Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Севастопольский государственный университет»

ИССЛЕДОВАНИЕ МЕТОДОВ АДРЕСАЦИИ И ПРОГРАММИРОВАНИЯ АРИФМЕТИЧЕСКИХ И ЛОГИЧЕСКИХ ОПЕРАЦИЙ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам по дисциплине

Архитектура вычислительных устройств

для студентов, обучающихся по направлению

09.03.02 Информационные системы и технологии

09.03.03 Прикладная информатика

очной и заочной форм обучения

Севастополь
2020

УДК 004.732

Исследование методов адресациии и программирования арифметических и логических операций. Методические указания к лабораторным занятиям по дисциплине "Архитектура вычислительных устройств". Сост. Чернега В.С., Дрозин А.Ю. — Севастополь: Изд-во СевГУ, 2020— 14 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине “Архитектура вычислительных устройств“. Целью методических указаний является помощь студентам в выполнении лабораторных работ по исследованию архитектуры 16-разрядных процессоров и персональных ЭВМ, а также по программированию различных задач на языке ассемблера процессора Intel 8086. Излагаются краткие теоретические и практические сведения, необходимые для выполнения лабораторных работ, примеры составления программ, требования к содержанию отчета.

Методические указания рассмотрены и утверждены на методическом семинаре и заседании кафедры информационных систем
(протокол № 1 от 30 августа 2020 г.)

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

Рецензент: Кротов К.В., канд. техн. наук, доцент кафедры ИС

Лабораторная работа

ИССЛЕДОВАНИЕ МЕТОДОВ АДРЕСАЦИИ И ПРОГРАММИРОВАНИЯ АРИФМЕТИЧЕСКИХ И ЛОГИЧЕСКИХ ОПЕРАЦИЙ

1. ЦЕЛЬ РАБОТЫ

Изучить основные директивы языка ассемблера, исследовать их воздействие на процесс ассемблирования и формирования листинга программы.

Исследовать особенности функционирования блоков 16-разрядного микропроцессора при выполнении арифметических и логических операций и при использовании различных способов адресации. Приобрести практические навыки программирования на языке ассемблера МП 8086 арифметических и логических операций с применением различных способов адресации.

2. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1 Основные директивы ассемблера

Операторы языка "Ассемблер", которые позволяют управлять процессом ассемблирования и формирования листинга называются директивами или псевдокомандами. Они действуют только в процессе ассемблирования и не генерируют машинных кодов. Наиболее часто используются следующие директивы.

PAGE строк, символов в строке — задает формат распечатки программы, т.е. количество строк на листе и число символов в строке. Максимальное количество строк 255, символов ─ 132. По умолчанию в ассемблере установлено PAGE 60,80.

TITLE текст — задает текст, который будет печататься на каждой странице листинга. В качестве текста рекомендуется указывать имя программы, под которым она находится в каталоге на диске.

SEGMENT [параметры] — указывает используемый в программе сегмент. Любые программы содержат по крайней мере один сегмент кода. Имя сегмента должно обязательно присутствовать, быть уникальным и не совпадать со стандартным набором слов, используемых в языке. Для описания сегмента используется следующий формат:

```
Имя  Директива  Операнд   имя SEGMENT  [параметры]
       :
       :
имя   ENDS.
```

Директива SEGMENT может содержать три типа параметров: выравнивание, объединение и класс. Обычно *выравнивание* сегмента осуществляется по параграфам. Поэтому типовым параметром является PARA, который принимается

также по умолчанию. Параметр *объединение* определяет, объединяется ли данный сегмент в процессе компоновки с другими сегментами. Возможны следующие виды объединений: STACK, COMMON, PUBLIC, AT-выражение и MEMORY. Сегмент стека определяется следующим образом:

```
имя SEGMENT  PARA  STACK .
```

Если отдельно ассемблируемые программы должны объединяться компоновщиком, то можно использовать типы объединений: PUBLIC, COMMON и MEMORY. В других случаях тип объединения можно не указывать.

Параметр *класс* заключается в кавычки используется для группирования относительных сегментов при компоновке.

Директива PROC выделяет отдельные процедуры в сегменте. Если сегмент имеет только одну процедуру, то он оформляется следующим образом:

```
имя_сегмента      SEGMENT     PARA
имя_процедуры     PROC        FAR
          :
                  RET
имя_процедуры     ENDP
имя_сегмента      ENDS.
```

Директивы ENDP и ENDS указывают соответственно конец процедуры и сегмента. Операнд FAR сообщает загрузчику DOS, что начало данной процедуры является точкой входа для выполнения программы.

Директива ASSUME операнд - указывает Ассемблеру назначение каждого сегмента. Например:

```
ASSUME  SS: им_стек,  DS:им_дан,  CS:им_код.
```

Запись SS: им_стек означает, что ассемблер должен инициализировать регистр SS адресом им_стек.

Директива END завершает всю программу. Операнд содержит имя, указанное в директиве PROC, которое было обозначено как FAR.

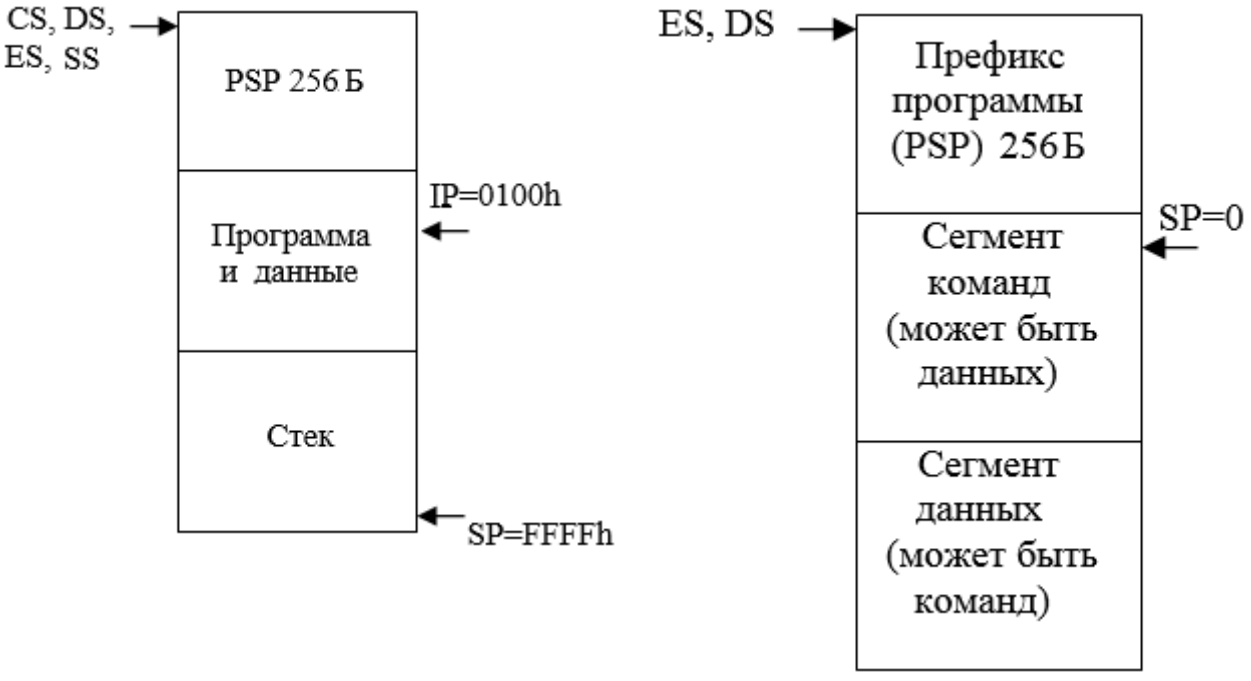
2.2. Размещение EXE- COM- программ в памяти компьютера

При загрузке исполняемой программы в оперативную память, кроме собственно машинных команд программы и обрабатываемых ею данных, она содержит вспомогательную информацию, называемую ***префиксом программного сегмента*** (***PSP*****).** Длина *PSP равна* 256 (*100h*) байтов и эта структура данных находится в начале любой машинной программы, вне зависимости от того, получена ли программа загрузкой *com-* или *exe*-файла. При этом ни *com-*, ни *exe*-файл *PSP* не содержит. Информация в *PSP* записывается загрузчиком и используется ОС при выполнении запросов со стороны программы, а также может быть использована самой программой.

В первых двух байтах *PSP* находится код машинной команды прерывания DOS ***int* 20h – возврат в DOS.** (В нашем случае эта команда вернет управление

в *эмулятор*, так как мы запускаем нашу программу не непосредственно из *DOS*, а из *эмулятора*). В следующих двух байтах *PSP* находится верхняя граница (адрес сегмента) ОП. Обычно она равна величине *A000h*. Затем в следующих пяти байтах *PSP*, начиная с *05h*, находится вызов диспетчера функций *DOS*. Далее в двенадцати байтах *PSP*, начиная с *0Ah*, содержатся копии векторов прерываний с номерами *22h*, *23h*, *24h*. Эти прерывания *DOS* использует для управления выполнением программы. При этом прерывание *22h* применяется для вызова той подпрограммы, которая должна получить управление при завершении прикладной программы. Прерывание *23h* происходит при нажатии клавиш <*Ctrl*>+<*C*>. (*DOS* использует для обработки этой комбинации специальный обработчик.) Причинами прерывания *24h* являются аппаратные ошибки при работе с ПУ. Примерами этих ошибок являются: 1) попытка записи на защищенный диск; 2) нет бумаги на принтере; 3) неизвестное устройство. Хранение копий векторов перечисленных прерываний обусловлено тем, что в случае, если прикладная программа изменяет содержимое векторов (с целью выполнить свою обработку прерываний), то после ее завершения *DOS* восстанавливает прежнее содержимое векторов, используя копии из *PSP*. Кроме этого, в PSP содержится информация, служащая для управления файлами. Часть ячеек памяти зарезервировано.

Схемы размещения программ com- и exe-форматов показана на рисунках 2.1,а и 2.1,б.



а) com-формат б) exe-формат

Рисунок 2.1 – Схема размещения программ в памяти компьютера

Загрузка COM-программы

В процессе загрузки com-программы операционная система выполняет следующие действия:

- сегментные регистры CS, DS, ES, SS устанавливаются на начало PSP;
- регистр SP устанавливается на конец сегмента PSP;
- вся область памяти после PSP выделяется программе;
- в стек записывается слово 0000;
- указатель команд IP устанавливается на 100h (начало программы) с помощью команды JMP по адресу PSP :100h.

Размещение .EXE-программы в памяти

Программа в памяти начинается с префикса программного сегмента (PSP), который образуется и заполняется операционной системой в процессе загрузки программы в память. Затем в памяти располагаются сегменты в том порядке, как они объявлены в тексте программы.

В процессе загрузки в память сегментные регистры автоматически инициализируется следующим образом: ES и DS указывают на начало PSP. Поэтому перед началом выполнения программы нужно сохранить их значения, чтобы после окончания можно было обратиться к PSP.

Регистры CS и SS указывает на начало сегмента команд (DOS делает автоматически). Поскольку сегмент данных оказывается не адресуемым, то необходимо инициализировать регистр DS (и ES).

```
begin: mov ax, data;          data – начало сегмента данных
       mov ds, ax
       push DS
       pop ES
       .
       .
       mov ax, 4c00h;         функция завершения программы
       int 21h;               вызов DOS
```

Пример EXE-программы

При инициализации ассемблерной EXE-программы система DOS предъявляет следующие требования;

1) указать ассемблеру, какие сегментные регистры должны соответствовать конкретным сегментам;
2) сохранить в стеке адрес, содержащийся в регистре DS;
3) записать в стек нулевой адрес;
4) загрузить в регистр DS адрес сегмента данных.

Пример оформления ассемблерной программы:

```
                    Page 60, 132
                    TITLE EXAMPLE (EXE)
;-----------------------------------------
STACKSG SEGMENT PARA STACK    ‘stack’
```

```
; Указывать обязательно, т.к. по ней осуществляется автоматическая инициализация SS и SP
; Если тип объединения STACK, то при компоновке; одноименные сегменты располагаются
; рядом, при COMMON – накладываются друг на друга
            DW 10 DUP(?);
STACKSG ENDS
;-------------------------------------------------------------------------------
DATASG    SEGMENT PARA ‘Data’
            NAME1 DB ‘ASSEMBLERS’;
            NAME2 DB 10DUP(‘ ‘)                ; зарезервировать 10 слов
DATASG ENDS
      ;-------------------------------------------------------------------------
      ; -------------------------------------------------------------------------
CODESG  SEGMENT PARA ‘Code’
; для правильной работы LINK   необходимо указать ‘Code’
BEGIN     PROC FAR

ASSUME CS:CODESG; DS:DATASG, SS:STACKSG, ES:DATASG
; Показывает транслятору, какие регистры закрепляются за сегментами, которые затем
; используются по умолчанию
            PUSH DS                 ;  записать DS в стек
            SUB AX, AX              ;  записать 0 в стек
            PUSH AX
            MOV AX, DATASG      ; регистры CS и SS инициализируются автоматически
            MOV DS, AX
            MOV ES, AX
; ---------------------------------------------------------------------------------------------
            MOV AX, 0123H
            ADD AX, 0025H
            MOV BX, AX
            ADD BX, AX
            SUB AX, AX
            NOP
; ---------------------------------------------------------------------------------------------

            CLD
            LEA SI, NAME1
            LEA DI, NAME2
            MOV CX, 10
      REP MOUSB                 ;  переслать 10 байтов из  NAME1 NAME2
; ---------------------------------------------------------------------------------------------
           RET                  ;  возврат в DOS
BEGIN     ENDP
CODESG ENDS
            END [BEGIN]
; операнд может быть опущен, если эта программа должна быть скомпонована
; с другим (главным) модулем. Для обычных программ операнд содержит имя, указанное в
```

; директиве PROG, (то есть в точку входа в главную процедуру).
; Этим адресом загружается IP.

Пример оформление COM-программы

```
                    Page 60,80
          TITLE EXAMPLE_COM                для пересылки и сложения
Codes     SEGMENT   PARA   'Code'
          ASSUME CS:CODESG, DS:CODESG, SS:CODESG, ES:CODESG
          ORG 100H                                  ; начало в конце PSP
BEGIN: JMP MAIN                                     ; обход через данные
;-------------------------------------------------------------------------
          ALPHA DW 250
          BETA DW 125
          GAMMA DW ?
;-------------------------------------------------------------------------
MAIN   PROC NEAR
          MOV AX, ALPHA
          ADD AX, BETA
          MOV GAMMA, AX
          RET                                       ; вернуться в DOS
                                                    ; можно вместо RET INT20H
MAIN   ENDP
Codes     ENDS
END       BEGIN
```

2.3 Методы адресации микропроцессорных систем

В процессе программирования МП 8086 используется 7 режимов адресации.

1. **Регистровая адресация**. Операнд находится в одном из регистров общего назначения, а в ряде случаев — в сегментном регистре:

```
mov  dx,bx
add   bx,di
```

2. **Непосредственная адресация**. 8- или 16-битовая константа содержится в команде в качестве источника:

```
mov  al,45
mov  ax,1024
```

3. **Прямая адресация**. Эффективный адрес берется из поля смещения команды. Применяется в основном, если операндом служит метка:

```
table  db 10, 20, 30
.............
mov  ax, table
```

Микропроцессор добавляет адрес к сдвинутому на 4 разряда содержимому регистра DS и получает 20-разрядный адрес операнда.

4. **Косвенная регистровая адресация**. Эффективный адрес содержится в базовом регистре BX, регистре указателя базы BP или индексном регистре (DI или SI). Косвенные регистровые операнды заключают в квадратные скобки:

```
mov ax,[bx]
```

По этой команде в регистр AX загружается содержимое ячейки памяти, адресуемой значением регистра BX.

5. **Базовая адресация**. Эффективный адрес вычисляется с помощью сложения значения сдвига с содержимым одного из базовых регистров BX или BP. Например, при доступе к элементам таблицы адрес начала таблицы предварительно записывается в регистр BX:

```
mov ax,[bx]+8
```

6. **Прямая адресация с индексированием**. Эффективный адрес вычисляется как сумма значений сдвига и одного из индексных регистров (DI или SI). Такой способ адресации целесообразно применять при доступе к элементам таблицы, когда сдвиг указывает на начало таблицы, а индекс - на элемент таблицы.

```
table db 10, 20, 30, 40
.............
mov di,2 ; загрузить в индексный регистр
         ;номер выбираемого байта минус 1
mov al,table[di]  ; загрузить 3 байт таблицы в al
```

7. **Базово-индексная адресация**. Эффективный адрес вычисляется как сумма значений базового регистра, индексного регистра и, возможно, сдвига. Это удобно при работе с двухмерными таблицами: базовый регистр содержит начальный адрес массива, значения сдвига и индексного регистра представляют собой смещения по строке и столбцу.

```
table dw  1024, 1048, 2048,3600
      dw  4100, 5000, 600, 2000
      dw  80, 300, 4000, 5000
   .............
value db 2              ; номер элемента в строке-1
   .............
mov bx,table            ;адрес таблицы
mov di,16               ;адрес начала третьей строки
mov ax,value[bx][di] ;загрузка элемента
table(3,3)=4000
```

3. ОПИСАНИЕ ЛАБОРАТОРНОЙ УСТАНОВКИ

Лабораторный стенд для исследования архитектуры и способов программирования на языке ассемблера 16-разрядных микропроцессоров состоит из персонального компьютера, на котором установлен программный эмулятор 16-разрядного микропроцессора типа Intel 8086 (отечественный аналог МП КР1810). Эмулятор отображает на экране персонального компьютера программную модель исследуемого процессора, а также позволяет создавать и редактировать

тексты программ на языке ассемблера МП 8086, выполнять их ассемблирование и исследование процессов модификации регистров процессора, дампов памяти и портов в пошаговом и реальном режимах отладки программ. Работа с эмулятором подробно описана в методических указаниях по предыдущей работе.

4. ПРОГРАММА ИССЛЕДОВАНИЙ

4.1. Изучить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Повторить команды пересылки данных, а также команды арифметических и логических операций (выполняется в процессе домашней подготовки к лабораторной работе).

4.2. Изучить методы адресации, используемые в 16-разрядных процессорах и особенности оформления программ в exe- и com-форматах (выполняется во время домашней подготовке к работе).

4.3. Составит программу, состоящую из следующих процедур обработки строк:

4.3.1. Заполнить 100+10*i* ячеек области памяти, начинающейся с адреса MAS1 рядом натуральных чисел. Здесь i – последняя цифра номера Вашей зачетной книжки.

4.3.2. Переслать массив слов из области памяти, начиная с адреса MAS1 в область с начальным адресом MAS2.

4.3.3. Найти в заданном массиве число, равное двум последним цифрам Вашей зачетной книжки и определить его индекс.

4.3.4. Переслать в память с адресом 2020:300 диагональные элементы матрицы размером 8×8. Значения элементов матрицы должны быть определены в сегменте данных программы.

5. Произвести отладку разработанных программ в пошаговом режиме и проследить за изменениями содержимого регистров

6. Произвести ассемблирование программы и получить объектный и исполняемый модуль программы в Exe-формате и ее листинг.

7. Рассчитать время выполнения программы.

5. СОДЕРЖАНИЕ ОТЧЕТА

4.1 Цель и программа работы.

4.2 Текст и листинг ассемблерной программы для заданного варианта.

4.3 Выводы по работе.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Каково различие между директивой и командой?

6.2. Назовите директивы определения данных ассемблера и поясните механизм их действия.

6.3. Какие директивы применяются для оформления процедур?

6.4. Какие типы сегментов используются в ассемблерных программах и каково их назначение?

6.5. Поясните назначение параметров выравнивания и объединения, используемых в директивах SEGMENT.

6.6. Когда и в каких случаях применяется директива ORG?

6.7. Что конкретно подразумевает директива END, если она завершает: а) программу, б) процедуру, в) сегмент?

6.8. Какие операции необходимо произвести в процессоре до начала выполнения программы?

6.9. Назовите команды арифметических операций и поясните использование регистров процессора при каждой операции.

6.10.С какой целью в начале кодового сегмента в стек заносится содержимое сегментного регистра DS, а затем нулевое значение?

6.11.Каково назначение директивы ASSUME?

6.12.Расскажите об особенностях размещения в памяти ЭВМ программ с расширениями .exe и .com.

6.13.Нарисуйте схему подключения 16- разрядного порта к МП 1810ВМ86, если в наличии имеются только микросхемы 580ВВ55 или 580ВА86.

6.14.Каково назначение вывода M/IO в МП 8086 и нарисуйте схему подключения устройств с его использованием.

6.15.Какая информация хранится в заголовке .exe –программы, его назначение и размер?

6.16.Зачем к исполняемому модулю добавляется префикс программного сегмента, какой его размер и какая информация в нем хранится?

6.17.Расскажите о методах адресации, используемых в МП-системах, и объясните в каких случаях целесообразно использование этих методов?

6.18.Объясните особенности использования строковых команд.

6.19.Каким образом можно изменять направление просмотра строк?

6.20.Чем отличаются команды CMPS и SCAS?

6.21.Как обеспечить ввод данных с группы 16-разрядных портов с максимальной скоростью?

6.22.В чем состоит суть защищенного режима работы процессора и как осуществляется защита?

6.23.Что такое дескриптор сегмента, из каких частей он состоит и как используется при защите памяти?

6.24.Как организуется виртуальная память и как используется дескриптор для ее поддержки?

6.25.Расскажите об архитектуре 16-разрядного процессора второго поколения и приведите его схему.

6.26.Расскажите о регистрах 16-разрядного процессора второго поколения и особенностях их использования в защищенном режиме.

6.27.Расскажите о многозадачном режиме работы процессора, составе и назначении сегмента состояния задачи.

7. СПИСОК РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ

7.1. Абель П. Язык Ассемблера для IBM PC и программирования / Пер. с англ. Ю.В.Сальникова. — М.: Высш. школа, 1992. — 447 с.Новиков Ю.В. Основы микропроцессорной техники: Учебное пособие/Ю.В. Новиков, П.К. Скоробогатов. — М.: Интернет-университет информационных технологий; БИНОМ, 2006. — 359 с.

7.2. Программирование на ассемблере для процессоров персонального компьютера / М.К. Маркелов, Л.С. Гурьянова, А.С. Ишков, А.С. Колдов, С.В. Волков.— Пенза: ПГУ, 2013 .— ISBN 978 -5-94170-537-5 http://rucont.ru/efd/210624?cldren=0

7.3. Федотова Д.Э. Архитектура ЭВМ и систем [Электронный ресурс]: лабораторная работа. Учебное пособие/ Федотова Д.Э.— Электрон. текстовые данные.— М.: Российский новый университет, 2009.— 124 с.— Режим доступа: http://www.iprbookshop.ru/21263

7.4. Чернега В.С. Архитектура информационных систем . Конспект лекций / В.С. Чернега. – Севастополь: Изд-во СевГУ, 2019 – 160 с.

Заказ № _____ от «___» _____ 2020 г. Тираж _____ экз.
Изд-во СевГУ