Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования
«Севастопольский государственный университет»

ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ОБРАБОТКИ СТРОКОВЫХ ДАННЫХ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

Технические средства информационных систем

для студентов, обучающихся по направлению

09.03.02 Информационные системы и технологии

09.03.03 Прикладная информатика

очной и заочной форм обучения

Севастополь

2022

УДК 004.732

Исследование методов программирования обработки строковых данных. Методические указания к лабораторным занятиям по дисциплине "Технические средства информационных систем" / Сост. Чернега В.С., Дрозин А.Ю. — Севастополь: Изд–во СевГУ, 2022— 13 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине “ Технические средства информационных систем “. Целью методических указаний является помощь студентам в выполнении лабораторных работ по исследованию методов программирования с использованием строковых данных на языке ассемблера процессора x8086. 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 2022 г.)

Допущено учебно–методическим центром СевГУ в качестве методических указаний.

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

Лабораторная работа

ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ОБРАБОТКИ СТРОКОВЫХ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Изучить основные команды языка ассемблера для обработки строковых данных и команды передачи управления, исследовать их воздействие на процесс ассемблирования и формирования листинга программы.

Исследовать особенности функционирования блоков 16–разрядного микропроцессора при выполнении команд обработки строк и передачи управления. Приобрести практические навыки программирования на языке ассемблера МП 8086 задач обработки линейных массивов.

2. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1 Программирование с использованием строковых данных

Строкой (String) в языке ассемблера называют набор (цепочку) однотипных символов, байтов или слов. Она обычно заканчивается нулевым символом, который равен « 0». Строковые (цепочечные) команды используются для эффективной обработки массивов.

Для обработки строковых данных ассемблер имеет пять команд обработки строк:

MOVS – переслать один байт или одно слово из одной области памяти в другую;
LODS – загрузить из памяти один байт в регистр AL или одно слово в регистр AX;
STOS – записать содержимое регистра AL или AX в память;
CMPS – сравнить содержимое двух областей памяти, размером в один байт или в одно слово;
SCAS – сравнить содержимое регистра AL или AX с содержимым памяти.

Цепочечная команда может быть закодирована для повторяющейся обработки одного байта или одного слова за одно выполнение в цепочечных командах (для однобайтовых и двухбайтовых вариантов). Для адресации цепочки–источника предусмотрен индексный регистр SI, а для адресации цепочки–приемника – регистр DI. Форматы строковых команд приведены в таблице 2.1.

Таблица 2.1 – Форматы строковых команд

Команда	Операнды	Байт	Слово
MOVS	DI,SI	MOVSB	MOVSW
LODS	AL,SI или AX,SI	LODSB	LODSW
STOS	DI,AL или DI,AX	STOSB	STOSW
CMPS	SI,DI	CMPSB	CMPSW
SCAS	DI,AL или DI,AX	SCASB	SCASW

Регистр SI обычно связан с регистром сегмента данных – DS:SI. Регистр DI всегда связан с регистром дополнительного сегмента – ES:DI. Следовательно, команды MOVS, STOS, CMPS и SCAS требуют инициализации регистра ES (обычно адресом в регистре DS). **COM программа автоматически инициализирует регистры ES и DS.**

Для повторения строковой команды используется префикс REP. Префикс записывается непосредственно перед цепочечной командой, например, REP MOVSB. Для использования префикса REP необходимо установить количество повторений в регистре CX. При выполнении цепочечной команды с префиксом REP происходит уменьшение на 1 значения в регистре CX до нуля. Таким образом, можно обрабатывать строки любой длины.

Для задания направления перемещения по строкам применяется флаг D, который устанавливается командой STD и сбрасывается командой CLD. При нулевом значении флага при каждом повторении происходит увеличение адресов источника и приемника, а при единичном – уменьшение.

Пример. Пусть необходимо выполнить пересылку 20 байт из STRING1 в STRING2. Предположим, что оба регистра DS и ES инициализированы адресом сегмента данных:

```
STRING1     DB   20 DUP('*')
STRING2     DB   20 DUP(' ')
. . . . .
            CLD                 ;Сброс флага DF
            MOV  CX,20          ;Счетчик на 20 байт
            LEA  DI,STRING2     ;Адрес области "куда"
            LEA  SI,STRING1     ;Адрес области "откуда"
REP MOVSB                       ;Переслать данные
```

При выполнении команд CMPS и SCAS возможна установка флагов состояния, так чтобы операция могла прекратиться сразу после обнаружения необходимого условия. Ниже приведены модификации префикса REP для этих целей:

REP – повторять операцию, пока CX не равно 0;

REPZ или REPE – повторять операцию, пока флаг ZF показывает "равно или ноль". Прекратить операцию при флаге ZF, указывающему на не равно или не ноль или при CX равном 0;

REPNE или REPNZ – повторять операцию, пока флаг ZF показывает "не равно или не ноль". Прекратить операцию при флаге ZF, указывающему на "равно или нуль" или при CX равным 0.

В ассемблерных программах вообще, и при работе с цепочками данных широко используются команды передачи управления (команды ветвления).

2.2 Команды передачи управления

Команды передачи управления процессора 8086 подразделяются на команды безусловных переходов, условных переходов, вызовов, возвратов, управления циклами и команды прерываний.

Сегментная организация программной памяти определяет две основные разновидности команд передачи управления. Передача управления в пределах текущего сегмента кода называется внутрисегментной – при этом модифицируется только регистр IP и адрес перехода может быть представлен одним словом. Такая передача управления называется ближней (тип NEAR), а ее вариант с сокращеннным диапазоном адресов переходов – короткой. Передача управления за пределы текущего сегмента кода называется межсегментной или дальней (тип FAR) – при этом необходимо модифицировать содержимое регистров IP и CS и адрес перехода представляется двумя словами (сегмент : смещение).

Команды безусловного перехода имеют общую мнемонику JMP. Команда короткого безусловного перехода содержит во втором байте смещение, которое интерпретируется как знаковое целое. Диапазон значений байта смещения составляет –128 – +127. Если смещение положительное, осуществляется переход вперед, а если отрицательное – переход назад.

Команда ближнего безусловного перехода может либо непосредственно содержать 16–битное смещение, либо косвенный адрес 16–битного смещения. Диапазон смещения составляет –32768 – +32767 байт относительно адреса команды, находящейся после команды JMP. Команда дальнего безусловного перехода реализует прямой и косвенный межсегментнные переходы.

Форматы команд:

JMP disp – ближний прямой переход
JMP mem/reg – ближний косвенный переход
JMP addr – дальний прямой переход
JMP mem – дальний косвенный переход

В системе команд процессора 8086 имеется 19 двухбайтных команд условных переходов. При выполнении этих команд анализируется некоторое условие, закодированное текущими состояниями флагов, и если оно выполняется, то осуществляется переход, а если нет, то выполняется следующая по побядку команда.

Все условные переходы являются короткими. Некоторые команды для удобства программирования могут иметь несколько различных мнемонических обозначения.

Мнемонические обозначения команд:

1) Команды для работы с беззнаковыми числами:
JA/JNBE —— переход, если больше;
JAE/JNB/JNC – переход, если больше или равно;
JB/JNAE/JC – переход, если меньше;
JBE/JNA – переход, если меньше или равно.
2) Команды для работы со знаковыми числами:
JG/JNLE – переход, если больше;
JGE/JNL – переход, если больше или равно;
JL/JNGE – переход, если меньше;
JLE/JNG – переход, если меньше или равно;
JNS – переход, если больше нуля;
JS – переход, если меньше нуля.
3) Команды, общие для знаковых и беззнаковых чисел:
JE/JZ – переход, если равно / переход, если ноль;
JNE/JNZ – переход, если не равно / переход, если не ноль;
JNO – переход, если нет переполнения;
JO – переход, по переполнению.
4) Прочие команды:
JCXZ – переход, если содержимое регистра CX равно нулю;
JNP/JPO – переход при отсутствии четности;
JP/JPE – переход по четности.

Форматы команд такие же, как у короткого безусловного перехода.

Команда вызова подпрограммы CALL передает управление с автоматическим сохранением адреса возврата в стеке. В поле операнда этой команды находится метка первой команды вызываемой подпрограммы.

При переходе к подпрограмме необходимо временно запомнить адрес команды, находящейся после команды CALL. Этот адрес называется адресом возврата. После того, как подпрограмма закончит свои действия, завершающая

ее команда возврата RET передает управление по запомненному адресу возврата. Адрес возврата запоминается в стеке.

Формат команды:

CALL disp - непосредственный ближний вызов;
CALL mem/reg - косвенный ближний вызов;
CALL addr - непосредственный дальний вызов;
CALL mem - косвенный дальний вызов.

Каждая подпрограмма должна содержать минимум одну команду возврата RET, которая возвращает управление вызывающей программе. Такая передача управления осуществляется путем извлечения из стека адреса возврата, включенного в него командой вызова подпрограммы.

Для управления циклами применяются три команды. В них предусматривается использование регистра CX в качестве счетчика цикла.

В поле операнда команд управления циклами находится метка первой команды цикла (8-битовое смещение). Диапазон переходов этих команд составляет -128 - +127 байт от следующей команды.

Команда LOOP производит декремент регистра CX и, если содержимое CX не равно нулю, происходит переход к началу цикла. В противном случае выполняется следующая по порядку команда.

Мнемоники LOOPE/LOOPZ определяют одну и ту же машинную команду, которая производит декремент регистра CX, а затем передает управление в начало цикла, если содержимое CX не равно нулю и ZF=1. В противном случае выполняется следующая по порядку команда.

Мнемоники LOOPNE/LOOPNZ также определяют одну и ту же машинную команду, которая производит декремент регистра CX, а затем передает управление в начало цикла, если содержимое CX не равно нулю и ZF=0. В противном случае выполняется следующая по порядку команда.

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) emu8086. 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.Повторить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Изучить команды строковых операций (выполняется в процессе домашней подготовки к лабораторной работе).

4.2.Изучить команды передачи управления 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. Исследовать программу пересылки массивов, приведенную в приложении. Оптимизировать данную программу за счет использования префикса повторения.

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

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

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

4.3.5. Вычислить сумму элементов массива MAS1, созданном п. 4.3.1.

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

6. Произвести ассемблирование программ и получить объектный и исполняемый модуль программ в Exe–формате и их листинг.

7. Рассчитать время выполнения программ.

5. СОДЕРЖАНИЕ ОТЧЕТА

4.1 Цель и программа работы.

4.2 Текст и листинг ассемблерной программы с комментариями 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. Что представляет собой строка и какие строковые команды использубтся ддля обработки строк?

6.9. В чем состоит отличие использование индексных регистров DI и SI?

6.10. Каким образом задается направление перемещение по строке?

6.11. Как ассемблер определяет что нужно пересылать байт или слово?

6.12. Как определяется количество повторений команды при использовании префикса REP?

6.13. Какие существуют модификации префикса повторения REP?

6.14. В чем состоит особенность выполнения команды передачи управления в пределах одного сегмента памяти и передачи управления между сегментами?

6.15. Назовите команды условного перехода и на основании чего определяется условие передачи управления?

6.16. Какие команды используются для управления циклами?

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

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

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

6.20. Расскажите об архитектуре 16–разрядного процессора второго поколения и приведите его схему.

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

6.22. 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. Макуха В.К.. Микропроцессорные системы и персональные компьютеры: учебник для вузов/В.к. Макуха, В.А. Микерин. – М.: Изд–во Юрайт, 2022. – 156 с. https://www.urait.ru/book/mikroprocessornye–sistemy–i–personalnye–kompyutery–492153
7.3. 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
7.4. Новожилов О.П. Архитектура ЭВМ и систем в 2 Ч. Часть 1. Учебное пособие для академического бакалавриата / О.П. Новожилов. – М.: Изд–во Юрайт, 2019. – 276 с. https://urait.ru/viewer/arhitektura–evm–i–sistem–v–2–ch–chast–1–494314#page/1
7.5. Чернега В.С. Архитектура информационных систем. Конспект лекций / В.С. Чернега. – Севастополь: Изд–во СевГУ, 2019 – 160 с.

Приложение

Программа пересылки массивов

```
; multi–segment executable file template.

data segment

NAME1   DB      'ABCDEFGHI'
NAME2   DB      'JKLMNOPQR'
NAME3   DB      'STUVWXYZ*'
ends


stack segment
   dw   128  dup(0)
ends


code segment

start:
; set segment registers:
   mov ax, data
   mov ds, ax
   mov es, ax

  ; add your code here

       LEA     SI,NAME1        ; Инициализация адресов
       LEA     DI,NAME2        ; NAME1 и NAME2
       MOV     CX,09           ; Переслать 9 символов
B20:
       MOV     AL,[SI]         ; Переслать из NAME1
       MOV     [DI],AL         ;Переслать в NAME2
       INC     SI              ; Следующий символ вNAME1
       INC     DI              ;Следующая позиция в NAME2
       DEC     CX              ; Уменьшить счетчик цикла
       JNZ     B20             ;Счетчик  > 0? Да – цикл
      ; RET                    ;Если счетчик = 0, то выйти из цикла

       LEA     SI,NAME2        ; Инициализация адресов
```

```
        LEA     DI,NAME3    ; NAME2 и NAME3
        MOV     CX,09       ;переслать 9 символов
C20:
        MOV     AL,[SI]     ; Переслать из NAME2
        MOV     [DI],AL     ; Переслать в  NAME3
        INC     DI          ;следующий символ в NAME2
        INC     SI          ; следующая позиция в NAME3
        LOOP    C20         ;уменьшить счетчик,
                            ; если не ноль, то цикл
                            ; если счетчик 0, то вернуться
         mov ax, 4c00h ; exit to operating system.

        int 21h
ends

end start ; set entry point and stop the assembler.
```