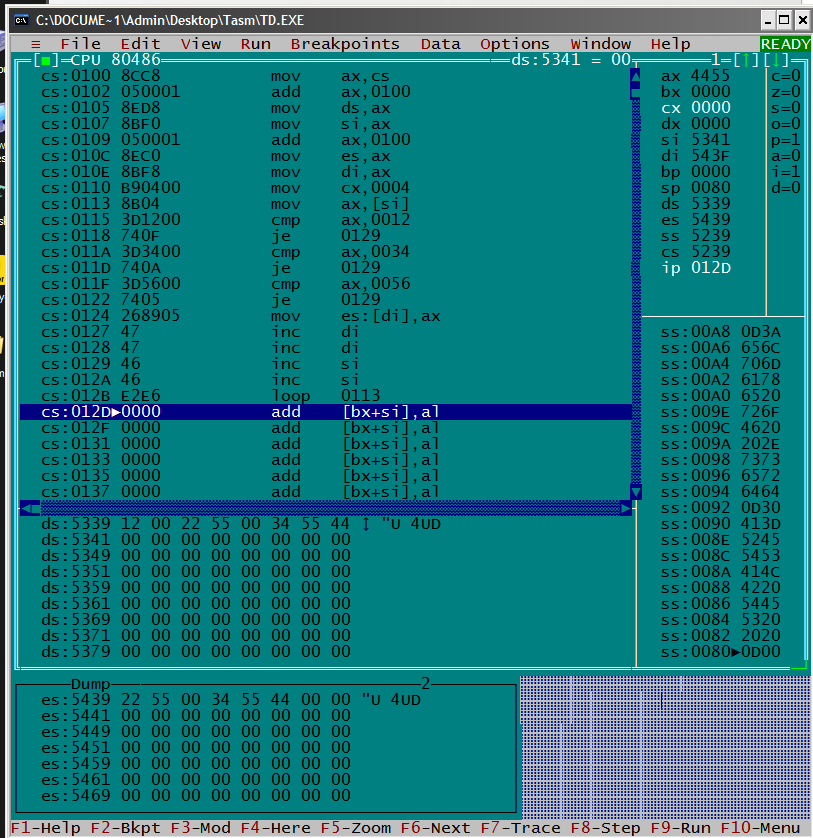
Всё – цикл

КОНЕЦ

Особенности:

Так как, согласно заданию, мы работаем со словами (2 байта = 16 бит) необходимо это учитывать при увеличении счетчиков регистров источника и приемника данных.

**Реализация в отладчике**



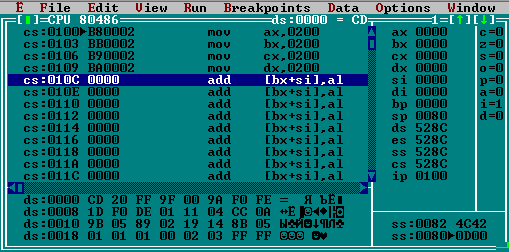
Входные данные:

12 00 22 55 00 34 55 44

Выходные данные:

22 55 00 34 55 44 00 00

**Способы адресации**



**Непосредственная адресация** – операнд располагается в последних байтах команды.

**mov ax,100**10111000 00 01  
 КОП ax

**Регистровая адресация** – операнд находится в одном из общих (сегментных) регистрах.

**mov si,di**10001011 11110111

КОП d w mod reg r/m  
1 – запись в регистр, код которого находится в reg

1 – операнд является байтом

11 – операнд в регистре

110 – адрес регистра si

111 – адрес регистра di

**Прямая (абсолютная) адресация** – эффективный адрес является частью команды (байты смещения в команде). Прямая адресация может быть дальней с указанием дальнего адреса сегмент:смещение, такая адресация используется только в командах межсегментных переходов и вызовов подпрограмм.

**mov bx,[128]**10001011 00011110 28 01

КОП d w mod reg r/m

1 – запись в регистр, код которого находится в reg

1 – операнд является байтом

00 – операнд в памяти, без смещения

011 – адрес регистра bx

110 – метод адресации DS:disp16

**Косвенная адресация -** операнд находится в памяти, для вычисления эффективного адреса используются значения базовых, индексных регистров и, возможно, смещение.

**mov ax, [bx+si+100]**10001011 10000000 00 01

КОП d w mod reg r/m

1 – запись в регистр, код которого находится в reg

1 – операнд является байтом

10 – операнд в памяти, 2 байта смещения

000 – адрес регистра ax

000 – метод адресации DS:BX+SI+disp16

**Неявная адресация -** адресуемый объект указывается с помощью одного байта команды вместе с кодом операции без выделения специального поля.

**push si**01010110  
 КОП si

**Формат команд**

Для программиста ассемблерная команда – это строка вида:

КОП Оп1, Оп 2

где КОП - код операции (например, mov… add… jmp…)

Оп1 и Оп2 – операнды, операнд 1 - приемник, операнд 2 - источник.

Для процессора команда – это кодовая последовательность определенной длины, в которой содержится информация о том «что делать» и «с чем делать».

Форматы команд и информация о кодировании отдельных полей команд – см. asmmet.doc (в методичке полно опечаток, но если будет понятен принцип, и у Вас есть отладчик, то это не страшно☺)

Совсем чуть-чуть теории + эксперименты с отладчиком, и абсолютно все станет ясно.

1. У разных команд разная длина: от 1 до 6 байтов.

Общий формат команды (6 байтов):

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| КОП | **d** | **w** |  | **mod** | **reg** | **r/m** |  | Смещ. Мл.байт | Смещ. Ст.байт |  | Данные Мл.байт | Данные Ст.байт |

1 байт - КОП – код операции. Два младших бита d, w:

**d** – направление (относительно поля reg постбайта):



**w** – тип операндов (байт или слово):



2-ой байт, или постбайт, определяет операнды, участвующие в команде. Поля постбайта: **mod reg** **r/m,** один операнд кодируется полем **reg,** другой полями: **mod** + **r/m.**

1. **mod** (режим, 2 бита) – определяет, как интерпретировать поле **r/m**;
2. **reg** (регистр, 3 бита) – определяет регистр, участвующий в команде;
3. **r/m** (регистр или память, 3 бита) – совместно с полем **mod** указывает местоположение операнда: если операнд находится в регистре, то в **r/m** указан код регистра, если операнд в памяти, то задается метод адресации.



**2. Справочная информация**

Таблица 1. Коды регистров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер регистра | Рег. общего назначения | | Сегмент.  Регистры | Короткий номер сегментного регистра |
| W = 1 | W = 0 |
| 000 | AX | AL | ES | 00 |
| 001 | CX | CL | CS | 01 |
| 010 | DX | DL | SS | 10 |
| 011 | BX | BL | DS | 11 |
| 100 | SP | AH |  |  |
| 101 | BP | CH |  |  |
| 110 | SI | DH |  |  |
| 111 | DI | BH |  |  |

Таблица 2. Коды методов адресации ( **reg** и **r/m)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Операнд в памяти | | | Операнд в регистре | |
| MOD | 00 | 01 | 10 | 11 | |
| R/M | W = 0 | W = 1 |
| 000 | DS:  (BX)+(SI) | DS:  (BX)+(SI)+D8 | DS:  (BX)+(SI)+D16 | AL | AX |
| 001 | DS:  (BX)+(DI) | DS:  (BX)+(DI)+D8 | DS:  (BX)+(DI)+D16 | CL | CX |
| 010 | SS:  (BP)+(SI) | SS:  (BP)+(SI)+D8 | SS:  (BP)+(SI)+D16 | DL | DX |
| 011 | SS:  (BP)+(DI) | SS:  (BP)+(DI)+D8 | SS:  (BP)+(DI)+D16 | BL | BX |
| 100 | DS:  (SI) | DS:  (SI)+D8 | DS:  (SI)+D16 | AH | SP |
| 101 | DS:  (DI) | DS:  (DI)+D8 | DS:  (DI)+D16 | CH | BP |
| 110 | DS:  D16 | SS:  (BP)+D8 | SS:  (BP)+D16 | DH | SI |
| 111 | DS:  (BX) | DS:  (BX)+D8 | DS:  (BX)+D16 | BH | DI |

Префикс замены сегментного регистра

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | R E G | | 1 | 1 | 0 |
|  |  |  |  |  |  |  |  |

**Вывод**

При выполнении данной работы были получены навыки работы с отладчиком ***Turbo Debugger***, были изучены структуры и форматы команд. Разработана простейшая программа на ассемблере и выполнена в пошаговом режиме.