Министерство образования и науки
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Южный федеральный университет»
Факультет математики, механики и компьютерных наук

Практические занятия по курсу «Микропрограммирование»

Ростов-на-Дону
2010

Содержание

Программное обеспечение

Виртуальная машина под управлением OC FreeDOS 1.0 с установленными: последней версией пакета ssh2dos (0.2.1; в дистрибутиве FreeDOS старая, неработающая в некоторых важных для нас случаях версия), программами wasm и nasm (есть в полном дистрибутиве FreeDOS). Первый нужен для использования утилиты sftptodos, являющейся sftp-клиентом со стандартным ftp-интерфейсом UNIX.

Для подключения к sftp-серверу из Windows можно использовать WinSCP или утилиту PSFTP, относящуюся к пакету программ PuTTY. В большинстве современных дистрибутивов GNU/Linux утилита sftp присутствует по умолчанию.

В ситуации ограниченного доступа в интернет возможно использовать **локальный sftp сервер** (на домашнем компьютере). Для установки локального sftp-сервера в Linux-системах достаточно установить пакет openssh-server. Для Windows-систем существует некоторое число руководств в сети.

Для программирования на языке ассемблера под Windows используется пакет Masm32.

Необходимые знания и навыки

Использование DOS-shell (cd, md, copy, del, etc.), использование редактора edit.exe, использование консольного клиента ftp (см. раздел «Команды» справки man). Знакомство с архитектурой компьютера (желательно, в объёме семестрового курса) и устройством процессоров семейства x86. Знание английского языка с возможностью беглого чтения технических текстов.

Задание 1. Архитектура x86 и язык ассемблера

Тема: низкоуровневые языки программирования, общие сведения об архитектуре x86, регистры процессора, адресация в реальном режиме, пример программы «Hello, World» на языке ассемблера.

Литература*: [1: гл. 2, п. 1; гл. 3, п. 1–3; гл. 4, п. 1, 2, 3], [9: ch. 2, par. 1-4; ch. 3, par. 1, 2, 4; ch. 4, par. 1–3; 4.4.1]

Замечание: в [1] используется MASM-совместимый синтаксис, в [9] — синтаксис NASM.

1. Создать программу «Hello world» в синтаксисе NASM (компиляция командой nasm) — с помощью документации NASM (ch. 8). EXE файл из объектного файла получается точно также, как рассматривалось в классе, с помощью линковщика val.
2. Создать программу «Hello world», аналогичную представленной в классе, но имеющую формат COM-файла: содержимое сегмента кода меняется мало, а инструкции объявления сегментов программы просто исчезают. В синтаксисе MASM и NASM (используя, соответственно, главу 4 у Зубкова и документацию NASM или книги [9, 10]).
3. Эссе на тему: «Выбор диалекта ассемблера для курса микропрограммирования» по статье «Which Assembler is the Best?», имеющейся литературе, опыту написания программ (дискуссиям в интернете, Википедии и т. д.). Вы должны аргументированно показать, что один ассемблер лучше другого по набору параметров, которые вы считаете

* Список литературы приведён в конце пособия

важными. Справку по другим ассемблерам при необходимости можно получить в Википедии. Объём: ~ лист А4.

Требования к выполнению заданий:

- Все программы должны иметь исходный текст и скомпилированный код (объектные и EXE/COM файлы).
- Исходный текст должен быть тщательно откомментирован (однострочные комментарии начинаются символом «;» и продолжаются до конца строки).

Задание 2. Базовые конструкции языка ассемблера

Темы: директивы объявления данных, арифметические и логические операции (работа с беззнаковыми целыми), (без)условные переходы в программе, моделирование циклов.

Литература: [1: 2.3.1—7], [10: гл. 3, 4].

Пример программы

```
;       numtostr.asm
;       print a number in hex

.model small

.stack 100h

.data
        num             dw      20      ; number to print
        msk             dw      0fh     ; mask to get first 4 bits
        zero_char       dw      '0'     ; constants
        a_char          dw      'A'     ;       to make ASCII hex-figures
        result          db      4 dup (0),'h$'  ; result string

.code
start:
        mov     ax, _DATA       ; loading address
        mov     ds, ax          ;       of the data segment

        mov     bx, offset result       ; loading addr of result string
                                        ; to manipulate C-pointer-style
        mov     cl, 0           ; shift to get next 4-bit group
                                ;       to the bits #1-3 (begining of byte)
cont_print:
        mov     ax, num
        shr     ax, cl
        and     ax, msk
        cmp     ax, 10
        jge     letter          ; figure 'A'..'F'
        add     ax, zero_char   ; figure '0'..'9'
        jmp     print
letter:
```

```
        sub     ax, 10
        add     ax, a_char
print:
        mov     [bx], al        ; print current figure into result string
        inc     bx              ; move to place for the next figure
        add     cl, 4
        cmp     cl, 16
        jl      cont_print
; end of cycle

        mov     dx, offset result
        mov     ah, 09
        int     21h

;exit:
        mov     ah, 4ch
        int     21h
END start
```

Все исходные числовые значения, о которых упоминается в заданиях, должны быть заданы в сегменте данных (если не указано иное).

Вариант 1

1. Собрать (т. е. получить исполняемый файл) программу, приведённую выше. Изменить программу так, чтобы число выводилось с обычным порядком следования разрядов (младшие разряды в конце).
2. Заполнить непрерывный блок памяти первыми 10 членами последовательности Фибоначчи.
3. Вычислить полином $p(x) = 2x^3 + 5x^2 + 8x + 7$ в точке $x = 7$. Используйте операцию битового сдвига вместо умножения там, где возможно. Добавьте код вывода результата на консоль (используйте код, рассмотренный в классе).
4. Вычислите сумму двух чисел, расположенных каждое в непрерывном блоке памяти из 5 байт. Результат должен быть записан в третий блок памяти из 6 байт. Учитывайте переносы из старших разрядов.
5. Напишите программу, которая переводит данную строку, содержащую запись числа в шестнадцатеричной системе счисления, в строку, содержащую запись того же числа в двоичной системе счисления. Распечатайте результат.
6. Все числовые значения, встречающиеся в текстах всех программ, и носящие семантику константы (неизменяемого значения), объявить с помощью директивы equ или = (см. книгу Кипа Ирвина, п. 3.5.1 и 3.5.3).

Вариант 2

1. Собрать (т. е. получить исполняемый файл) программу, приведённую выше. Изменить программу так, чтобы число выводилось с обычным порядком следования разрядов (младшие разряды в конце).
2. Заполнить непрерывный блок памяти убывающими положительными чётными числами, начиная с числа 42.

3. Вычислить полином $p(x) = 4x^4 + 8x + 6$ в точке $x = 5$. Используйте операцию битового сдвига вместо умножения там, где возможно. Добавьте код вывода результата на консоль (используйте код, рассмотренный в классе).
4. Вычислите разность двух чисел, расположенных каждое в непрерывном блоке памяти из 5 байт с условием, что уменьшаемое больше вычитаемого. Результат должен быть записан в третий блок памяти из 5 байт. Учитывайте возникающий недостаток.
5. Напишите программу, которая переводит данную строку, содержащую запись числа в восьмеричной системе счисления, в строку, содержащую запись того же числа в шестнадцатеричной системе счисления. Распечатайте результат.
6. Числовые значения, встречающиеся в текстах всех программ, и носящие семантику константы (неизменяемого значения), объявить с помощью директивы equ или = (см. книгу Кипа Ирвина, п. 3.5.1 и 3.5.3).

Задание 3. Базовые конструкции языка ассемблера. Системные прерывания ввода-вывода

Темы: циклы (окончание): loop*; стек (push/pop) и его использование для сохранения состояния; прерывание таймера int 15h; работа с дисплеем средствами BIOS (int 10h): видеорежимы, атрибуты символов, видеостраницы; прямая запись в видеопамять; ввод средствами DOS (int 16h), подготовка буфера для ввода.

Литература: [1: 2.3.{1,7}, 4.{3,4}, 4.7.1], [11: гл. 1, гл. 4]

1. Используя стек, проверить корректность скобочной структуры (сбалансированность скобок) в заданной строке.
2. Используя loop*, нарисовать прямоугольник с диагоналями из символов ‘*’ с заданными длинами сторон и координатами левой верхней вершины.
3. Вывести в заданную (константами) точку экрана «цветовое двоичное представление» заданного 16-разрядного числа: каждый бит, равный 0, печатается пробелом с одним цветом фона, равный 1 — другим. Цвета заданы константами. Выполнить в двух вариантах: с использованием прерываний BIOS и с прямой записью в видеопамять.
4. Верно ли, что введённая с клавиатуры строка состоит из одних латинских букв?
5. Бегущая строка: считать строку с клавиатуры, организовать её перемещение по экрану слева-направо, затем сверху-вниз. Без использования видеостраниц и с их использованием.

Задание 4. Процедуры в языке ассемблера

Темы: синтаксис вызова и определения процедур в языке ассемблера, передача параметров в регистрах и в стеке, передача параметров по значению и по адресу, методы возврата значений из процедур, рекурсивные процедуры, реализация в языках высокого уровня.

Литература: [5: гл. 10, «Процедуры»; гл. 15], [1: 5.2], [4: гл. 3, «Реализация рекурсивных процедур»].

Оформить отдельными процедурами вывод шестнадцатиразрядного числа на экран в шестнадцатеричной и десятичной форме, число передаётся в стеке. Реализовать процедуру, вводя-

щую целое шестнадцатиразрядное число с клавиатуры, результат возвращается через стек (возврат значения через стек реализовывать с помощью фиктивного параметра, см. книжку Юрова [5], конец 15-ой главы).

Вариант 1

1. Реализовать процедуру очистки экрана.
2. Реализовать процедуру, которая выводит содержимое заданного участка памяти на экран в шестнадцатеричном формате (в каждой строке по несколько значений). Адрес участка передаётся в регистре BX, длина участка (в байтах) — в CX.
3. Реализовать процедуру GCD, вычисляющую НОД двух чисел, используя алгоритм Евклида. Числа и результат передаются через стек.
4. Реализовать процедуру, вычисляющую сумму двух дробей. В стеке передаётся три адреса: двух операндов и результата, адрес указывает на область в памяти где подряд лежат два шестнадцатиразрядных числа, числитель и знаменатель. Числитель и знаменатель операндов и результатов передаются как отдельные числа через стек. Результат должен быть несократимой дробью.
5. Используя рекурсию, вычислить факториал числа.

Вариант 2

1. Реализовать процедуру вывода вывода на консоль «символа новой строки» (т. е. пары символов с кодами 0Dh, 0Ah).
2. Реализовать процедуру, которая выводит содержимое регистров на экран (в каждой строке по несколько значений). Флаги должны выводиться индивидуально (например, «CF=0»).
3. Реализовать процедуру, вычисляющую частное и остаток от деления двух чисел, используя только лишь сложение и вычитание.
4. Реализовать процедуру, вычисляющую произведение двух компле́ксных чисел. В стеке передаётся три адреса: двух операндов и результата, адрес указывает на область в памяти где подряд лежат два шестнадцатиразрядных числа, вещественная и мнимая часть (целые числа).
5. Используя рекурсию, вычислить заданную степень числа.

Задание 5. Раздельная трансляция. Составные типы данных

Темы. Ключевые слова public, extrn для использования подпрограмм и данных, определённых в разных модулях. Процедура линковки, объединение сегментов данных, кода и стека разных модулей. Массивы в языке ассемблера, адресация, масштабирование индекса. Структуры в языке ассемблера: определение типа, определение переменной, использование переменной. Понятие об объединениях и записях.

Литература. Раздельная трансляция: [5: гл. 15]. Адресация: [1: 2.2]. Составные структуры данных: [5: гл. 13], [1: 3.3.2].

Указания. Решения всех заданий необходимо оформлять процедурами в отдельных .asm-файлах и демонстрировать их работу, вызывая эти процедуры из других .asm-файлов. В частности, вывод результата процедуры на консоль чаще всего осуществляется в вызывающей программе. Для вывода следует использовать процедуры, созданные при выполнении предыдущих домашних заданий: оформите их отдельным модулем. Параметры процедур —

например, число элементов массива, адрес массива в памяти, адрес структуры в памяти — передаются через стек.

При использовании структуры в разных модулях объявление её типа должно быть вынесено в отдельный файл с расширением `.inc`. (например, `student.inc`, который содержит только объявление структуры `student` и больше ничего). Этот файл должен подключаться во все программы, использующие данную структуру, с помощью директивы ассемблера `include <имя_файла>` (без угловых скобок, например: `include student.inc`). Разумеется, один .inc-файл может содержать объявление нескольких (желательно, связанных между собой логически) структур.

Все массивы в заданиях, если не оговорено иное, считаются целочисленными, причём элементы являются *16-разрядными* числами. Не забывайте комментировать исходный код.

Вариант 1

1. Дан массив размера N и целые числа K и L ($1 \le K \le L \le N$). Найти сумму элементов массива с номерами от K до L включительно.
2. Дан целочисленный массив размера N, не содержащий одинаковых чисел. Проверить, образуют ли его элементы арифметическую прогрессию. Если образуют, то вернуть разность прогрессии, если нет — 0.
3. Дан двумерный массив размером $M \times N$. Вывести его элементы в следующем порядке: первая строка слева направо, вторая строка справа налево, третья строка слева направо, четвертая строка справа налево и т. д.
4. Дан двумерный массив размером $M \times N$. Найти максимальный среди минимальных элементов ее строк.
5. Определить тип `Date` — структуру с полями `day` (день), `month` (месяц) и `year` (год). Определить процедуру `PrevDate(d)` с параметром типа `Date`, которая преобразует дату `d` к предыдущей дате. Применить процедуру `PrevDate` к различным датам для демонстрации разных случаев изменения даты (меняется только день, или день и месяц, или день месяц и год).
6. Определить тип `Student` — структуру с полями `name` (фамилия, максимум 20 символов), `group` (номер группы от 1 до 11), `marks` (5 оценок за последнюю сессию, если экзаменов было меньше пяти лишние байты заполняются нулями). Определить процедуру, которая в массиве студентов находит количество отличников в заданной группе.

Вариант 2

1. Дан массив размера N и целые числа K и L ($1 \le K \le L \le N$). Найти сумму всех элементов массива, кроме элементов с номерами от K до L включительно.
2. Дан целочисленный массив размера N. Проверить, чередуются ли в нем четные и нечетные числа. Если чередуются, то вывести 0, если нет, то вывести порядковый номер первого элемента, нарушающего закономерность.
3. Дан двумерный массив размером $M \times N$. Вывести его элементы в следующем порядке: первый столбец сверху вниз, второй столбец снизу вверх, третий столбец сверху вниз, четвертый столбец снизу вверх и т. д.
4. Дан двумерный массив размером $M \times N$. Найти минимальный среди максимальных элементов ее столбцов.

5. Определить тип `Date` — структуру с полями `day` (день), `month` (месяц) и `year` (год). Определить процедуру `NextDate(d)` с параметром типа `Date`, которая преобразует дату `d` к следующей дате. Применить процедуру `PrevDate` к различным датам для демонстрации разных случаев изменения даты (меняется только день, или день и месяц, или день месяц и год).
6. Определить тип `Student` — структуру с полями `name` (фамилия, максимум 20 символов), `group` (номер группы от 1 до 11), `marks` (5 оценок за последнюю сессию, если экзаменов было меньше пяти лишние байты заполняются нулями). Определить процедуру, которая в массиве студентов находит количество студентов заданной группы, чья фамилия начинается на заданную букву.

Задание 6. Динамически выделяемая память

Темы: назначение динамически выделяемой памяти, интерфейсы использования динамически выделяемой памяти в Dos. Стандартный интерфейс (прерывание 21h), опция использования Upper memory area (UMA), файл (fd)config.sys. Системы использования памяти за пределом первого мегабайта: Expanded memory system (EMS), eXtended memory system (XMS), их драйверы: emm386.exe, himem.sys. Область памяти выше первого мегабайта размером в 64 KiB - 16 b — High Memory Area (HMA). Порции выделения памяти и их влияние на практику программирования.

Литература. Описание различных областей памяти и способов доступа к ним: [Wikipedia: Conventional memory, Extended memory, UMA, HMA, EMS, XMS]. Описание интерфейсов: [1: 4.9].

Реализовать работу с динамическим массивом целых чисел с помощью стандартных средств Dos (int 21h). В центре реализации должна быть процедура, записывающая в i-ую позицию массива полученное целое число. Конкретное представление самого массива определить самостоятельно (на основе интерфейса работы с динамической памяти Dos). Продемонстрировать работу программы на примере, где пользователь должен вводить целые числа в цикле, а программа — записывать их в массив. В начале программы нужно дать возможность задать стратегию распределения памяти Dos (с учётом UMA, прерывание int 21h, AH=58h). Условие окончание цикла ввода — ввод пустой строки. В конце программа должна распечатать введённый массив.

Вариант 1:

- реализовать основные процедуры работы с линейным двусвязным списком целых чисел (вставка в произвольное место, удаление произвольного элемента, печать списка), используя интерфейс EMS. В основной программе продемонстрировать работоспособность созданных процедур.

Вариант 2:

- реализовать основные операции работы с линейным двусвязным списком целых чисел (вставка в произвольное место, удаление произвольного элемента, печать списка), используя интерфейс XMS. В основной программе продемонстрировать работоспособность созданных процедур.

Замечание: для ввода и вывода использовать процедуры, созданные ранее.

Задание 7. Цепочечные (string) инструкции

Краткое описание цепочечных инструкций

1. Имена команд: `movs*, cmps*, lods*, stos*, scas*` — выполняют один шаг цикла с соответствующей семантикой (`mov` копирует, `cmp` сравнивает, scas выполняет поиск и т.д.), сдвигает указатель.

   * — возможное окончание указывает на размер элемента цепочки (b, w, d). Если отсутствует, размер определяется по аргументам:
   `movs str1, str2 ; str1, str2 только для указания размера`

   В версиях с окончанием аргументов нет.
2. Итерации задаются префиксом rep*

   rep movsb
3. Длина цепочки задаётся в регистре CX.
4. Расположение цепочек: ds:si, es:di.
5. Направление обхода цепочек: флаг DF (cld, std). DF=0 — в направлении увеличения адресов, DF=1 — уменьшения.

Пример программы

```
; prints first char which is different in str1 from str2
; usinf string instructions

.model small

.stack 100h

.data
str1    db      'String 1 - !!1'
str2    db      'String 1 - !!!'
len     dw      $-str2
crlf    db      10,13,'$'

.code
start:
        mov     ax, _data
        mov     ds, ax
        mov     es, ax   ; prepare segment register for string instruction

        lea     si, str1 ; load offset of 'source-string'
        lea     di, str2 ; load offset of 'destination-string'
        cld              ; clear DF - direction flag - scan forward
                                ; (not backward)
        mov     cx, len  ; strings' length
        repe cmpsb

        je    exit ; if CF isn't set, strings are equal - go to exit

print:
                         ; print different char
        dec     di       ; the different char is in di-1 position
        mov     dl, es:di
        mov     ah, 2
        int     21h

                         ; print crlf
```

```
        mov     dx, offset crlf
        mov     ax, 9
        int     21h

exit:
        mov     ah, 4ch
        int     21h
END start
```

Литература: [5: глава 12], [1: 2.3.8].

Указания. Все задания следует выполнять в виде процедур, которые принимают значения (символы, числа, адреса цепочек) через стек и так же, если необходимо, возвращают результат. Процедуры должны иметь осмысленные имена. Параметрам процедур, хранимым на стеке, следует присваивать псевдонимы (например, `s1 equ [BP+4]`). Для каждой процедуры нужно писать «основную программу», которая бы демонстрировала работу процедуры.

«Целое число» означает «целое 16-разрядное число», если не оговорено иное. Следует искать возможности использовать цепочечные команды для любых задач и подзадач.

Вариант 1

1. Дана цепочка символов и её длина, подсчитайте количество символов пунктуации (‘,’, ‘.’, ‘?’, ‘!’).
2. Даны цепочка-источник, содержащая заданное количество целых чисел, цепочка-приёмник однобайтовых элементов не меньшей длины и некоторое целое число. Записать в цепочку-приёмник позиции заданного числа в цепочке-источнике. После последней записанной позиции вставить 0.
3. Дана цепочка-источник чисел и её длина, а также цепочка-приёмник такой же длины. Скопировать в цепочку-приёмник числа из источника, разделив все чётные числа пополам.
4. **Индексная сортировка массива цепочек**. Задан адрес блока памяти, содержащего цепочки положительных чисел, разделённые числом 0, а также количество этих цепочек. Кроме того, задана цепочка s из двухбайтовых элементов, длина которой совпадает с количеством цепочек в блоке. В результате работы процедуры цепочка s должна содержать адреса цепочек исходного блока в таком порядке, относительно которого ссылаемые по этим адресам цепочки кажутся отсортированными. Адрес процедуры-предиката сравнения цепочек передаётся в качестве параметра процедуры. (Для сортировки последовательности элементов произвольного типа необходим лишь предикат сравнения двух элементов этого типа). Продемонстрировать работу процедуры в случае сортировки по возрастанию и по убыванию суммы элементов в цепочках.

Вариант 2

1. Дана цепочка чисел и её длина, подсчитайте количество чисел, кратных двум, трём или пяти.
2. Даны цепочка-источник, содержащая заданное количество целых 32-разрядных чисел, цепочка-приёмник однобайтовых элементов не меньшей длины и некоторое целое 32-разрядное число. Записать в цепочку-приёмник позиции заданного числа в цепочке-источнике. После последней записанной позиции вставить 0.
3. Дана цепочка-источник символов и её длина, а также цепочка-приёмник такой же длины. Скопировать в цепочку-приёмник символы из источника, переведя все строчные («маленькие») латинские буквы в прописные.

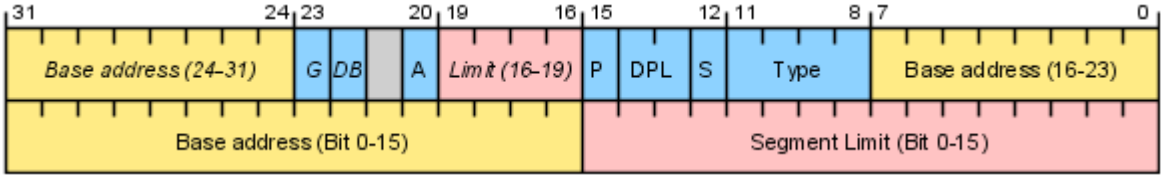
4. **Индексная сортировка массива цепочек**. Задан адрес блока памяти, содержащего цепочки символов, разделённые знаком '$', а также количество этих цепочек. Кроме того, задана цепочка s из двухбайтовых элементов, длина которой совпадает с количеством цепочек в блоке. В результате работы процедуры цепочка s должна содержать адреса цепочек исходного блока в таком порядке, относительно которого ссылаемые по этим адресам цепочки кажутся отсортированными. Адрес процедуры-предиката сравнения цепочек передаётся в качестве параметра процедуры. (Для сортировки последовательности элементов произвольного типа необходим лишь предикат сравнения двух элементов этого типа). Продемонстрировать работу процедуры в случае сортировки в лексикографическом порядке («как в словарях») по возрастанию и по убыванию.

Задание 8. Программирование перехода в защищённый режим

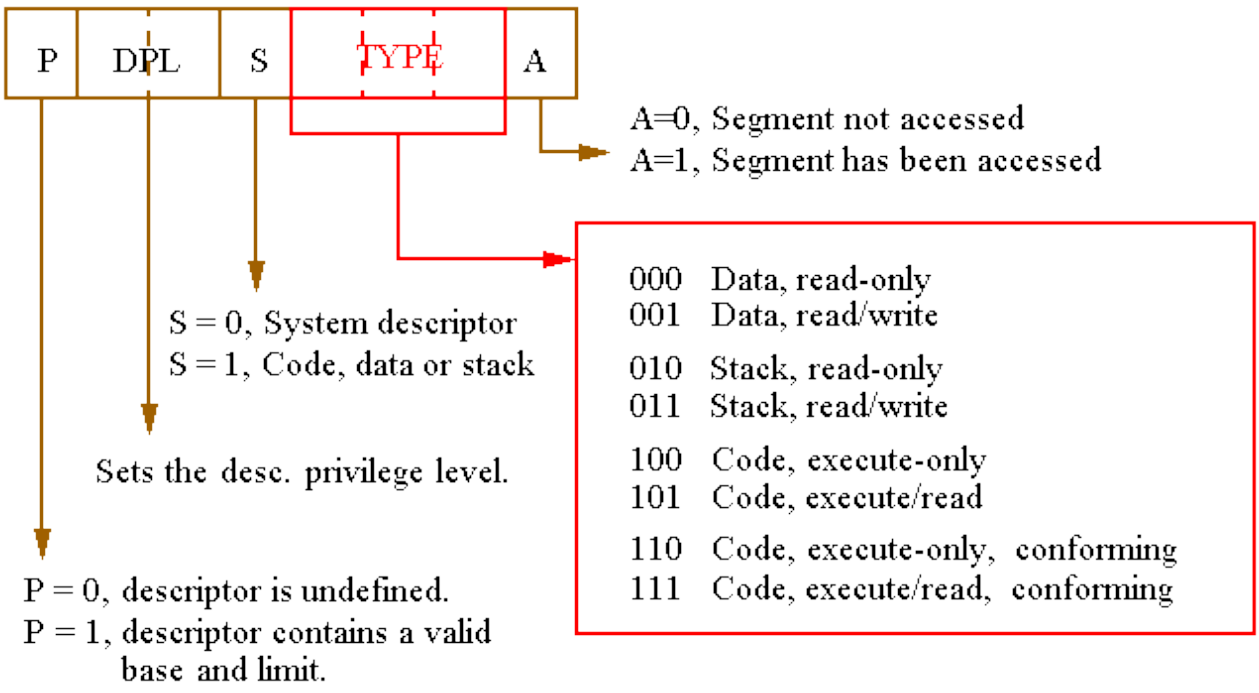
Темы. Основные цели введения защищённого режима: безопасная многозадачность, более гибкая адресация (поддержка бóльших объёмов памяти, страничная модель, поддержка виртуальной памяти). Системные регистры процессора для работы в защищенном режиме:

- управления (cr0-cr3);
- системных адресов (gdtr, ldtr, idtr, tr);
- отладки (dr0-dr7).

Дескриптор сегмента и его структура:



Байт прав доступа дескриптора сегмента AR, его структура:



Значения сегментных регистров в защищённом режиме (2 бита RPL — Requested Privilege Level, 1 бит для выбора между LDT и GDT, 13 бит для селектора в таблице дескрипторов сегментов); обзор действий, необходимых для перехода в защищённый режим из реального:

- формирование GDT,
- заполнение регистра gdtr (линейный адрес GDT),
- отключение прерываний,
- переход в защищённый режим (флаг в регистре cr0),
- установка значений сегментных регистров (в первую очередь, CS — при помощи инструкции jmp).

Теневые регистры и возвращение в реальный режим:

- заполнение теневых регистров (путём перезаписи значений в сегментные регистры, но с предварительно изменённой информацией в дескрипторах сегментов),
- переключение в реальный режим (флаг в cr0),
- заполнение сегментных регистров в соответствии с требованиями реального режима,
- разрешение прерываний.

Литература: [2: урок 16], Wikipedia: Protected Mode.

Задание. Создать программу, реализующую схему перехода из реального режима в защищённый и обратно. Добавить к ней ввод строки текста с клавиатуры в реальном режиме и вывод этой строки в защищённом режиме при помощи записи в видеопамять (пример такой записи имеется в [2]).

Задание 9. Программирование под Windows. Создание графических интерфейсов на Win32 API

Темы. Роль операционной системы в прикладном программировании. Основные библиотеки Win32 API (kernel32.dll, user32.dll, shell32.dll) и их статические (lib) версии в пакетах программирования.

Соглашения о вызове подпрограмм (calling conventions)

- передача параметров (регистры, стек, общая память),
- порядок размещения параметров на стеке,
- используемые вызываемой подпрограммой регистры,
- ответственный за раскрутку стека (вызывающий/вызываемый);

соглашения stdcall (стек — справа-налево — EAX, ECX, EDX (EAX может использоваться для возвращения результата) — вызываемый). Манглирование (декорирование) имён.

Пример программы

```
; winurl.asm
;
;     building:
;           ml /c /coff /Cp winurl.asm
;           link /LIBPATH:C:\masm32\lib winurl.obj /subsystem:windows
;
;     note: masm32 executables should be in the PATH, if not add them:
;           set %PATH%=%PATH%;C:\masm32\lib
;
```

```
include     kernel32.inc
include     shell32.inc

      .386
      .model flat

      .const
            URL db      'http://edu.mmcs.sfedu.ru/course/view.php?id=15',0

      .code
_start:                 ; entry point label SHOULD strart with underscore ('_')
      xor   ebx,ebx
      push  ebx         ; specify how an application is to be displayed
      push  ebx         ; default (working) directory
      push  ebx         ; parameters to be passed to the application
      push  offset URL  ; the file or object on which to execute
                        ;     the specified operation
      push  ebx         ; operation
      push  ebx               ; owner window
      call  ShellExecute      ; ShellExecute(NULL,NULL,url,NULL,NULL,NULL)
      push  ebx               ; exit status
      call  ExitProcess       ; ExitProcess(0)
end   _start
```

```
; shell32.inc
; exporting functions from shell32.dll system library
;     (using lib-wrapper - shell32.lib)

      includelib  shell32.lib
            extrn __imp__ShellExecuteA@24:dword

      ShellExecute      equ   __imp__ShellExecuteA@24
```

```
; kernel32.inc
; exporting functions from kernel32.dll system library
;     (using lib-wrapper - kernel32.lib)

includelib kernel32.lib
      extrn __imp__ExitProcess@4:dword
      extrn __imp__GetStdHandle@4:dword
      extrn __imp__WriteConsoleA@20:dword
      extrn __imp__GetModuleHandleA@4:dword
      ;extrn      __imp__lstrlen@4:dword ; is absent in WinXP at least
      extrn __imp__GetCommandLineA@0:dword
      extrn __imp__CloseHandle@4:dword
      extrn __imp__GlobalAlloc@8:dword
      extrn __imp__GlobalLock@4:dword
      extrn __imp__GlobalFree@4:dword
      extrn __imp__CreateFileA@28:dword
      extrn __imp__ReadFile@20:dword
      extrn __imp__WriteFile@20:dword

ExitProcess       equ   __imp__ExitProcess@4
GetStdHandle      equ   __imp__GetStdHandle@4
WriteConsole      equ   __imp__WriteConsoleA@20
GetModuleHandle   equ   __imp__GetModuleHandleA@4
;lstrlen          equ   __imp__lstrlen@4
GetCommandLine    equ   __imp__GetCommandLineA@0
CloseHandle       equ   __imp__CloseHandle@4
GlobalAlloc       equ   __imp__GlobalAlloc@8
```

```
GlobalLock        equ   __imp__GlobalLock@4
GlobalFree        equ   __imp__GlobalFree@4
CreateFile        equ   __imp__CreateFileA@28
ReadFile          equ   __imp__ReadFile@20
WriteFile         equ   __imp__WriteFile@20
```

Литература. [1: гл. 7], Wikipedia: x86 Calling Conventions, Name Mangling. Программирование под Windows: [12: ch. 3, 6–8] — в особенности, ch. 3; MSDN: Win32 and COM Development. Создание GUI на Win32 API с использованием языка ассемблера: [1: 7.3], [5: 16], [10: 11.2].

Задание. Создать Win32-приложение с графическим интерфейсом, состоящим из окна, содержащего одну кнопку ‘Exit’.

При нажатии кнопки ‘Exit’, при нажатии на стандартную кнопку закрытия окна («крестик» в правом верхнем углу окна), при нажатии стандартной комбинации клавиш для закрытия окна (Alt+F4) пользователю должно выдаваться окно типа MessageBox с информацией о создателе программы.
Указанное окно MessageBox должно содержать две кнопки: OK и Отмена. В случае нажатия на первую приложение завершается, в случае нажатия на вторую приложение возвращается в исходное состояние («главное» окно с кнопкой ‘Exit’).

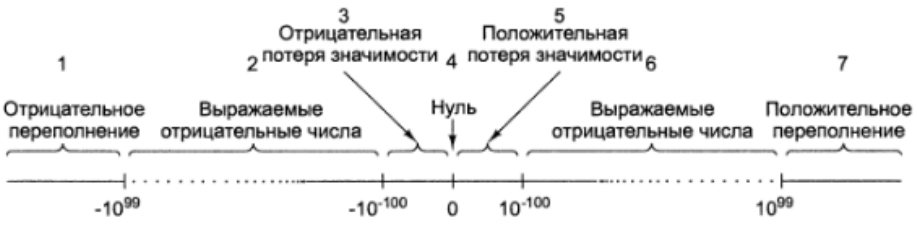
Если в указанном окне MessageBox более 10 секунд не нажато ни одной клавиши, приложение завершается (эффект такой же, как при нажатии на OK).

Примечание: замер времени может осуществляться с помощью системного таймера. Работа с ним организуется посредством функции SetTimer и события WM_TIMER. См. также статью MSDN: Using Timers.

Задание 10. Программирование архитектуры x87 (вычисления с плавающей точкой)

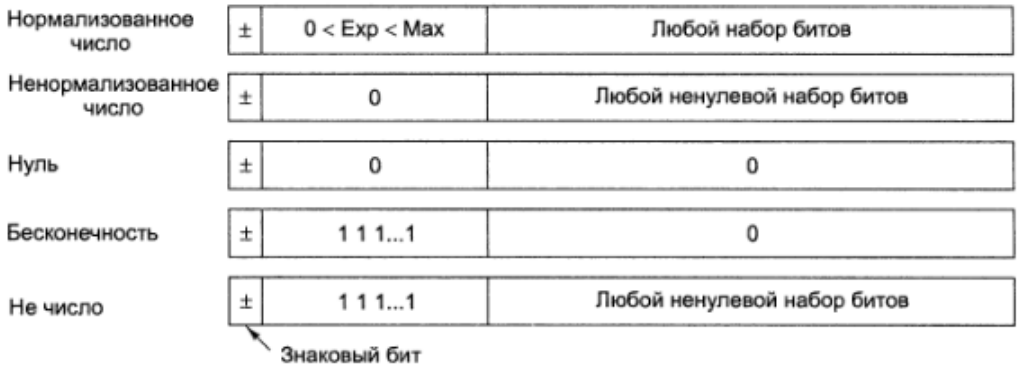
Темы

1. **Принципы представления чисел с плавающей точкой.** Экспоненциальная форма числа, мантисса, экспонента, знак. Нормализованная мантисса. Распределение представляемых в памяти вещественных чисел на вещественной прямой:



и вопросы, возникающие при вычислениях с этими числами (округления, переполнения, деления на нуль...).

2. **Стандарт IEEE 754.** Задачи стандарта. Типы данных, определяемые в стандарте: одинарная точность (32 бита), двойная точность (64 бита), расширенные вещественные (80 бит). Экономия на старшем бите нормализованной мантиссы; смещённая экспонента. Типы значений и их битовое представление (поля диаграммы описывают знак — смещённую экспоненту — мантиссу с неявным старшим битом соответственно):



3. **Архитектура x87**. Сопроцессор и его приемник, *интегрированный FPU* (floating-point unit). Регистры: 8 (вычислительные) + 6 (специальные). Стековый принцип доступа к вычислительным регистрам `R0`–`R7` посредством идентификатора `ST`; указатель на вершину «стека». Размещение обрабатываемых данных — в регистрах и в оперативной памяти (не в регистрах x86 и не в командах, т.е. запрет на непосредственные операнды).

   Формат команд x87: `F{B, I, ε}`*`op`*`[R][P]`. Основные группы команд (одна из возможных классификаций — по расположению операндов).

Литература. Представление вещественных чисел, IEEE 754: [13, прилож. Б], [Wikipedia: IEEE 754]. Программирование x87: [10: 17.4], [9: 18], справочное руководство: [1: 2.4.4–9].

Указание 1: для создания программ, использующих FPU, с помощью wasm нужно использовать ключ ассемблирования `-fpi87` (например, `wasm -fpi87 myfloat.asm`). Можно также использовать Masm32 под Windows.

Указание 2: все подпрограммы, которые необходимо создать, должны принимать параметры («заданные числа») и возвращать результат исключительно *с помощью стека*.

Указание 3: работу каждой подпрограммы надо демонстрировать на нескольких примерах, причём входные данные и результат надо выводить на консоль.

Указание 4: при необходимости использовать константы 0, 1, e, π… воспользуйтесь стандартными константами x87 (см. [1: 2.4.8]).

Вариант 1

1. Внимательно прочтите указания 1–4 выше. Все числа в вашем варианте — вещественные одинарной точности.
2. Напишите процедуру, печатающую переданное число на консоль.
3. Напишите функцию, вычисляющую значение арксинуса заданного числа с помощью его арктангенса, который можно получить стандартной функцией `FPATAN`, см. [1: 2.4.7] (arcsin(x) = 2 arctan(x / (1 + sqrt(1 – x^2)))).
4. Напишите функцию, считающую значение sin(x) в заданной точке x с помощью ряда Тейлора для синуса (sin(x)= x – x^3/3! + x^5/5! – …) с заданной точностью *eps*. Покажите, что значение, возвращаемое вашей функцией, действительно отличается не более чем на *eps* от истинного значения синуса, получив последнее с помощью стандартной функции `FSIN`.

Вариант 2

1. Внимательно прочтите указания 1–4 выше. Все числа в вашем варианте — вещественные двойной точности.
2. Напишите процедуру, печатающую переданное число консоль.
3. Напишите функцию, вычисляющую значение арккосинуса заданного числа с помощью его арктангенса, который можно получить стандартной функцией `FPATAN`, см. [1: 2.4.7] (arccos(x) = 2 arctan(sqrt(1 – x^2) / 1 + x), $0 \le x \le 1$).
4. Напишите функцию, считающую значение cos(x) в заданной точке x с помощью ряда Тейлора для косинуса (cos(x)= $1 - x^2/2! + x^4/4! - \ldots$) с заданной точностью *eps*. Покажите, что значение, возвращаемое вашей функцией, действительно отличается не более чем на *eps* от истинного значения косинуса, получив значение последнего с помощью стандартной функции `FCOS`.

Примеры заданий итоговой контрольной работы

Создайте процедуру, определяющую, является ли заданная строка палиндромом (то есть симметричной относительно середины, например, «abcba», «abba»); параметры и результат процедуры передаются через стек. Выведите на консоль результаты работы процедуры для разных входных данных.

- Реализуйте алгортим поиска простых чисел, не превосходящих n, «Решето Эратосфена». Для этого: (1) выделите память из n слов (n определяется статически), заполните её числами от 2 до n; (2) организуйте цикл, забивающий составные числа, кратные каждому последующему простому числу, нулями; (3) сдвиньте выделенные простые числа так, чтобы между ними не стояло нулей; распечатайте результат на консоль. Преобразования (1)–(3) должны давать такой эффект:
  - $\xrightarrow{(1)}$ 2, 3, 4, 5, 6, 7, 8, . . . $\xrightarrow{(2)}$ 2, 3, 0, 5, 0, 7, 0, . . . $\xrightarrow{(3)}$ 2, 3, 5, 7, . . .

- Создайте подпрограмму, которая по заданным коэффициентам {ai} находит значение многочлена $p(x) = a_0 + a_1x + \cdots + a_nx_n$ в заданной точке x_0. Параметры должны передаваться через стек. Для вычисления следует использовать формулу
$$p(x) = a_0 + x(a_1 + x(a_2 + ... + x(a_{n-1} + anx) ...)).$$
(Вычисление нужно проводить, начиная с самой глубоко вложенной пары скобок.) Выведите на консоль результаты работы подпрограммы для нескольких наборов входных данных.
- Создайте подпрограмму, которая по заданным начальным значениям $\{a_i\}$ и коэффициентам $\{f_i\}$ находит значение n-го члена линейной рекуррентной последовательности (ЛРП): $a_n = f_1a_{n-1} + f_2a_{n-2} + ... f_ka_{n-k}$. Параметры должны передаваться через стек. Продемонстрируйте работу программы для ЛРП:
$$a_n = 4a_{n-1} + 5a_{n-2} + 2a_{n-3}\ ,\ a_1 = 1,\ a_2 = 3,\ a_3 = 6,$$
распечатав на консоль её члены a_{2n}, n = 2, 3, ... , 5.

- Написать рекурсивную процедуру, возводящую квадратную матрицу A в неотрицательную степень n, используя алгоритм:

$$\mathrm{Pow}(A,n) = \begin{cases} E, & \text{если } n = 0; \\ A \cdot \mathrm{Pow}(A, n-1), & \text{если } n \text{ нечётно}; \\ (\mathrm{Pow}(A, n/2))^2, & \text{если } n \text{ чётно}. \end{cases}$$

Процедура не должна использовать никаких глобальных данных. Вывести результат работы процедуры на консоль, подсветив красным цветом максимальный элемент в полученной матрице.

- В отдельном модуле реализуйте процедуру вывода на экран информации о дескрипторе сегмента защищённого режима, адрес которого (сегмент и смещение) передаётся через стек. Базовый адрес (как и размер) сегмента должен печататься одним числом, скрывая тем самым особенности размещения этого значения в дескрипторе. Для вывода информации о типе сегмента (биты 41–43 дескриптора) вместо большого числа условных операторов удобно использовать массив строк `types`, индексы которого будут значениями поля типа сегмента (например, `types[3] = "Stack, read/write"`). Продемонстрируйте работу процедуры, написав вызывающую её «основную программу».
- В отдельном модуле реализуйте (рекурсивный) алгоритм Быстрой сортировки для массивов целых 16-разрядных чисел с максимальным использованием цепочечных команд языка ассемблера. Адрес массива (сегмент и смещение) передаётся через стек. Продемонстрируйте работу процедуры, написав вызывающую её «основную программу», выведите на консоль результаты.

Задачи по заданию 3

- На пустом экране организовать перемещение курсора при нажатии клавиш w-s-a-d (вверх-вниз-влево-вправо, соответственно), нажатие остальных клавиш игнорировать. При нажатии одной из клавиш предыдущая позиция закрашивается фиксированным цветом; экран очищается каждый раз, когда очередная нажатая управляющая клавиша не совпадает с последней нажатой управляющей.
  Указание: использовать прерывание DOS для ввода символа с клавиатуры без эха (AH=08H).
- В центре экрана по горизонтали, с заданным отклонением от центра колеблется маркер (одна позиция экрана с нестандартным цветом фона). При нажатии на клавиш a и d отклонение уменьшается и увеличивается, соответственно.

  Указание: использовать прерывание DOS для ввода символа с клавиатуры без эха (AH=08H) и для проверки состояния клавиатуры (AH=0Bh).

Задачи по заданию 4

- Создать рекурсивную подпрограмму, вычисляющую двойной факториал числа:

  $$N!! = N\cdot(N-2)\cdot(N-4)\cdot\ldots$$

  Параметр и результат передаются на стеке.
- Создать рекурсивную подпрограмму, вычисляющую заданную степень числа по формуле:

  $$X^N = (X^{N/2})^2 \text{ при четных } N > 0,$$

  $$X^N = X\cdot X^{N-1} \text{ при нечетных } N > 0,$$

Параметры и результат передаются через стек.

Задачи по заданию 5

- Описать структуру Firefox (красная панда) с полями weight (вес, целое число), age (возраст, целое число) — в отдельном включаемом inc-файле. В отдельном файле реализовать процедуру вычисления суммарного веса в заданном массиве записей о красных пандах (адрес и размер массива, результат передаются через стек), в теле процедуры использовать оператор ‘.’ обращения к полю структуры. В отдельном файле сформировать массив красных панд и продемонстрировать работу созданной процедуры (вывести на консоль результат).
- Описать структуру Person (человек) с полями surname (10 символов), age (возраст) — в отдельном включаемом inc-файле. В отдельном файле реализовать процедуру печати на консоль фамилий людей старше заданного возраста из заданного массива записей о людях (адрес и размер массива, минимальный возраст передаются через стек), в теле процедуры использовать оператор ‘.’ обращения к полю структуры. В отдельном файле сформировать массив людей и продемонстрировать работу созданной процедуры.

Список литературы

1. Зубков С.В. Assembler. Для DOS, Windows и Unix. 2-е изд. / М.:ДМК — 2000.
2. Юров В., Хорошенко С. Assembler: учебный курс. / СПб.:Питер Ком, 1999.
3. Пирогов В.Ю. Ассемблер для Windows. / М.: Издатель Молгачева С.В., 2002.
4. Юров В.И. Assembler. Практикум. 2-е изд. / СПб.: Питер, 2006.
5. Юров В.И. Assembler. Учебник для ВУЗов. 2-е изд. / СПб: Питер, 2003.
6. Пильщиков В.Н. Программирование на языке ассемблера IBM PC. / М.:Диалог-МИФИ, 1998.
7. Финогенов К. Г. Самоучитель по системным функциям MS-DOS — 3-е изд. / М.:Горячая линия — Телеком, 2001.
8. Carter P. PC Assembly Language. / Avail. online, 2005.
9. Dandamudi S.P. Introduction to Assembly language programming / Springer, 2005.
10. Ирвин К. Язык ассемблера для процессоров Intel / 4-е изд. Вильямс, 2005.
11. Кулаков В. Программирование на аппаратном уровне. — 2-е изд. / СПб: Питер, 2003.
12. Petzold. Programming Windows, 5th ed. / Mic. Press, 1998.
13. Таненбаум Э. Архитектура компьютера / Питер, 2007.

Ресурсы интернета:

- Сайт, посвящённый языкам ассемблера x86: http://webster.cs.ucr.edu/
- Группа новостей comp.lang.asm.x86
- ЖЖ-сообщество assembler