Федеральное агентство по образованию Российской Федерации

Ульяновский государственный технический университет

МАШИННО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам
для студентов специальности
220100 «Вычислительные машины, комплексы, системы и сети»

Составитель В.Н.Негода

Ульяновск 2003

УДК 681.3(076)
ББК 32.973-01я7
М38

Рецензент
кандидат технических наук, доцент, декан ФИСТ В.В.Шишкин

Одобрено секцией методических пособий научно-методического совета университета

М38 Машинно-ориентированное программирование. Методические указания по лабораторным работам для студентов специальности 220100 / Сост. В.Н.Негода. — Ульяновск: УлГТУ, 2003. — 32 с.

Написаны в соответствии с рабочей программой дисциплины «Машинно-ориентированное программирование» для студентов третьего курса специальности «Вычислительные машины, комплексы, системы и сети». Представлены варианты индивидуальных заданий, требования к порядку проведения лабораторных работ и содержанию отчетов, а также справочные данные по архитектуре Intel 80x86.

Подготовлены на кафедре «Вычислительная техника».

УДК 681.3(076)
ББК 32.973-01я7

Введение

Рабочая программа учебной дисциплины «Машинно-ориентированное программирование» для студентов специальности 220100 «Вычислительные машины, комплексы, системы и сети» предусматривает выполнение ряда лабораторных работ по освоению технологии разработки программ на языке ассемблера. В рамках каждой лабораторной работы студент самостоятельно выполняет индивидуальное задание, заключающееся в разработке ассемблер-программы относительно невысокой сложности.

Задания, представленные в данных методических указаниях, направлены на то, чтобы студент осмыслил механизмы представления и обработки наиболее распространенных типов данных средствами самого компьютера без использования языков программирования высокого уровня. Лабораторные задания охватывают процессы преобразования данных между внутренним и внешним представлениями, обработку текстов, чисел, массивов, структурированной информации и моделирование функциональных узлов вычислительной техники. При этом активно используются знания и умения, получаемые в курсах «Информатика» и «Схемотехника».

При выполнении лабораторных заданий приходится использовать большой объем справочных данных по форматам и системе команд, способам адресации и программированию на ассемблере, а также базовым механизмам представления различных структур данных в ассемблер-программе и в программно-доступных компонентах вычислительной машины. Эти данные предоставляются студентам в электронном виде и фигурируют в литературе, список которой приводится в конце данных методических указаний. Предельно лаконичное представление основных справочных данных по архитектуре Intel 80x86 приведено в последнем разделе методических указаний. Студенту рекомендуется в самом начале учебного курса взять на вооружение эти материалы, с тем чтобы свести к минимуму потери времени, связанные с недостатком справочных данных и/или неумением ими пользоваться.

Варианты заданий, фигурирующие в данных методических указаниях, служат основой для формирования постановок задач не только для лабораторных исследований, но и для практической части экзаменационных билетов. В этой связи студенту будет весьма полезно в ходе самостоятельных занятий поработать с задачами за пределами своих вариантов.

Лабораторная работа 1

Основы технологии программирования на ассемблере

Цель работы

Приобретение умений и навыков выполнения основных технологических операций, используемых при программировании на ассемблере.

Порядок работы и содержание отчета

1. Изучение технических документов, выдаваемых для знакомства с инструментальными средствами программирования на ассемблере и опций запуска программ компиляции и компоновки. *В отчете должно быть описание содержания основных операций циклического процесса «редактирование–ассемблирование–компоновка–выполнение» и основных опций запуска программ ассемблерования и компоновки.*

2. Анализ исходных модулей, выдаваемых для изучения технологии программирования на ассемблере.

3. Выполнение макрообработки и анализ результатов макрорасширения.

4. Ассемблирование и анализ листинга программы MOPL1.

5. Трансляция модуля библиотечных функций MOPL1L.ASM.

6. Компоновка модулей MOPL1.OBJ, MOPL1L.OBJ и анализ карты загрузки.

7. Запуск программы в среде DOS. Анализ результатов ее работы и фиксация объемов объектных и загрузочного модулей.

8. Разработка bat-файла MOPL1.BAT, обеспечивающего повторение процесса «редактирование-ассемблирование-компоновка-выполнение» с возвратом к редактированию при обнаружении ошибок ассемблирования или компоновки, а также после выхода из программы с ненулевым кодом возврата.

9. Повторение пунктов 4-6 таким образом, чтобы в результаты трансляции и компоновки была помещена отладочная информация. Фиксация объемов объектных и загрузочного модулей и сравнение их с теми значениями, что имели место при выполнении пункта 7. *В отчете должны быть приведены объемы этих модулей.*

10. Запуск программы в среде отладчика. Выполнение отладки с использованием пошагового и автоматического режимов. Установка и отмена точек останова. Выполнение слежения за переменными.

11. Выполнение трех типов оптимизации программы — по размеру получаемого загрузочного модуля, по размеру исходного текста и по быстродействию выделенного фрагмента.

12. Модификация программы в соответствии с индивидуальным заданием и ее отладка. *В отчете должны быть тексты модифицированных исходных модулей с комментариями, листинги и карты загрузки, полученные в результате ассемблирования и компоновки, а также описание процесса отладки модифицированного кода.*

13. Составление отчета.

Примечание: пункты 4-6 первый раз выполняются таким образом, чтобы в результатах трансляции и компоновки не было отладочной информации.

Общие соглашения

1. Во всех вариантах заданий требуется добавить в программу функции ввода двух двадцатиразрядных двоичных чисел, вычисления заданного выражения и вывода результата.

2. Все умножения и деления на константы, кратные степени двойки, необходимо выполнять командами сдвига.

3. Можно использовать только машинные команды, работающие с восьми и 16-разрядными операндами. Данные представляются в дополнительном коде.

4. В списке вариантов заданий применяются следующие обозначения:

X, Y, Z — двадцатиразрядные числа в дополнительном коде;

x_n — n-й разряд числа X, рассматриваемый как значение логической переменной; $n = 0$ соответствует младшему разряду;

& — операция логического И над битами; знак операции может быть опущен; например, $x_1\, x_2$ означает $x_1 \,\&\, x_2$;

$|$ — операция логического ИЛИ над битами, имеющая меньший приоритет по сравнению с И;

$\bar{x}_n$ — операция НЕ над битом x_n;

$|=$, $\&=$ — операции ИЛИ и И с присваиванием результата левому операнду;

$v = q\ ?\ a1 : a2$ — условный оператор присваивания в стиле языка Си.

Варианты заданий

1. $Z = f_1 ?\ X/4 + 4 * Y : X/8 - Y; z_3 = \bar{z}_3; z_2 \,|= \ z_{19}; z_7 \,\&= \ z_8;$

   $f_1 = x_1\, x_2\, x_3 \,|\, x_2 \bar{x}_3\, x_4 \,|\, \bar{x}_1\, \bar{x}_2 \,|\, x_1\, \bar{x}_2\, x_3\, \bar{x}_4 \,|\, x_3\, x_4.$

2. $Z = f_2 ?\ X * 2 - X/8 : Y/2 + X * 4; z_5 = \bar{z}_2; z_8 \,|= \ z_9; z_{17} \,\&= \ z_0;$

   $f_2 \ = x_1\, \bar{x}_2 \,|\, x_1\, \bar{x}_2\, x_3 \,|\, x_2\, x_3 \,|\, \bar{x}_1\, x_2\, \bar{x}_3.$

3. $Z = f_3?\ X/2 + 2 * Y : X * 4 - Y; z_6 \mid\!= z_3; z_1 8 = \bar{z}_{19}; z_7 \,\&\!= z_8;$
   $f_3 = x_2\ x_4 \mid x_1\ \bar{x}_3 \mid \bar{x}_1\ x_3\ \bar{x}_4 \mid x_3\ x_4 \mid \bar{x}_1\ \bar{x}_2\ x_3.$
4. $Z = f_4?\ X * 8 - Y * 8 : X/8 + Y/8; z_5 \mid\!= z_8; z_4 = \bar{z}_1 1; z_{10}\ \&\!= z_{17};$
   $f_4 = x_1\ \bar{x}_2\ x_3 \mid x_1\ \bar{x}_3\ \bar{x}_4 \mid \bar{x}_1\ \bar{x}_2\ \bar{x}_3\ x_4 \mid \bar{x}_2\ x_3\ x_4 \mid x_1\ \bar{x}_2\ \bar{x}_4.$
5. $Z = f_5?\ X + Y * 2 : X/2 - Y; z_6 = \bar{z}_2; z_{16}\ \&\!= z_{12}; z_{11} \mid\!= z_{13};$
   $f_5 = x_1\ x_2 \mid \bar{x}_2\ \bar{x}_3 \mid x_1\ \bar{x}_2\ x_3 \mid \bar{x}_1\ \bar{x}_3 \mid \bar{x}_1\ x_2\ x_3.$
6. $Z = f_6?\ X * 2 - Y : X/8 + Y; z_7 \& = z_4; z_9 \mid\!= z_{11}; z_{15} = \bar{z}_{17};$
   $f_6 = \bar{x}_1\ \bar{x}_2\ x_3 \mid x_1\ x_3 \mid x_2\ \bar{x}_3 \mid x_2\ x_4 \mid x_1\ \bar{x}_3\ \bar{x}_4.$
7. $Z = f_7?\ X/2 + Y * 2 : X * 8 - Y * 2; z_8 \mid\!= z_3; z_{14} = \bar{z}_{13}; z_{19}\ \&\!= z_7;$
   $f_7 = \bar{x}_2\ \bar{x}_3 \mid x_1\ x_2\ x_3 \mid x_1\ \bar{x}_3 \mid \bar{x}_1\ \bar{x}_2\ x_3 \mid \bar{x}_1\ x_2.$
8. $Z = f_8?\ X * 4 - Y : X/4 + Y/2; z_3 = \bar{z}_8; z_{15}\ \&\!= z_{16}; z_{11} \mid\!= z_8;$
   $f_8 = x_1\ x_2 \mid x_3\ x_4 \mid \bar{x}_1\ \bar{x}_2\ x_3 \mid x_2\ \bar{x}_3\ \bar{x}_4 \mid x_2\ x_3.$
9. $Z = f_9?\ X/4 + Y * 4 : X * 4 - Y * 4; z_2\ \&\!= z_8; z_4 \mid\!= z_{14}; z_{15} = \bar{z}_{10};$
   $f_9 = x_1\ x_2 \mid x_2\ x_3 \mid \bar{x}_1\ \bar{x}_4 \mid x_1\ \bar{x}_2\ x_3.$
10. $Z = f_{10}?\ X * 8 - Y/4 : X + Y/4; z_5 \mid\!= z_7; z_{12} = \bar{z}_{11}; z_{13}\ \&\!= z_{15};$
    $f_{10} = x_1\ \bar{x}_2\ \bar{x}_4 \mid \bar{x}_1\ x_2 \mid x_1\ x_3 \mid x_2\ x_3\ x_4 \mid x_1\ x_4.$
11. $Z = f_{11}?\ X/8 + Y * 8 : X * 2 - Y * 8; z_7 = z_8; z_{11}\ \&\!= z_{14}; z_{18} \mid\!= z_{10};$
    $f_{11} = x_1\ \bar{x}_2\ x_3 \mid x_2\ \bar{x}_3\ x_4 \mid x_3\ \bar{x}_4 \mid x_1\ x_4 \mid \bar{x}_2\ \bar{x}_3.$
12. $Z = f_{12}?\ X - Y/8 : X + Y/8; z_4\ \&\!= z_3; z_{11} \mid\!= z_{13}; z_8 = \bar{z}_9;$
    $f_{12} = x_1\ x_3\ x_4 \mid x_1\ \bar{x}_2 \mid \bar{x}_3\ \bar{x}_4 \mid x_2\ x_4 \mid \bar{x}_1\ x_2\ \bar{x}_4.$
13. $Z = f_{13}?\ X + X/8 : Y * 2 - Y/8; z_{10} \mid\!= z_{11}; z_9 = \bar{z}_8; z_7\ \&\!= z_4;$
    $f_{13} = x_1\ x_3 \mid x_2\ \bar{x}_4 \mid \bar{x}_1\ \bar{x}_2\ x_4 \mid \bar{x}_1\ \bar{x}_3\ x_4 \mid \bar{x}_1\ \bar{x}_4.$
14. $Z = f_{14}?\ X * 2 + X * 8 : X/8 + Y * 8; z_8 = \bar{z}_5; z_4\ \&\!= z_2; z_1 \mid\!= z_0;$
    $f_{14} = x_1\ x_2\ x_5 \mid x_1\ \bar{x}_3\ x_4 \mid \bar{x}_2\ x_3\ \bar{x}_4\ x_5 \mid x_1\ \bar{x}_3\ \bar{x}_5 \mid \bar{x}_1\ \bar{x}_2\ \bar{x}_3\ x_4.$
15. $Z = f_{15}?\ X/2 - X/8 : X/8 - Y/4; z_0\ \&\!= z_{15}; z_{14} \mid\!= z_{18}; z_1 = \bar{z}_{19};$
    $f_{15} = x_1\ x_2\ x_4 \mid x_1\ x_3 \mid x_2\ \bar{x}_4 \mid \bar{x}_1\ \bar{x}_3\ x_4 \mid \bar{x}_1\ \bar{x}_2.$
16. $Z = f_{16}?\ X * 4 + Y * 2 : X/8 + Y * 4; z_{17} \mid\!= z_{16}; z_{13} = \bar{z}_{15}; z_{11}\ \&\!= z_9;$
    $f_{16} = x_1\ x_2\ \bar{x}_3 \mid \bar{x}_2\ x_3 \mid \bar{x}_1\ \bar{x}_2 \mid \bar{x}_2\ \bar{x}_3.$
17. $Z = f_{17}?\ X/4 + Y * 2 : X/8 - Y/2; z_7 = \bar{z}_4; z_2\ \&\!= z_8; z_6 \mid\!= z_4;$
    $f_{17} = x_1\ x_3 \mid x_2\ x_3 \mid \bar{x}_1\ \bar{x}_3 \mid \bar{x}_1\ \bar{x}_2\ x_3 \mid x_1\ x_2\ \bar{x}_3.$
18. $Z = f_{18}?\ X * 8 - X * 2 : X/8 + Y * 2; z_2\ \&\!= z_1; z_3 \mid\!= z_7; z_5 = \bar{z}_{17};$
    $f_{18} = x_1\ x_2 \mid \bar{x}_1\ \bar{x}_3 \mid \bar{x}_1\ \bar{x}_2\ \bar{x}_4 \mid \bar{x}_1\ \bar{x}_2\ x_3\ x_4 \mid x_1\ x_4.$
19. $Z = f_{19}?\ X/8 + X/2 : X/8 - Y; z_{12} \mid\!= z_{11}; z_{13} = \bar{z}_{15}; z_{17}\ \&\!= z_0;$
    $f_{19} = x_1\ \bar{x}_2\ x_4 \mid \bar{x}_3\ \bar{x}_1\ x_4 \mid \bar{x}_1\ x_2\ x_3\ \bar{x}_4 \mid x_1\ \bar{x}_3\ x_4 \mid \bar{x}_1\ \bar{x}_2.$

20. $Z = f_{20}?\, Y + X/8 : X/8 - Y * 2; z_{12} = \bar{z}_{11}; z_{10}\ \&= z_5; z_4 \mid= z_{14};$
$f_{20} = \bar{x}_1\, \bar{x}_2\, x_3 \mid x_1\, x_2\, x_3 \mid \bar{x}_1\, x_2\, x_3 \mid x_1\, \bar{x}_2\, x_3.$

21. $Z = f_{21}?\, Y * 2 - X * 8 : X * 8 + Y/2; z_5\ \&= z_{13}; z_{11} \mid= z_8; z_9 = \bar{z}_{13};$
$f_{21} = x_1\, \bar{x}_2\, x_3\, x_4 \mid \bar{x}_1\, x_2\, \bar{x}_3\, x_4 \mid x_1\, x_2\, x_3\, \bar{x}_4.$

22. $Z = f_{22}?\, Y/2 + X/4 : X/4 - Y * 4; z_0 \mid= z_1; z_7 = \bar{z}_6; z_{12}\ \&= z_{14};$
$f_{22} = x_1\, x_2\, x_3 \mid \bar{x}_1\, x_2\, x_3\, x_4 \mid \bar{x}_1\, x_2\, x_3\, x_4 \mid x_1\, \bar{x}_2\, \bar{x}_3\, \bar{x}_4.$

23. $Z = f_{23}?\, Y * 4 - X * 4 : X * 4 + Y/4; z_7 = \bar{z}_5; z_4\ \&= z_2; z_{11} \mid= z_{10};$
$f_{23} = \bar{x}_1\, \bar{x}_2\, x_3\, \bar{x}_4 \mid x_1\, x_2\, x_3 \mid x_1\, x_2\, x_3\, x_4 \mid \bar{x}_1\, x_2\, x_3\, \bar{x}_4.$

24. $Z = f_{24}?\, Y/4 + X * 2 : X/2 - Y * 8; z_1\ \&= z_0; z_3 \mid= z_4; z_8 = \bar{z}_{19};$
$f_{24} = \bar{x}_1\, x_2\, x_3\, x_4 \mid x_1\, x_2\, x_3\, x_4 \mid x_1\, \bar{x}_2\, \bar{x}_3 \mid x_1\, x_2\, x_3\, x_4.$

25. $Z = f_{25}?\, Y * 8 - X/2 : X * 2 + Y/8; z_3 \mid= z_4; z_8 = \bar{z}_{10}; z_{11}\ \&= z_{16};$
$f_{25} = x_1\, x_2\, x_3\, x_4 \mid x_1\, x_2\, x_3\, \bar{x}_4 \mid x_1\, \bar{x}_2\, \bar{x}_3\, x_4 \mid x_1\, x_2\, \bar{x}_3.$

26. $Z = f_{26}?\, Y/8 + X : X/8 - Y; z_2 = \bar{z}_0; z_{12}\ \&= z_{11}; z_9 \mid= z_{13};$
$f_{26} = x_1\, \bar{x}_2\, \bar{x}_3\, x_4 \mid x_1\, x_2\, x_3\, x_4 \mid x_1\, x_2\, \bar{x}_3\, x_4 \mid x_1\, x_2\, x_3\, \bar{x}_4.$

27. $Z = f_{27}?\, Y - X * 8 : X * 8 + Y * 8; z_{19}\ \&= z_{16}; z_{15} \mid= z_{16}; z_{11} = \bar{z}_{10};$
$f_{27} = \bar{x}_1\, x_3 \mid \bar{x}_1\, x_2 \mid x_1\, x_3\, x_4 \mid \bar{x}_1\, x_2\, x_3 \mid \bar{x}_1\, \bar{x}_2.$

28. $Z = f_{28}?\, Y * 16 + X/4 : X * 4 + Y/8; z_5 \mid= z_4; z_8 = \bar{z}_{11}; z_{11}\ \&= z_{17};$
$f_{25} = \bar{x}_1\, \bar{x}_2\, x_3\, x_4 \mid x_1\, x_2\, x_3\, x_4 \mid x_1\, \bar{x}_2\, \bar{x}_3\, x_4 \mid x_1\, x_2\, \bar{x}_3.$

29. $Z = f_{29}?\, Y/4 - X : X/16 - Y; z_{12} = \bar{z}_0; z_{11}\ \&= z_{14}; z_9 \mid= \bar{z}_{17};$
$f_{29} = \bar{x}_1\, \bar{x}_2\, \bar{x}_3\, x_4 \mid \bar{x}_1\, x_2\, x_3\, x_4 \mid x_1\, x_2\, x_3\, x_4 \mid x_1\, \bar{x}_2\, x_3\, \bar{x}_4.$

30. $Z = f_{30}?\, Y/8 + X * 8 : X * 8 + Y * 4; z_{16}\ \&= z_{15}; z_{15} \mid= z_{13}; z_{10} = \bar{z}_{19};$
$f_{30} = \bar{x}_1\, \bar{x}_3 \mid \bar{x}_1\, \bar{x}_2 \mid x_1\, x_3\, x_4 \mid \bar{x}_8\, x_2\, x_3 \mid \bar{x}_1\, x_2\, x_5.$

Лабораторная работа 2

Разработка программ обработки текстов и целых чисел

Цель работы

Изучение методов представления текстовой информации и целых чисел различной разрядности в программах на языке ассемблера и памяти машины. Освоение программно-технических приемов организации доступа к строкам, их частям и совокупностям. Приобретение умения программировать на ассемблере обработку текстовой информации и ввод-вывод целых чисел в различных системах счисления.

Порядок работы и содержание отчета

1. Анализ индивидуального задания и разработка способов представления объектов задачи в памяти и методов доступа к ним. *Текст постановки задачи должен быть включен в отчет.*

2. Разработка программы на языке ассемблера. *В отчете должно быть изложение способов представления объектов задачи в памяти и методов доступа к ним, а также текст программы с комментариями.*

3. Разработка контрольных примеров. *Описание и обоснование контрольных примеров должно быть включено в отчет.*

4. Отладка программ. *Описание процесса отладки включается в отчет.*

5. Составление отчета.

Общие соглашения

1. Программы всех заданий вводят текст со стандартного ввода *stdin* и выводят на стандартный вывод *stdout*.

2. Тестирование должно быть организовано двумя способами: а) на стандартный вход перенаправляется текстовый файл входных данных любого контрольного примера; б) свободно формируемые входные данные вводятся с терминала.

3. Если входные данные представляют собой последовательность слов, располагаемых в строках, то по умолчанию в качестве разделителя слов выступает один или несколько пробелов.

4. Если во входном тексте требуется распознавать числа в заданной системе счисления, то по умолчанию числа считаются беззнаковыми и их разрядность ограничена возможностью разместить число в 16-разрядном слове.

Варианты заданий

1. Программа вводит до 10 строк текста, рассматриваемого как последовательность слов. Все слова помещаются в массив, где каждое слово завершается байтом 0. Формируется массив адресов каждого слова. Пользуясь массивом адресов, выполняется вывод каждого слова на отдельной строке. Перед словом в шестнадцатиричном виде выводится адрес элемента массива, затем через двоеточие само слово. Все слова, которые могут быть интерпретированы как восьмиричные числа, подвергаются суммированию и сумма выводится в восьмиричном виде после списка входных слов. Число входных слов от 1 до 25.

2. Программа вводит до 10 строк текста. Каждому слову приписывается индекс, равный либо числовому значению самого слова, если оно может быть проинтерпретировано как восьмиричное число, либо коду первого символа, если такая интерпретация невозможна. Список слов выводится в порядке

возрастания индексов. Перед каждым словом выводится конструкция «НОМЕР:ИНДЕКС: », где НОМЕР — десятичный номер слова в исходном списке при нумерации от единицы, ИНДЕКС — двоичное представление значения индекса слова.

3. Программа вводит ключевое слово. Затем вводится последовательность слов. Выполняется поиск ключевого слова в этой последовательности. В зависимости от результатов поиска выводится одно из двух сообщений: «Не найдено» либо «Слово номер N», где N — десятичный номер слова во входной строки при нумерации от единицы. Если ключевое слово представляет собой двоичное число, то во входной последовательности ищется его числовой эквивалент, но представленный в шестнадцатиричном коде. Например, ключевое слово 111111011 считается равным слову 1FB или слову 1fb. Количество слов во входной строке от 1 до 30. Длина ключевого слова от 1 до 10 символов.

4. Программа вводит строку текста. Для каждой буквы, за исключением разделительных пробелов, вычисляется число ее вхождений в текст. Выводится гистограмма вхождений букв в текст. Любое слово, которое может интерпретироваться как восьмиричное число, исключается из процесса построения гистограммы вхождений букв. Для таких слов строится гистограмма чисел. Гистограмма букв, выводится первой и содержит столько строк, сколько различающихся букв имеет текст. Формат строки:
<символ>:<код символа>:<число вхождений><пробел> **...
Число символов '*' равно числу вхождений символа во введенную строку. Число вхождений выводится в десятичном виде. Код символа - в восьмиричном виде. Входная строка может иметь длину до 80 символов.
Формат строки гистограммы чисел имеет вид:
<число>:<число вхождений><пробел> ++...
Количество символов '+' равно количеству вхождений числа во введенную строку. Само число, начинающее строку гистограммы, выводится в шестнадцатеричном виде, а число вхождений — в десятичном виде.

5. Программа вводит строку текста и выводит все слова, для которых чтение слева направо совпадает с чтением справа налево, за исключением слов, которые могут быть интерпретированы как шестнадцатиричные числа, возможно имеющие знак '-'. Выходные слова выводятся по одному на каждой строке. Для чисел выполняется суммирование и сумма всех чисел выводится в шестнадцатиричном виде на отдельной строке после вывода симметричных слов.

6. Программа вводит несколько строк, количество которых не более 24. Среди этих строк могут быть совпадающие. Количество разделительных пробелов между словами не влияет на результат сравнения. Важна одинаковая последовательность слов. Возможно несколько совпадающих строк. После

ввода выполняется вывод, в котором из каждой группы совпадающих строк участвует только по одной строке. Если какое-либо слово одной строки может быть интерпретировано как восьмиричное число, а соответствующее ему слово другой строки — как равное ему по значению шестнадцатиричное число, то эти слова считаются равными. В начало каждой выходной строки вставляется конструкция вида «N_1:N_2: », где N_1 — десятичный номер строки выходного текста, N_2 — десятичный номер этой строки во входном тексте. Нумерация от единицы.

7. Вводятся две строки и выполняется их сравнение. Равными считаются строки, имеющие одинаковую последовательность слов, но число разделяющих слова пробелов или горизонтальных табуляций может быть различным. Если какое-либо слово первой строки может быть интерпретировано как десятичное число, а соответствующее ему число второй строки — как шестнадцатиричное и эквивалентное по значение первому, то слова считаются равными. Например, строка «Это контрольная строка 248» считается равной строке «Это контрольная строка F8». Результат сравнения сообщается через вывод одного из двух сообщений: «Равны», «Нарушение равенства в слове N», где N — десятичный номер того слова первой строки, для которого обнаружено несовпадение с соответствующим словом второй строки. Число слов в строке — не более 30.

8. Программа вводит строку текста и выводит слова строки в порядке возрастания их длины, размещая каждое слово в отдельной строке выходного текста. Если какие-либо слова могут быть интерпретированы как двоичные числа, то они в первую выходную последовательности не попадают, а выводятся после преобразования в шестнадцатиричный код во второй последовательности в порядке возрастания числовых эквивалентов.

9. Программа вводит строку текста русского языка и выводит три строки: первая строка совпадает с введенной; вторая строка повторяет исходную в части гласных букв, а вместо согласных ставится '*'; третья строка повторяет исходную в части согласных букв, а вместо гласных ставится '#'.
Все слова входной строки, которые могут быть интерпретированы как шестнадцатиричные числа, выводятся в первой строке в исходном виде, во второй строке в двоичном виде, а в третьей строке - в десятичном.

10. Программа вводит две строки текста и выводит те слова первой из этих строк, для которых нет совпадающих во второй строке. Слова строк разделяются произвольным числом символов пробела и горизонтальной табуляции. Если какое-либо слово первой строки может быть интерпретировано как восьмиричное число, а соответствующее ему слово второй строки — как равное ему по числовому значению десятичное число, то эти слова считаются равными. После вывода результирующей строки выводится строка вида «N_1, N_2, N_3», где N_1, N_2 — количество слов первой и второй входных строк,

N_3 — количество слов результирующей строки. Числа представляются в десятичном виде.

11. Программа вводит строку текста и выводит множество символов этой строки в порядке убывания числа вхождений каждого символа в строку. Выходное множество символов представляется в формате: «$C_1 : N_1, C_2 : N_2, ...$», где C_i — символ, N_i — число вхождений этого символа, представленное в десятичном виде. Если какие-либо слова строки могут быть интерпретированы как беззнаковые пятиричные числа, то цифры этих чисел не попадают в процесс формирования выходного множества. Сумма всех таких чисел выводится в двоичном виде во второй строке выходного текста.

12. Программа вводит строку из нескольких слов, разделенных одним или более пробелами, и выводит множество символов, равное пересечению множеств символов двух выбранных из строки слов. Выбор осуществляется следующим образом. Если какое-либо слово может быть интерпретировано как шестнадцатиричное число, то оно рассматривается в качестве номера следующего слова. Для построения пересечения множеств символов выбираются слова, имеющие самое наименьшее и самое наибольшее значение номера. Если в строке нет ни одной пары <шестнадцатиричное число — слово>, то выбираются два самых длинных слова. Если есть только одна пара <шестнадцатиричное число — слово>, то в качестве второго слова выбирается самое длинное из оставшихся, т.е. не вошедших в эту пару. Результат выводится в следующем формате: 1) $C_1 : N_1$, 2) $C_2 : N_2, ...$, где C_i — символ, N_i — восьмиричный код символа C_i.

13. Программа вводит два слова управляющей строки, разделенных одним или более пробелом, затем вводит обрабатываемую строку из нескольких слов. Производится преобразование и вывод этой строки таким образом, что в ней выделяется фрагмент и любое вхождение первого слова управляющей строки в этом фрагменте заменяется вторым словом. Фрагмент выделяется следующим образом. Между некоторыми словами может находится специальная метка, представляющая собой двоичное число. Фрагмент выбирается таким образом, чтобы внутри него не было других меток, кроме крайних, и чтобы среди всех подобных фрагментов, выбранный имел максимальную сумму значений меток. В выходной строке метки заменяются на десятичные числа, равные по значению двоичным меткам.

14. Программа вводит строку слов, разделенных одним или более пробелов, и выводит каждое слово на отдельной строке, позиционируя его посередине. По умолчанию ширина строки — 80 символов. Если во входной строке встречается слово, которое может быть интерпретировано как десятичное число, то это слово рассматривается как новое значение ширины, относительно которой производится определение середины. Это число в вывод результата не включается. После вывода всех спозиционированных посередине слов

формируется и выводится строка следующего формата: «N_1, N_2, N_3», где N_1 — количество выведенных слов, N_2 — длина самого короткого слова, N_3 — длина самого длинного слова. Эти числа выводятся в восьмиричном виде.

15. Программа вводит управляющую строку и строку данных. Управляющая строка в качестве первого значащего символа (не пробела) имеет одну из трех букв: 'R','L' или 'M'. Эти буквы задают позиционирование выходной строки согласно правилу: 'R' — прижать к правой границе строки терминала; 'L' — вывести от начала строки терминала; 'M' — вывести посередине. После символа позиционирования в управляющей строке через один или более пробелов фигурирует двоичное значение ширины выходной строки. В строке данных перед выводом с позиционированием удаляются все лишние пробелы таким образом, что остаются только по одному разделяющему пробелу между словами. После вывода строки данных формируется и выводится строка следующего формата: «N%», где N — десятичное значение доли строки данных в ширине строки вывода, выраженное в процентах.

16. Программа вводит строку слов, выводит пронумерованный список слов, располагая каждое слово на отдельной строке в формате: «N. слово», где N — десятичный номер. Затем вводится список номеров выбранных слов, в котором номера представляются в шестнадцатиричном коде, и выводится список отобранных слов в том же формате, что использовался при выводе первого списка, но номер N представляется в восьмиричном виде, причем, это номер слова в исходном списке, а не в результирующем.

17. Программ вводит несколько слов на строке и выводит их на терминал в порядке возрастания специально вычисляемых числовых индексов. Индексом входного слова, которое может быть интерпретировано как десятичное число, является шестнадцатиричный эквивалент этого числа. Индекс любого из остальных слов определяются как многоразрядное шестнадцатиричное число, каждая пара цифр которого представляет код символа. Код левого символа слова образует старшие разряды индекса. Каждое слово упорядоченного списка выводится на отдельной строке в формате: «N :слово», где N — шестнадцатиричное значение индекса. Максимальная длина слова – 10 символов.

18. Программа вводит строку символов, где имеются символы горизонтальной табуляции (ГТ). Затем вводится список десятичных номеров начальных позиций зон табуляции. Выводится строка разметки выходного формата, содержащая в начале каждой зоны табуляции двухразрядный шестнадцатиричный номер соответствующей позиции. Остальные позиции этой строки заполняются символами '-'. Затем выводится входная строка, каждый символ горизонтальной табуляции которой обеспечивает позиционирование курсора выходной строки в начало следующей зоны табуляции. Позиционирование выполняется через вывод пробелов.

19. Программа вводит текст, содержащий слова как латиницы, так и кириллицы. В тексте может быть использована одна из трех кодировок кириллицы: windows-1251, cp-866, koi-8. Программа вычисляет процентный состав групп кодов символов и количество слов, которые могут быть проинтерпретированы как двоичные, восьмиричные и десятичные. Результат выводится в следующем формате:

Общий процент кириллицы	$N_1\%$
в том числе:	
строчных windows-1251	$N_2\%$
прописных windows-1251	$N_3\%$
строчных cp-866	$N_4\%$
прописных cp-866	$N_5\%$
строчных koi-8	$N_6\%$
прописных koi-8	$N_7\%$
Чисел всего	N_8
в том числе:	
двоичных	N_9
восьмиричных	N_{10}
десятичных	N_{11}

Число N_1 для этой таблицы определяет процент от всего входного текста, числа $N_2..N_6$ — проценты от той части кодов текста, что лежит в диапазоне 80h..0FFh. К числам относятся только те слова, которые могут быть преобразованы в формат WORD, т.е. в 16-разрядное внутреннее представление. При этом двоичные числа исключаются из счета восьмиричных и десятичных, а восьмиричные — из счета десятичных.

20. Программа вводит многострочный текст, в котором в неформатированном виде представлено множество векторов длины три. Первый компонент вектора — фраза из одного и более слов, разделенных пробелами. Второй компонент — шестнадцатиричное число, третий компонент — двоичное число. Компоненты во входном тексте разделены запятыми, а вектора — точкой с запятой. После каждого из этих разделителей может быть один или более пробелов. Программы упорядочивает вектора в порядке возрастания второго компонента и выводит их, форматируя так, чтобы каждый вектор располагался на отдельной строке и весь выходной текст образовывал три колонки с выравниванием значений по левой границе. Расстояние между колонками — 2 пробела. Числа во второй и третьей колонках выводятся в десятичном виде.

21. Программа вводит текст, содержащий до 30 строк, анализирует и формирует результаты этого анализа в следующем формате:

Всего символов	N_1
в том числе:	

разделителей	N_2
цифр	N_3
букв	N_4
Всего слов	N_5
в том числе:	
десятичных чисел	N_6
шестнадцатиричных чисел	N_7
аббревиатур	N_8
Всего предложений	N_9

Предложения завершаются символами '.', '?','!'. Десятичными и шестнадцатиричными числами считаются любые слова, которые могут быть корректно преобразованы в знаковое 16-разрядное число внутреннего формата. Аббревиатурами считаются любые слова, содержащие только прописные символы и имеющие длину более 1. Все числа в выходном тексте представляются в десятичном виде. Кодировка входного текста — windows-1251.

22. Программа вводит многострочный текст, завершающийся пустой строкой. Все его слова, которые могут быть интерпретированы как двоичные числа, преобразовываются в шестнадцатиричные. Если в какой-либо строке имеется единственное слово, которое может интерпретироваться как десятичное число из диапазона 12..80, то это число рассматривается как новое значение ширины выводимого текста. Выходной текст выравнивается по ширине так, что каждый фрагмент текста до нового назначения ширины образует прямоугольник. Выравнивание должно выполняться вставкой пробелов, равномерно распределяемых между словами, и при необходимости вставкой дополнительных разделителей строк. Числа, определяющие ширину текста, в выходной текст не включаются.

23. Программа вводит текст описания подпрограммы-функции на языке Си. Заголовок функции содержит описания типов формальных параметров непосредственно в скобках после имени функции. Перед заголовком может следовать комментарий, выделяемый парами символов '/*', '*/'. Введенный текст преобразуется к формату, где описания типов формальных параметров вынесены между закрывающей скобкой списка параметров и фигурной скобкой, открывающей тело процедуры. Все слова комментария, которые могут быть проинтерпретированы как шестнадцатиричные числа, преобразуются в десятичный либо восьмиричный код в зависимости от значения первого символа внутри комментария: 'D' — десятичный, 'C' — восьмиричный.

24. Программа вводит текст ассемблер-программы для архитектуры x86, содержащий только несколько директив объявления данных DW. В содержимом директив могут быть только десятичные, восьмиричные и двоичные константы. Программа формирует выходной текст, представляющий собой объявление тех же самых данных на языке Си. Этот текст должен содержать

один оператор int, в котором все константы для инициализации переменных представлены в десятичном виде.

25. Программа вводит текст объявления группы переменных Си-программы с помощью оператора long. В тексте могут быть инициализированные и неинициализированные скалярные переменные и одномерные массивы. Константы инициализации могут быть десятичными, шестнадцатиричными и восьмиричными. Программа формирует текст последовательности директив ассемблера x86, эквивалентный по смыслу входному тексту. В выходном тексте все константы инициализации должны быть шестнадцатиричными.

26. Программа вводит до 10 операторов ассемблера, содержащих мнемоническое представление команды «INC» I8086 любых допустимых форматов только с регистровой и абсолютной адресацией. В тексте не может быть имен переменных. Константы, задающие абсолютные адреса, могут быть в десятичном, двоичном и шестнадцатиричном коде. Программа формирует листинг трансляции входной последовательности операторов, где результат трансляции представляется в шестнадцатиричном коде.

27. Программа вводит текст, содержащий в каждой строке по одному слову. Если во входном тексте обнаруживаются слова, которые могут быть интерпретированы как беззнаковые восьмиричные числа, то эти числа преобразуются в десятичный код. Выходной текст формируется из полученного в результате такого преобразования путем разворота его на 90 градусов по часовой стрелке. Например, для текста из слов 'Лист', 'лежит', 'на', '177777', на отдельных строках должны быть выведены такие буквосочетания:

```
Л л н 6
и е а 5
с ж   5
т и   3
  т   5
```

28. Программа вводит текст таблицы формата «$N * N$», где N находится в пределах от 2 до 6. В каждой клетке таблицы находится число, заданное в соответствии с требованиями синтаксиса языка Си. Константы могут быть десятичными, шестнадцатиричными и восьмиричными. Числа в строке таблицы разделяются пробелами и горизонтальными табуляциями, между строками нет никаких разделителей. Программа формирует и выводит таблицу формата «$(N+1)*(N+1)$», в последнем столбце и последней строке которой находятся суммы чисел соответствующих строк и столбцов. Все данные выводятся в десятичном виде и выравниваются по правой границе. Например, пусть входная таблица имеет вид:

```
 10   0x10
010    10
```

Тогда на выходе должна быть таблица:

```
 10  16  26
  8  10  18
 18  26  44
```

29. Программа вводит до 10 директив DB ассемблера x86, в которых фигурируют десятичные, шестнадцатиричные, двоичные и восьмиричные константы. По этим данным формируется временная диаграмма. В качестве заголовков сигналов диаграммы используются метки директив DB. Каждый бит двоичного представления числа оператора DB определяет значение сигнала в одном такте диаграммы. Это значение представляется парой символов в соответствии с правилами:
«:-» — 1 после 0,
«:_» — 0 после 1,
«__» — 0 после 0,
«--» — 1 после 1.
Старший разряд двоичного представления любого числа из директивы DB задает сигнал более раннего такта, нежели младший разряд. После вывода диаграммы для сигнала, задаваемого последней директивой DB, выводится ось времени. На этой оси каждому моменту, соответствующему началу такта с номером, кратным 5, выводится десятичное значение этого номера, а остальные такты представляются символом '+'. Номер первого такта равен единице. Значение всех сигналов до этого такта считаются равными нулю. Например, пусть на входе имеется следующая пара директив:

```
ABCD    DB  10, 0E3h, 255q
X12     DB  5, 01101010b,255
```

Тогда выходная диаграмма будет иметь вид:

```
ABCD    ________:-:_:-:_:-----:_____:-----:_:-:_:---:_:-
X12     __________:-:_:-:_:---:_:-:_:-:_:---------------
T       + + + + 5 + + + + 10+ + + + 15+ + + + 20+ + + + 25+ +
```

30. Программа вводит текст, содержащий до 10 операторов присваивания языка Си. В операторах используются однобуквенные идентификаторы переменных типа short, десятичные и шестнадцатиричные константы, а также двухместные операции: «−», «+», «=», «−=», «+=». В правой части оператора присваивания могут быть не более двух операндов. Последовательность операторов такова, что до любого использования переменной происходит ее определение по значению. Программа интерпретирует входной текст и выводит результирующие значения всех переменных в десятичном виде. Например, для входного текста «a = 0x10 + 2; b = 40 - a; a -=b;» выводится:

```
a = -4
b = 22
```

Лабораторная работа 3

Обработка чисел и структурированных данных

Цель работы:

Изучение различных форматов внутреннего и внешнего представления чисел, методов преобразования между этими представлениями, осуществляемого в процедурах ввода и вывода чисел. Приобретение умения программировать обработку числовой информации, массивов и структур на ассемблере.

Порядок работы и содержание отчета

1. Анализ индивидуального задания и разработка способов представления объектов задачи в памяти, методов доступа к ним и способов контроля входных данных. *Отчет должен содержать текст постановки задачи и изложение способов представления объектов задачи в памяти, методов доступа к ним и способов контроля входных данных.*

2. Разработка алгоритма решения задачи. *Алгоритм должен быть включен в отчет.*

3. Разработка программы на языке ассемблера. *Текст программы с комментариями включается в отчет.*

4. Разработка контрольных примеров. *Описание и обоснование контрольных примеров включается в отчет.*

5. Отладка программ. *В отчет включается описание процесса отладки.*

6. Составление отчета.

Общие соглашения

1. Каждое задание предполагает ввод текста со стандартного ввода *stdin* и вывод на стандартный вывод *stdout*.

2. В программе запрещено использовать команды MUL, DIV, IMUL, IDIV, а также команды процессора плавающей точки. В качестве алгоритмов реализации операций умножения и деления должны использоваться традиционные алгоритмы машинной арифметики, базирующиеся на операциях сложения, вычитания, сдвига.

3. Массивы данных должны располагаться в памяти так, чтобы элемент занимал минимально возможную память с учетом диапазона представления данных.

4. Для массивов структур обязательно представление совокупности полей структуры в одном элементе массива, т.е. нельзя использовать представление массива структур через несколько массивов.

5. В задачах, где вводятся числовые данные, необходимо выполнять контроль допустимости этих данных.

6. В задачах, где требуется реализовать арифметические операции, необходимо контролировать ситуации переполнения мантисс, переполнения порядков, исчерпания порядков.

7. При описании входных и выходных данных используются следующие обозначения: m, h, p — цифры мантиссы, характеристики и порядка числа с плавающей точкой; d, f — целая и дробная части числа с фиксированной точкой; L_m, L_h, L_p, L_d, L_f — разрядность соответственно мантиссы, характеристики, порядка, целой и дробной частей числа; $'\pm'$ — символ знака «плюс» или «минус», причем, знак «плюс» может быть опущен; L_A — количество элементов в массиве A. Нижняя граница любой из разрядностей по умолчанию равна 1. По умолчанию нижняя граница размерности массива равна 2, а верхняя — 20 (для двумерных массивов 20*20).

Варианты заданий

1. Программа вводит шестнадцатиричный размер одномерного массива A и текст операторов присваивания вида «$A[I] = N$», где I — восьмиричное значение индекса, N — десятичное значение элемента массива, представляющего собой вещественную константу, имеющую до 15 знаков перед десятичной точкой, и до 3 знаков дробной части. Все числа положительные. Неопределенные элементы массива заполняются нулевыми значениями. Затем вводятся значения индексов I и J и выводится в восьмеричном виде $A[I] + A[J] * A[J + 1]$. Размерность массива не более 100.

2. Программа вводит текст, первая группа строк которого задает значения элементов одномерного массива A. Формат такой строки: «$I : N$», где I — десятичное значение индекса, N — пятиричное представление I-го элемента массива, имеющее до 12 знаков перед десятичной точкой, и до 4 знаков дробной части. Все числа положительные. Неопределенные элементы массива заполняются нулевыми значениями. Вторая группа строк определяет значения индексов I и J с помощью операторов присваивания: «$I = K_1$ и $J = K_2$», где K_1 и K_2 — восьмиричные константы. Программа выводит в двоичном виде $(A[I] + A[J]) * A[I + 1]$.

3. Программа вводит текст, задающий содержимое массива A комплексных чисел через операторы вида «$I.D, M$», где I — десятичное значение индекса, D, M — десятичные значения соответственно действительной и мнимой части числа. Все значения лежат в диапазоне от 0 до 99. Неопределенные элементы массива заполняются нулевыми значениями обеих частей комлексного числа. Затем вводится шестнадцатиричное значение индекса I и выводится в десятичном виде модуль комплексного числа $A[I]$, вычисляемый по формуле $\sqrt{d^2 + m^2}$.

4. Программа вводит в десятичном виде число строк L и число столбцов C матрицы и вводит матрицу M десятичных чисел из диапазона 0-99999999

с помощью операторов вида «$I : J :: N$», где I, J — десятичные значения номеров строк и столбцов матрицы, N — значение элемента. Неопределенные элементы массива заполняются нулевыми значениями. Затем вводятся значения индексов I и J и выводится в восьмиричном виде $M[I, J] * M[J, I]$. Величины L и C лежат в пределах от 2 до 30.

5. Программа вводит не более 30 дат в формате «ДД.ММ.ГГ» и формирует массив A упакованных дат, где в одном 16-разрядном слове размещаются поля: 5 младших разрядов - день, 4 средних разряда - месяц и 7 старших разрядов - год. Затем вводится два индекса I, J и выводится в двоичном виде $A[I]$, $A[J]$ и в десятичном виде количество дней между этими двумя датами.

6. Программа вводит не более 20 строк текста, в последней из которых задается правило обработки всех предыдущих. Правило «$T : L_1, P_1, L_2, P_2$» задает вывод строк от позиции P_1 строки L_1 до позиции P_2 строки L_2. Первая из указанных строк возможно дополняется слева пробелами так, чтобы не изменилось положение выводимых символов строки относительно места, где они вводились. Правило «$R : L_1, P_1, L_2, P_2$» задает вывод прямоугольника между строками L_1, L_2 и колонками R_1, R_2. Параметры L_1, P_1, L_2, P_2 задаются в десятичном виде.

7. Программа вводит массив A не более 20 двоичных чисел в формате «$\pm.m \ldots m\mathrm{E}h \ldots h$», $L_m \leq 12, L_h \leq 11$. Вводятся десятичные значения индексов I, J и выводится десятичная сумма $A[I] + A[J]$ в формате «$\pm.m \ldots m\mathrm{P}\pm p \ldots p$».

8. Программа вводит массив A не более 18 восьмиричных чисел в формате «$\pm.m \ldots m\mathrm{E}h \ldots h$», $L_m \leq 5, L_h \leq 9$. Вводятся значения индексов I, J и выводится десятичная разность $A[I] - A[J]$ в формате «$\pm.m \ldots m\mathrm{E}h \ldots h$».

9. Программа вводит массив A не более 22 двоичных чисел в формате «$\pm d \ldots d.f \ldots f$», $L_d \leq 27, L_f \leq 18$. Вводятся значения индексов I, J и выводится десятичное произведение $(A[I] + 1) * A[J]$ в формате «$\pm.m \ldots m\mathrm{P}\pm p \ldots p$».

10. Программа вводит массив A не более 20 восьмиричных чисел в формате «$\pm d \ldots d.f \ldots f$», $L_d \leq 9, L_f \leq 12$. Вводятся значения индексов I, J и выводится десятичное произведение $(A[I] + 2) * (A[J] - 1)$ в формате «$\pm.m \ldots m\mathrm{E}h \ldots h$».

11. Программа вводит массив A не более 20 шестнадцатиричных чисел в формате «$d \ldots d.f \ldots f$», $L_d \leq 11, L_f \leq 9$. Вводятся значения индексов I, J и выводится десятичное произведение $(A[I] * (A[J] - 1)$ в формате «$\pm.m \ldots m\mathrm{E}h \ldots h$».

12. Программа вводит массив A не более 16 двоичных чисел в формате «$d \ldots d.f \ldots f$», $L_d \leq 35, L_f \leq 25$. Вводятся значения индексов I, J и выводится десятичное произведение $(A[I] - A[J]) * (A[J] - 1)$ в формате «$\pm.m \ldots m\mathrm{E}h \ldots h$».

13. Программа вводит массив A не более 20 десятичных чисел в формате «$d\dots d.f\dots f$», $L_d \le 9, L_f \le 7$. Вводятся значения индексов I, J и выводится двоичное произведение $(A[I] + 0.2) * 3$ в формате «$\pm.m\dots m\mathrm{E}h\dots h$».

14. Программа вводит массив A не более 20 шестнадцатиричных чисел в формате «$d\dots d.f\dots f$», $L_d \le 9, L_f \le 6$. Вводятся значения индексов I, J, K и выводится восьмиричная сумма $A[I] + min(A[J], A[K])$ в формате «$\pm.m\dots m\mathrm{E}h\dots h$».

15. Программа вводит массив A не более 20 десятичных чисел в формате «$d\dots d.f\dots f$», $L_d \le 10, L_f \le 7$. Вводятся значения индексов I, J, K и выводится шестнадцатиричная сумма $A[I] + (A[J] - A[K] * 5)$ в формате «$\pm.m\dots m\mathrm{E}h\dots h$».

16. Программа вводит массив A не более 20 восьмиричных чисел в формате «$d\dots d.f\dots f$», $L_d \le 11, L_f \le 10$. Вводятся значения индексов I, J, сортируется в порядке возрастания и выводится сегмент массива между этими двумя индексами ($A[I], A[J]$ входят в сегмент). Вывод выполняется в десятичном коде в формате «$\pm.m\dots m\mathrm{P}\pm p\dots p$».

17. Программа вводит массив A не более 20 шестнадцатиричных чисел в формате «$d\dots d.f\dots f$», $L_d \le 11, L_f \le 10$. Вводятся значения индексов I, J и выводятся все элементы массива, удовлетворяющие условию $min(A[I], A[J]) \le x \le max(A[I], A[J])$. Вывод выполняется в двоичном коде в формате «$\pm.m\dots m\mathrm{P}\pm p\dots p$».

18. Программа вводит массив $A[0], A[1], \dots, A[N]$ двоичных чисел в формате «$d\dots d.f\dots f$», $L_d \le 34, L_f \le 19$ и равный ему по количеству элементов массив целых беззнаковых восьмиричных чисел B, причем, $B \le N$. Выводятся десятичные значения массива $A[B[0]], A[B[1]] \dots, A[B[N]]$ в формате «$\pm.m\dots m\mathrm{P}\pm p\dots p$». Величина N — не более 30.

19. Программа вводит массив $A[0], A[1], \dots, A[N]$ восьмиричных чисел в формате «$\pm m\dots m.\mathrm{E}h\dots h$», $L_m \le 9, L_h \le 5$. Вводятся значения индексов I, J, K, для которых справедливо $I < J < K$, и выводится десятичное значение суммы $min(A[I], \dots, A[J]) + max(A[J+1], \dots, A[K])$ в формате «$\pm.m\dots m\mathrm{P}\pm p\dots p$». Величина N — не более 30.

20. Программа вводит массив A шестнадцатиричных чисел в формате «$\pm m\dots m.\mathrm{E}h\dots h$», $L_m \le 7, L_h \le 4$ и выводит десятичное значение среднего арифметического в формате «$\pm.m\dots m\mathrm{P}\pm p\dots p$».

21. Программа вводит текст фрагмента Си-программы, где объявляется массив целых чисел m с инициализацией значений, а затем следует оператор вида «$pri(m[k_1] \pm m[k_2] \pm m[k_3], d)$;», где $pri(x, d)$ — функция распечатки числа x в системе счисления d; d — 2, 10 или 16; k_i — константа; $\pm$ — либо «+», либо «-». Программа должна выполнить интерпретацию функции pri, выводя значение выражения в заданной системе счисления. Инициализация элементов массива m выполняется десятичными константами.

22. Программа вводит массив беззнаковых шестнадцатеричных чисел и сжимает этот массив путем размещения нескольких чисел в одном 32-разрядном машинном слове. Среди вводимых чисел находится максимальное, длина двоичного представления которого используется в качестве длины сегментов, из которых формируются упакованные последовательности чисел. В первом 32-разрядном слове результирующего массива младший байт задает длину сегмента, а остальные 3 байта — размер исходного массива, т.е. число сегментов в результирующем массиве. Результат упаковки выводится в двоичном виде.

23. Программа вводит объявления двух массивов 16-разрядных чисел в формате:

```
X   DW  X_0,X_1,\ldot ,X_N
Y   DW  Y_0,Y_1,\ldot ,Y_M
```

В этих операторах X_i, Y_i — двоичные константы. Каждый из массивов кодирует сжатый массив чисел так, что все его числа рассматриваются как последовательность двоичных разрядов. Первый байт последовательности задает число элементов исходного массива чисел (в общем случае не равный N или M), а второй байт — размер элемента в битах. Выполняется векторное сложение по правилу $c_i = a_i + b_i$, где a_i, b_i берутся соответственно из последовательностей двоичных цифр массивов X и Y. Результат выводится в виде последовательности десятичных чисел, разделяемых запятыми. В общем случае $(X_0 \neq Y_0)\&(N \neq M)$.

24. Программа вводит матрицу смежности графа, звенья которого взвешены целыми числами, и выводит список его звеньев. Значения в клетках матрицы представляются десятичными, шестнадцатиричными и восьмиричными константами, запасанными по правилам языка Си. Эти значения разделяются запятыми. Одна строка матрицы располагается в отдельной строке входного текста. Предварительно распознается, не является ли граф неориентированным. Если граф неориентированный, то выводится список ребер, где отсутствуют взаимно-обратные пары. Например, для m[2,10] = 0x14 и m[10,2] = 20 имеем две дуги ориентированного графа <10,2,20> и <2,10,20>, которые при установлении факта симметричности матрицы смежности в выводимом списке должны быть представлены одной тройкой. Если граф ориентированный, то выводится список его дуг. Все числовые значения выводимых троек представляются в десятичном виде.

25. Программа вводит текст, каждая строка которого содержит выражение вида «$x \pm y$», где x, y — двоичные, десятичные или шестнадцатиричные константы, сформированные по правилам языка ассемблера; $'\pm'$ — знак операции сложения или вычитания. Формируется массив Z результатов вычисления введенных выражений. Затем вводится массив M десятичных чисел, разделенных запятой. Программа вычисляет процентную долю тех элемен-

тов массива Z, которые равны соответствующим элементам массива M. Результат выводится в десятичном виде. Все числа, фигурирующие во входных данных, и результаты вычисления выражений умещаются в 32-разрядные машинные слова.

26. Программа вводит определение матрицы M, содержащее три оператора следующего формата:

```
K1 = C1
K2 = C2
M   DQ  M1, M2, \ldots
```

где C1, C2, M1, M2 — шестнадцатиричные и десятичные целые знакопеременные константы. K1 задает число строк матрицы, а K2 — число столбцов. Вводятся в десятчном виде значения индексов I, J. Выводится десятичная сумма $M[I, J] + M[J, I]$.

27. Программа вводит определение бинарного отношения с помощью матрицы M, содержащей n n-разрядных двоичных слов, располагающихся по одному на каждой строке; $n \leq 80$. Во внутреннем представлении один байт должен хранить содержимое 8 подряд расположенных клеток матрицы отношения. Программа распознает следующие свойства отношения: симметричность, ассимметричность, антисимметричность, рефлексивность, антирефлексивность, нерефлексивность. Результат выводится в виде шести строк, каждая из которых содержит имя свойства из вышеуказанного списка и цифру 1 или 0, означающие соответственно обладание или необладание отношением указанного свойства.

28. Программа вводит определение отношения эквивалентности с помощью матрицы M, содержащей $N * N$-разрядных двоичных слов, располагающихся по одному на каждой строке; $N \leq 80$. Во внутреннем представлении один байт должен хранить содержимое 8-ми подряд расположенных клеток матрицы отношения. Затем вводится строка с десятичными номерами элементов множества, над которыми задано отношение, и распознается не входят ли эти элементы в один класс эквивалентности.

29. Программа вводит матрицу десятичных чисел M и вектор восьмиричных чисел V. Формируется и выводится в шестнадцатиричном виде произведение матрицы на вектор.

30. Программа вводит десятичное число, задающее число строк квадратной матрицы M, а затем последовательность восьмиричных чисел в формате «$\pm d, \ldots, d$» ($L_d \leq 30$), которые образуют содержимое клеток M. Формируется и выводится в десятичном виде сумма элементов главной диагонали M.

Лабораторная работа 4

Моделирование цифровых вычислительных устройств

Цель работы

Изучение методов представления структурно-функциональной организации цифровых вычислительных устройств в ассемблер-программах. Приобретение умения программировать алгоритмы поведения средств вычислительной техники на уровне межрегистровых передач и уровне обработке сигналов.

Порядок выполнения работы и содержание отчета

1. Анализ схемы и алгоритмов функционирования заданного цифрового устройства. *В отчет должны быть включены текст постановки задачи и описание цифрового устройства.*

2. Разработка способов представления объектов цифрового устройства в памяти моделирующей программы и на экране дисплея. *В отчет включается описание внутреннего и внешнего представления объектов моделируемого устройства.*

3. Разработка алгоритмов моделирования. *Алгоритм включается в отчет.*

4. Разработка программы на языке ассемблера. *Текст программы с комментариями включается в отчет.*

5. Разработка контрольных примеров. *В отчет включается описание и обоснование контрольных примеров.*

6. Отладка программ. *Описание процесса отладки и файла таблицы состояний включаются в отчет.*

7. Составление отчета.

Общие требования

1. Программа моделирования должна воспринимать значения сигналов управления и входных данных из текстового файла и обеспечивать потактовое выполнение микроопераций с визуализацией содержимого объектов цифрового устройства на экране дисплея, либо в выходном файле типа «.DAT», если такой вывод разрешен аргументами командной строки запуска программы либо в стартовом диалоге.

2. В ходе моделирования отображаются состояния следующих объектов: а) внутренние состояния регистров, счетчиков, триггеров; б) выходы функциональных узлов; в) шины. Отображение состояний должно строиться в виде таблицы состояний, столбцы которой соответствуют выводам микросхем, а строки — номерам тактов дискретного времени. Значения состояний выводов могут представляться в двоичном, восьмиричном и шестнадцатиричном коде.

Состояние высокого импеданса представляется символом 'Z'. Преподаватель может задать иные формы отображения.

3. Состояния накапливающих узлов, переключаемых различными фронтами тактового сигнала, отображается по полутактам (после переднего фронта, после заднего фронта).

Варианты заданий

1. БИС К1804ВС1
2. БИС К1804ВС2 (без специальных функций АЛУ)
3. БИС К1804ВС2 (специальные функции АЛУ)
4. БИС К1804ВУ4
5. БИС К1804ВУ1 в типовом включении с К1804ВУ3
6. БИС К1804ВУ5 (0 в разряде MI(0) кода микроинструкции)
7. БИС К1804ВУ5 (1 в разряде MI(0) кода микроинструкции)
8. БИС К1804ВР2
9. БИС К1804ИР2+К1804ВА3
10. БИС К1804ИР3+К1804ВА1
11. БИС К1802ВС1 (0 в разряде MI(0) кода микроинструкции)
12. БИС К1802ВС1 (1 в разряде MI(0) кода микроинструкции)
13. БИС К1802ИР1+К1802ВВ1
14. БИС К1802ВР1
15. БИС К1802ВР2
16. БИС К1802ВР3+К1802ВР4
17. БИС К1802ВР5+К1802ИМ1
18. БИС К1800ВС1
19. БИС К1800ВУ1
20. БИС К1800РП6+К1800ВР8
21. БИС К1800ВТ3+К1800ВА7
22. БИС К580ИК51
23. БИС К580ВВ55
24. БИС К580ВГ75
25. БИС К580ВД79
26. Порт последовательной связи МК MSC-196 [8, с.367]
27. Генератор периодических сигналов БИС 8xC196MC/MD/MH [8, с.336]
28. ШИМ-генератор БИС 8xC196MC/MD/MH [8, с.326]
29. Последовательный интерфейс МК КМ1816ВЕ51 [12, с.67]
30. Асинхронный приемопередатчик МК AT90S2313 [10, с.55]

Справочные данные об архитектуре I80x86

Программно-доступные компоненты и способы адресации

Основными программно-доступными компонентами I80x86, задаваемыми в аргументах командных строк ассемблер-программы, являются регистры общего назначения (РОН), адресные регистры (базовые и индексные), сегментные регистры, счетчик (указатель) команд, регистр флагов, ячейки памяти. Регистры адресуются зачастую неявно, что закрепляет за ними определенную специализацию. В машинных командах могут использоваться части РОН. Например, AX=AH_AL=EAX[15:0], где $r1\_r2$ - регистровая пара, в которой $r1$ – старшая часть, $r[m:n]$ – диапазон разрядов, в котором m – номер старшего разряда. Ниже приводится список регистров общего назначения и адресных регистров. Первыми в списке указаны имена 32-разрядных регистров, которые могут использоваться только в 32-разрядном режиме работы процессора либо в 16-разрядном, но в командах с префиксом изменения длины операнда SIZ. Этот префикс в большинстве случаев вставляется ассемблером автоматически на основе факта фигурирования 32-разрядного регистра среди операндов.

`EAX/AX/AH/AL` РОН, часто неявно адресуемые как аккумулятор. AX=AH_AL=EAX[15:0].

`ECX/CX/CH/CL` РОН, используемые также как неявно адресуемые счетчики числа сдвигов в сдвиговых командах и как счетчики длины цепочки данных в строковых командах. CX=CH_CL=ECX[15:0].

`EDX/DX/DH/DL` РОН, используемые также для представления неявно адресуемой старшей части операнда повышенной точности и адреса порта ввода/вывода. DX=DH_DL=EDX[15:0].

`EBX/BX/BH/BL` РОН, а также базовый регистр, используемый через коды методов адресации. BX=BH_BL=EBX[15:0].

`ESP/SP` Указатель стека. В большинстве случаев адресуется неявно в командах организации подпрограмм и прерываний. SP=ESP[15:0].

`EBP/BP` Базовый регистр, используемый обычно для адресации локальных данных в стеке. BP=EBP[15:0]. Имеет важную особенность – фигурирование этого регистра в адресном выражении заставляет процессор использовать по умолчанию сегментный регистр SP.

`ESI/SI` Индексный регистр/РОН. В строковых командах используется неявно как указатель источника данных. SI=ESI[15:0].

`EDI/DI` Индексный регистр/РОН. В строковых командах используется неявно как указатель приемника данных. DI=EDI[15:0].

Счетчик команд IP/EIP не фигурирует в аргументах машинных команд и его программная доступность ограничивается командами управления порядком выполнения программы, организации подпрограмм и прерываний.

Регистр флагов RF хранит признаки результата выполненной операции и флаги управления процессом. При программировании прикладных программ используются следующие основные флаги:

`CF` Признак переноса (заема) (Carry Flag);

`PF` Признак четности числа единиц (Parity Flag);

`AF` Признак переноса (заема) из младшего полубайта для десятичной арифметики (Auxiliary Flag);

`ZF` Признак нулевого результата (Zero Flag);

`SF` Признак отрицательного результата (копия знакового разряда результата) (Sign Flag);

`OF` Признак переполнения результата (Overflow Flag);

TF	Флаг разрешения трассировки (Trap Flag, при TF=1 после выполнения команды происходит прерывание INT 1);
IF	Флаг разрешения аппаратных прерываний (Interrupt-enable Flag);
DF	Флаг управления направлением автоиндексации в строковых командах (Direction Flag, при DF=0 – автоинкремент индексного регистра, при DF=1 – автодекремент).

Способы адресации

Способы адресации операндов задаются в операторных строках ассемблер программы на основе следующих основных правил:

- непосредственный операнд задается константой, идентификатором константы или константным выражением;
- регистровая адресация задается либо неявно, либо представляется своим именем, либо именем, значение которого сопоставлено с регистром через директивы прямого присваивания;
- память адресуется через адресное выражение, перед которым может фигурировать описатель типа указателя (byte ptr, word ptr и т.п.); описатель может быть опущен, если контекст машинной команды однозначно определяет тип указателя.

Для представления основных адресных выражений будем использовать обозначения: *const* – константа или константное выражение (выражение, значение которого известно до выполнения программы, очень часто это метка декларации данных), *base* – базовый регистр, *ind* – индексный регистр, *scale* – масштаб индекса из множества {1,2,4,8}; ind_s – конструкция, представляющая либо индексный регистр *ind*, либо произведение $ind * scale$; *abs* – метка декларации данных, представляющая абсолютный адрес операнда; символ «|» – разделитель различных вариантов выражений. Основные варианты адресных выражений имеют вид:

$abs \mid [abs] \mid abs + const \mid [abs + const] \mid const[abs] \mid [const]$ – выражения абсолютного адреса, где *abs* – имя метки декларации данных, *const* – константное выражение;

$[base] \mid [ind_s]$ – адрес операнда находится в регистре;

$[base + const] \mid [ind_s + const] \mid const[base] \mid const[ind_s]$ – адрес вычисляется как сумма содержимого возможно масштабированного регистра и констрантного смещения;

$[base + ind_s] \mid [base][ind_s]$ – адрес вычисляется как сумма содержимого двух регистров, один из которых может масштабироваться;

$[base + ind_s + const] \mid const[base][ind_s] \mid [base][ind_s + const]$ – адрес вычисляется как сумма констрантного смещения и содержимого двух регистров, один из которых может быть масштабирован.

В 16-разрядном режиме работы процессора $base \in$ {BX,BP}, $ind \in$ {SI,DI}, причем, недопустимо адресное выражение [BP] и масштабирование индексного регистра. В 32-разрядном режиме базовыми могут быть любые 32-разрядные регистры (EAX, ...,EDI), однако, недопустима косвенно-регистровая адресация [EBP] и [ESP], индексными могут быть все регистры, кроме ESP.

Адресное выражение задает механизм вычисления так называемого эффективного адреса EA. Этот адрес преобразуется в дальнейшем в линейный адрес, который, в свою очередь, преобразуется в физический адрес через дополнительный механизм страничного преобразования. Для выполнения лабораторных работ по машинно-ориентированному программированию достаточно знать правила формирования линейного адреса. Этот адрес формируется как сумма EA и базового адреса сегмента памяти. В реальном режиме базовый адрес формируется по выражению: 16 * SR, где SR – сегментный регистр. В

защищенном режиме базовый адрес сегмента находится в дескрипторе сегмента, также связанного с сегментным регистром. В обоих режимах сегментный регистр задается либо соглашениями по умолчанию, либо явно через конструкцию вида «SR :» перед адресным выражением, например, ES:[BX]. Такая конструкция порождает в машинной программе так называемый префикс замены. При обработке данных наиболее важными являются следующие соглашения по умолчанию:

а) если в адресном выражении фигурирует регистр BP, либо EBP, либо ESP, то используется сегментный регистр SS;

б) для строковых команд операнд-источник адресуется через DS:SI либо DS:ESI, а операнд-приемник — через ES:DI либо ES:EDI, причем, только DS может быть заменен на другой сегментный регистр через префикс замены;

в) в других случаях используется регистр DS.

Система команд

При описании системы команд приняты следующие обозначения:

а) v – константа или константное выражение (значение метки – самый частый случай константного выражеия);

б) $v8/v16/v32/v48/v64$ – константы или константные выражения, для которых определена соответствующая разрядность; аналогично определяются разрядности для операндов других типов;

в) s – операнд-источник, который может находиться в регистре, памяти или команде (непосредственный операнд);

г) d – операнд-приемник, который может находится в регистре или памяти;

д) r – регистровый операнд;

е) m – операнд в памяти;

ж) информация о признаках результата представляется в последнем предложении спецификации команды списком флагов, в котором равенства вида «CF=0» и «CF=1» означают фиксацию соответствующих констант во флаге, конструкция вида «CF?» означает неопределенность флага, конструкция вида «CF» означает фиксацию во флаге соответствующего признака результата операции, а отсутствие флага в списке означает, что операция не влияет на его значение.

з) конструкции типа eAX, eCX и т.п. представляют возможность использования как 16-разрядного, так и 32-разрядного регистра, т.е. eAX эквивалентно AX/EAX.

Организация передачи управления

`Безусловные переходы`

`JMP SHORT` $v8$	Переход, задаваемый знаковым смещением $v8$ относительно IP
`JMP NEAR` $v16/v32$	Переход, задаваемый знаковым смещением $v16/v32$ относительно IP/EIP.
`JMP FAR` $v32/v48$	Переход, задаваемый FAR-адресом, загружаемым из машинной команды в CS:eIP.
`JMP WORD PTR` d	Внутрисегментный переход по адресу, равному операнду d.
`JMP DWORD/FWORD PTR` d	Межсегментный переход по FAR-адресу, равному операнду d.

`Условные переходы`

Команды условного перехода задаются через мнемокод вида «Jcnd v», где cnd — суффикс, определяющий условие ветвления, v — константа или константное выражение размером в байт, слово или двойное слово (в 16-разрядном режиме только байт). Обычно v представляется меткой оператора, куда нужно переходить, но в машинную команду ассемблер транслирует расстояние «прыжка» до точки перехода. Суффиксы «cnd» команд

условного перехода по флагам используются также в командах условной передачи данных CMOV*cnd* и условной установки байта SET*cnd*.

`Условные переходы по флагам и значению CX`

`JE/JZ` v	Переход, если равно (ZF=1).
`JNE/JNZ` v	Переход, если не равно (ZF=0).
`JC` v	Переход, если перенос (CF=1).
`JNC` v	Переход, если нет переноса (CF=0).
`JS` v	Переход, если отрицательно (SF=1).
`JNS` v	Переход, если неотрицательно (SF=0).
`JO` v	Переход, если переполнение (OF=1).
`JNO` v	Переход, если нет переполнения (OF=0).
`JP/JPE` v	Переход по четности числа единиц результата (PF=1).
`JNP/JPO` v	Переход по нечетности числа единиц результата (PF=0).
`JCXZ/JECXZ` v	Переход, если CX/ECX=0.

`Переходы по результату операции над числами со знаком`

`JG/JNGE` v	Переход, если больше (SF=OF И ZF=0).
`JGE/JNL` v	Переход, если больше или равно (SF=OF ИЛИ ZF).
`JL/JNGE` v	Переход, если меньше (SF≠OF).
`JLE/JNG` v	Переход, если меньше или равно (SF≠OF ИЛИ ZF).

`Переходы по результату операции над беззнаковыми числами`

`JA/JNBE` v	Переход, если выше (CF=0 И ZF=0).
`JB/JNAE` v	Переход, если ниже (CF=1).
`JAE/JNB` v	Переход, если не ниже (CF=0).
`JBE/JNA` v	Переход, если не выше (CF ИЛИ ZF).

`Организация циклов со счетчиком`

`LOOP` $v8$	eCX:=eCX-1 и переход, если eCX≠O
`LOOPE/LOOPZ` $v8$	eCX:=eCX-1 и переход, если eCX≠O и ZF=1
`LOOPNE/LOOPNZ` $v8$	eCX:=eCX-1 и переход, если eCX≠O и ZF=0

`Операции с флагами`

`CLC`	Сброс флага переноса (CF=0).
`STC`	Установка флага переноса (CF=1).
`CMC`	Инверсия флага переноса CF.
`LAHF`	Загрузка младшего байта регистра флагов в регистр AH.
`SAHF`	Загрузка флагов SF, ZF, AF, PF, CF из бит 7, 6, 4, 2, 0 регистра AH.
`POPF`	Извлечение данных из стека в регистр флагов (EFLAGS[15:0|]).
`PUSHF`	Помещение в стек регистра флагов (EFLAGS[15:0|]).
`POPFD`	Извлечение данных из стека в расширенный регистр флагов EFLAGS.
`PUSHFD`	Помещение в стек расширенного регистра флагов EFLAGS.

`Организация процедур и прерываний`

`CALL` s	Вызов процедуры. Аналогично JMP тип операнда s может быть NEAR, FAR, WORD PTR, DWORD PTR, FWORD PTR.
`RET/RETF`	Возврат из процедуры (внутрисегментный/межсегментный).
`RET/RETF` $v16$	Возврат из процедуры с освобождением в стеке блока параметров размером $v16$ байт при 16-разрядной адресации, и размером $v16$ слов — при 32-разрядной.
`ENTER` $v16$, $v8$	Выделение блока параметров размером $v16$ байт в стеке для процедуры с логической вложенностью $v8$.

LEAVE	Освобождение блока параметров в стеке.
INT $v8$	Выполнение программного прерывания с номером $v8$.
INT 3	Однобайтовый код вызова прерывания 3 для перехода к отладчику.
INTO	Выполнение программного прерывания 4, если OF=1.
IRET/IRETD	Возврат из прерывания с восстановлением из стека IP/EIP, CS, FLAGS/EFLAGS.
CLI	Запрет маскируемых аппаратных прерываний (IF=0).
STI	Разрешение маскируемых аппаратных прерываний (IF=1).

Пересылка данных

Копирование и обмен данными

MOV d, s	Копирование s в d.
MOVSX d, s	Копирование байта/слова s в слово/двойное слово d со знаковым расширением.
MOVZX d, s	Копирование байта/слова s в слово/двойное слово d с нулевым расширением.
CMOVcnd d, s	Копирование s в d, если выполняется условие «cnd» .
XCHG d, s	Взаимообмен данными между регистрами или регистром и памятью.
CMPXCHG d, s	Если AL/eAX=d, mo $d = s$,ZF=1, иначе AL/eAX= d,ZF=0.
CMPXCHG8B $m64$	Условная перестановка учетверенного слова. Если EDX_EAX=$m64$, mo ($m64$=ECX_EBX, ZF=1), иначе EDX_EAX=$m64$.
BSWAP s	Перестановка байт из порядка младший-старший (L-H) в порядок старший-младший (H-L).
XLAT/XLATB	Загрузка AL байтом с адресом [eBX+AL].

Стековые команды пересылки

POP d	Извлечение слова/двойного слова из стека в регистр или память d с последующим автоувеличением SP/ESP на 2/4.
POPA/POPAD	Извлечение 8 слов/двойных слов из стека в регистры eDI, eSI, eBP, eSP(извлечение без модификации регистра), eBX, eDX, eCX, eAX с последующим автоувеличением SP/ESP на 16/32.
PUSH s	Помещение слова/двойного слова в стек после автоуменьшения SP/ESP на 2/4.
PUSHA/PUSHAD	Помещение в стек регистров eAX, eCX, eDX, eBX, eSP (исходное значение), eBP, eSI, eDI с автоуменьшением SP/ESP на 2/4 перед записью в стек содержимого каждого регистра.

Загрузка указателей и ввод-вывод

LEA r, m	Загрузка эффективного адреса операнда m в регистр r.
LDS/LES/LFS/LGS/LSS r, m	Загрузка регистра r и сегментного регистра DS/ES/FS/GS/SS far-адресом, равным значению операнда m.
IN AL/eAX, p	Ввод из порта p ввода/вывода в AL/eAX; p – либо константа $v8$, либо DX.
OUT p, AL/eAX	Вывод в порт из AL/eAX.

Команды арифметической обработки

Традиционные операции двоичной арифметики

ADD d,s	Сложение $d := d + s$. Все флаги.
ADC d,s	Сложение двух операндов с учетом переноса от предыдущей операции $d := d + s$+CF. Все флаги.
SUB d,s	Вычитание $d := d - s$. Все флаги.
SBB d,s	Вычитание с заемом $d = d - s$−CF. Все флаги.

`CMP` d,s	Сравнение $d - s$ без сохранения разности. Все флаги.
`INC` d	Инкремент $d := d + 1$. Все флаги, кроме CF.
`DEC` d	Декремент $d := d - 1$. Все флаги, кроме CF.
`NEG` d	Изменение знака операнда $d := 0 - d$. ZF. CF=1.
`MUL` s	Умножение беззнаковое AX:=AL$*s$/DX_AX:=AX$*s$/EDX_EAX:=EAX$*s$. OF=CF=0, если результат помещается в AL/AX/EAX, иначе OF=CF=1; SF?,ZF?,AF?,PF?.
`IMUL` s	Умножение знаковое (аналогично MUL).
`DIV` s	Деление беззнаковое (AL:=AX DIV s; AH:=AX MOD s)/ (AX:=DX_AX DIV s; DX:=DX_AX MOD s)/ (EAX:=EDX_EAX DIV s; EDX:=EDX_EAX MOD s); вызов INT 0, если результат не помещается в AL/AX/EAX. Все флаги неопределены.
`IDIV` d	Деление знаковое (аналогично DIV).
`XADD` d,s	Обмен содержимым и сложение ($< d, s >:=< s + d, d >$.
`CBW/CWDE`	Преобразование байта AL в слово AX (расширение знака AL в AH – AH заполняется битом 7 AL) или слова AX в двойное слово EAX.
`CWD/CDQ`	Преобразование слова AX в двойное слово DX:AX или двойного слова EAX в учетверенное EDX:EAX через расшинение знака.

`Команды десятичной арифметики`

`DAA/DAS`	Десятичная коррекция AL после сложения/вычитания двух упакованных чисел.
`AAA/AAS`	Десятичная коррекция после сложения/вычитания двух неупакованных чисел.
`AAD`	Десятичная коррекция перед делением неупакованного двузначного числа.
`AAM`	Десятичная коррекция после умножения двух неупакованных чисел.

Команды логических операций

`NOT` d	Инвертирование всех разрядов $d :=$ NOT d.
`AND` d,s	Логическое И: $d := d$ AND s. CF=OF=0,SF,ZF,AF?,PF.
`OR` d,s	Логическое ИЛИ: $d := d$ OR s. CF=OF=0,SF,ZF,AF?,PF.
`XOR` d,s	Исключающее ИЛИ: $d := d$ XOR s. CF=OF=0,SF,ZF,AF?,PF.
`TEST` d,s	Проверка бит (логическое И d AND s без записи результата). CF=OF=0,SF,ZF,AF?,PF.

Команды сдвигов

`SHL` $d,\ s$	Логический сдвиг влево d на s разрядов ($s \in \{1, CL, v8\}$). Справа вдвигается 0, а старший бит выдвигается в CF. CF,SF,ZF,AF?,PF, OF при $s = 1$, OF? при $s > 1$).
`SHR` $d,\ s$	Логический сдвиг вправо. Аналогично SHL, но 0 вдвигается слева, а в CF выдвигается младший бит.
`SAL` $d,\ s$	Арифметический сдвиг влево. Совпадает с SHL.
`SAR` $d,\ s$	Арифметический сдвиг вправо. Отличается от SHR тем, что знаковый разряд d сохраняет свое значение.
`ROL/ROR` $d,\ s$	Циклический сдвиг влево/вправо. CF, OF при $s = 1$, OF? при $s > 1$).
`RCL/RCR` $d,\ s$	Циклический сдвиг влево/вправо с включением в кольцо сдвига флага CF. CF, OF при $s = 1$, OF? при $s > 1$).
`SHLD/SHRD` $d,\ r,\ s$	Логический сдвиг влево/вправо операнда d, таким образом, будто r образует с ним пару d_r/r_d, однако, разряды r выдвигаются в d, но в конечном итоге r сохраняет исходное значение. OF?, SF, ZF, AF?, PF, CF.

Команды обработки бит и байт

BSF/BSR r, s	Сканирование бит вперед(от младшего)/назад(от старшего) до встречи бита со значением 1 и запись номера этого бита в r. ZF.
BT d, s	Тестирование бита – CF:=$d[s]$.
BTR/BTS d, s	Тестирование и сброс/установка бита – (CF:=$d[s]$, $d[s]$=0/1).
BTC d, s	Тестирование и инвертирование бита – (CF:=$d[s]$, $d[s]$:= NOT $d[s]$).
SETcnd $d8$	Если выполняется условие cnd, то $d8$:=01h, иначе $d8$:=00h.

Префиксы команд

SIZ	Изменение установленной по умолчанию разрядности данных 16/32 на 32/16 для следующей команды.
ADDRSIZ	Изменение умалчиваемой разрядности адреса 32/16 на 16/32 для следующей команды.
CS:/SS:/DS:/ES:/FS:/GS:	Явное назначение сегментного регистра для адресации памяти в следующей команде.
LOCK	Захват локальной шины на время выполнения инструкции.
REP/REPE/REPZ/REPNE/REPNZ	Префикс повтора строковых операций (см. ниже).

Команды поддержки обработки массивов

Основу поддержки обработки массивов в I80x86 образуют так называемые строковые команды. Эти команды используют неявную адресацию через DS:eSI и ES:eDI с постмодификацией индексного регистра либо в сторону увеличения (DF=0), либо в сторону уменьшения (DF=1) на размер адресуемого данного, т.е. на 1, 2 или 4. Для far-адреса DS:eSI допускается замена сегментного регистра через префикс.

MOVSB/MOVSW/MOVSD	Копирование байта/слова/двойного слова из памяти по адресу DS:eSI в память по адресу ES:eDI; модификация eSI, eDI.
LODSB/LODSW/LODSD	Копирование байта/слова/двойного слова из памяти с адресом DS:eSI в AL/AX/EAX; модификация eSI.
STOSB/STOSW/STOSD	Запись байта/слова/двойного слова из AL/AX/EAX в память по адресу ES:eDI; модификация eDI.
CMPSB/CMPSW/CMPSD	Сравнение массивов байт/слов/двойных слов через фиксацию в регистре флагов признаков результата операции вычитания ((far *) DS:eSI - (far *) ES:eSI); модификация eSI, eDI. Все флаги.
SCASB/SCASW/SCASD	Сравнение с фиксацией в регистре флагов признаков результата операции вычитания AL/AX/EAX из байта/слова/двойного слова по адресу ES:eDI; модификация eDI. Все флаги.
INSB/INSW/INSD	Запись байта/слова/двойного слова, введенного из порта с адресом DX, в память по адресу ES:eDI; модификация eDI.
OUTSB/OUTSW/OUTSD	Вывод байта/слова/двойного слова из памяти с адресом DS:eSI, в порт с адресом DX; модификация eSI.
REP	Префикс повтора строковых операций до обнуления счетчика eCX, счетчик декрементируется на каждом повторе.
REPE/REPZ	Повторение, пока ZF=1 и CX $>$ 0.
REPNE/REPNZ	Повторение, пока ZF=0 и CX $>$ 0.
CLD/STD	Сброс/установка флага направления. При DF=0 – автоинкремент eSI, eDI, при DF=1 – автодекремент.
BOUND r, m	Проверка нахождения содержимого регистра r в границах, заданных двумя подряд расположенными ячейками памяти m и генерация прерывания INT 5 при нарушении границ.

Библиографический список

1. Юров В. И. Assembler. Учебник. – СПб.: Питер, 2002.
2. Юров В. И. Assembler. Практикум. – СПб.:
3. Юров В. И. Assembler. Специальный справочник. – СПб.: Питер, 2000.
4. Зубков С.В. Assembler. Для DOS, Windows и UNIX. – М.: ДМК, 1999.
5. Пустоваров В.И. Ассемблер: программирование и анализ корректности машинных программ. – Киев: BHV, 2000.
6. Рудаков П.И., Финогенов К.Г. Язык ассемблера: уроки программирования. – М.: Диалог-МИФИ, 2001.
7. Пирогов В.Ю. Ассемблер для Windows. – М.: Издатель Молгачева С.В., 2002.
8. Козаченко В.Ф. Микроконтроллеры: руководство по применению 16-разрядных микроконтроллеров Intel MCS-196/296 во встроенных системах управления. — М.: Изд-во ЭКОМ, 1997.
9. Микропроцессоры и микропроцессоные комплекты интегральных схем: Справочник. В 2 т./В.Б.Абрайтис, Н.Н.Аверьянов, А.И.Белоус и др.; Под ред. В.А.Шахнова. — М.: Радио и связь, 1988.
10. Голубцов М.С. Микроконтроллеры AVR: от простого к сложному. — М.: СОЛОН-ПРЕСС, 2003.
11. Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник/Под общ. ред. С.Т.Хвоща. — Л.: Машиностроение, 1987.
12. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. — М.: Энергоатомиздат, 1990.

Учебное издание

Машинно-ориентированное программирование

Методические указания

Составитель НЕГОДА Виктор Николаевич

Редактор: Н.А.Евдокимова

Подписано в печать 14.01.2003. Формат 60×84 1/16. Бумага оберточная.

Печать офсетная. Усл. печ.л. 1,86. Уч.-изд.л. 1,50. Тираж 100 экз.

Заказ № .

Ульяновский государственный технический университет,

432027, г.Ульяновск, ул.Северный Венец, 32.

Типография УлГТУ, 432027, г.Ульяновск, ул. Сев.Венец 32.