Е.А. Бордаченкова, И.В. Горячая
Методические указания к пособию
«Задачи и упражнения по языку ассемблера MASM»

Лекционный курс «Архитектура ЭВМ и язык ассемблера» содержит, в частности, две взаимосвязанные части:
1. Модельные ЭВМ 2. Ассемблер MASM.
Первая часть предназначена
1) для объяснения принципов работы компьютеров;
2) для обсуждения различных архитектурных моделей, которые существуют и сейчас;
3) для подготовки студентов к изучению ассемблера MASM.
На семинарских занятиях делается акцент на 1 и 3 пунктах, прорабатываются только те модели, которым имеется соответствие в архитектуре IA-32.
Вторая часть курса конкретизирует материал первой части; привязывает к реальному миру идеальные архитектурные модели.

Язык MASM рассматривается как пример ассемблера – т.е. как машиннозависимый язык. Поздние наслоения создателей языка с целью приблизить MASM к языкам высокого уровня не включены в курс по двум причинам: во-первых, в курсе важно сделать очевидной связь средств языка MASM с архитектурными особенностями процессора IA-32; во-вторых, в курс отбирались конструкции, обработка которых ассемблером более-менее предсказуема, реализована без ошибок.

Язык ассемблера и особенности архитектуры IA-32 изучаются на логическом уровне. Это значит, например, что студенты не должны уметь выписывать машинные команды с расшифровкой битов, но должны понимать структуру машинной команды (что в машинной команде есть поля – группы битов; что есть поле для КОП и поле для операндов) и понимать, как ассемблерная команда соотносится с машинной.

Необходимо добиться, чтобы на семинарских занятиях у студентов сложилось представление о языке MASM не как о наборе беспорядочных фактов, чтобы студенты поняли логику, которая всё же есть в архитектуре IA-32 и в MASM'е, которая вытекает из
1) базовых архитектур УМ-Р, УМ-М
2) принципов работы ассемблера.
Следовательно, мы избегаем перегружать студентов случайными возможностями языка.

Важно, чтобы студенты чётко понимали, что правила языка ассемблера основаны на архитектуре процессора и на принципах работы программы-ассемблера. Для того, чтобы достичь такое понимание, нужно
1) напоминать архитектурные особенности, изученные в теме Модельные ЭВМ (при изучении арифметических команд, команд переходов, массивов) и
2) разбирать, как ассемблер обрабатывает (транслирует) конструкции языка.
С этой целью в помощь преподавателям в задачнике предложены задачи на анализ листинга.

Методически принципиально, чтобы при решении задач студенты использовали **только тот материал, который приведён в задачнике** в теоретической части текущего параграфа и предыдущих параграфов.

В каждой теме есть наиболее важные моменты, которые студенты должны понять для полноценного освоения курса. Эти моменты перечислены в данных методических указаниях соответствующего параграфа задачника. Там же указаны наиболее существенные задачи. Как правило, на каждую особенность языка ассемблера, на каждый приём программирования в задачнике даны несколько задач (либо несколько пунктов в одной задаче) для того, чтобы преподаватели имели возможность выбрать задачи, более подходящие для решения на семинаре, и задачи для включения в домашнее задание. На каждый приём студент, как правило, должен решить хотя бы две задачи. В «Методических указаниях» даны примеры самостоятельных работ по каждому параграфу задачника.

1. Директивы определения данных. Директивы "=" и equ. Адресные и константные выражения

Цель занятия

1. Уяснить соответствие между описательными предложениями программы и байтами объектного кода.
2. Понять процесс выполнения директив ассемблером: заполнение байтов объектного кода, построение таблицы имён.
3. Понять и запомнить действие директив для разных видов операндов.
4. Запомнить синтаксис конструкций.
5. Разобраться с выражениями:
   - они вычисляются ассемблером, а не процессором;
   - разные способы записи константных и адресных выражений, смысл операторов;
   - как вычисляется значение выражения;
   - в чём принципиальная разница между константными и адресными выражениями.

Важные задачи: 1.1, 1.2, 1.4, 1.6, 1.9. 1.10, 1.15, 1.17, 1.18, 1.21, 1.24. 1.25

2. Команды пересылок. Оператор ptr.

Цель занятия

1. Запомнить общие требования к операндам команд, основанные на особенности системы команд процессора: одинаковый размер операндов; два операнда из ОП не могут использоваться.
2. Научиться определять вид операнда: регистр, память (адресное выражение), непосредственный (константное выражение). Понять, как проверять правильность записи команд.
3. Повторить особенности представления чисел: перевёрнутое представление, как выглядит машинное представление разной длины (байт, слово, двойное слово, четверное слово) для одного и того же числа.
4. Выучить команды пересылок.

Важные задачи: 2.1, 2.2, 2.4, 2.5, 2.7, 2.8, 2.12, 2.13, 2.14, 2.15, 2.16

Самостоятельная работа

Задача 1. Определить, какое число окажется в регистре EAX в результате выполнения указанного фрагмента. Ответ дать в **16-ричной** системе счисления.

```
; описания
N DW 2h, 0CDEh, 0Fh
M DD 12345h
; команда
  MOV EAX, M-3          ; EAX - ?
```

Ответ: EAX = 45000F0Ch

Задача 2. Определить, какое число окажется в регистре BX в результате выполнения указанного фрагмента. Ответ дать в **десятичной** системе счисления.

```
; описание
Y DB  132
; команда
  MOVSX  BX, Y           ; BX - ?
```

Ответ: BX = -124

3. Арифметические команды.

Два семинара:

1. Команды сложения и вычитания (ADD, ADC, SUB, SBB, INC, DEC, NEG)
2. Команды умножения и деления (MUL, IMUL, DIV, IDIV, CBW, CWD, CDQ)

Цель занятий

1. Запомнить общие требования к операндам команд, основанные на особенности системы команд процессора: одинаковый размер операндов; два операнда из ОП не могут использоваться.
2. Запомнить арифметические команды, особенности их работы и требования на операнды, вытекающие из семантики команд.
3. Повторить правила установки арифметических флагов.
4. Уяснить преимущества, которые даёт дополнительный код для представления чисел со знаком (общий алгоритм для работы с числами без знака и с числами со знаком). Реализовать алгоритмы сложения и вычитания длинных чисел в столбик.

Важные задачи

Семинар 1: 3.1, 3.6, 3.8, 3.9, 3.11, 3.12, 3.13, 3.14
Семинар 2: 3.15, 3.16, 3.17, 3.18, 3.20, 3.21, 3.22, 3.24 (3.29), 3.25 (3.28), 3.31, 3.32

Самостоятельная работа

Задача 1.

```
X DQ ?
A DB ?      ;  числа без знака
```

Реализовать следующий оператор присваивания: X := X+A. Использовать не более трёх команд.

Задача 2.

```
A DW ?
B DB ?      ;  числа со знаком
```

Реализовать следующий оператор присваивания: B := (A-13) mod B.

Указание для преподавателей: разность (A-13) необходимо формировать в виде двойного слова; размер делителя – двойное слово (деление на слово не подойдёт !)

4. Команды ввода и вывода. Структура программы.

Цель занятия

1. Сопоставить структуру машинной программы для УМ с объектным кодом и с ассемблерной программой (блок команд, блок данных).
2. Разобрать, по каким предложениям программы генерируется объектный код (заполняются байты), и какие предложения чисто информативные.
3. Разобрать особенности команд ввода-вывода.
4. Повторить представление чисел (неразличимость машинного представления чисел со знаком и чисел без знака). Повторить правила записи выражений, потренироваться определять адрес конкретного байта относительно начала строки, научиться использовать разные константные и адресные выражения.

Некоторые задачи удобно давать решать на компьютерах.

Важные задачи: 4.1, 4.3, 4.4, 4.5, 4.6, 4.8, 4.11, 4.16, 4.17, 4.19, 4.21

Самостоятельная работа (выполняется на компьютере)

Задача 1. Написать программу решения задачи. Ввести неотрицательное число (по inint). Используя outchar, напечатать младшую цифру в записи этого числа в троичной системе.

Задача 2. Написать программу, которая печатает в отдельных строках Ваше

1) имя;
2) фамилию;
3) имя и фамилию.

5. Команды перехода

Цель занятия

1. Важно выработать у студентов привычку разрабатывать схемы программного управления на паскале, прежде чем писать текст на ассемблере.
2. Практика в реализации паскалевских управляющих конструкций на ассемблере в соответствии со схемами, данными в лекциях. Разобрать, почему при написании программ на ассемблере укороченный условный оператор предпочтительней полного условного оператора и почему цикл с постусловием лучше цикла с предусловием. Практиковаться в реализации паскалевского цикла for.
3. Дать понимание, что команды условного перехода анализируют текущее значение флагов. Напомнить, что арифметические команды устанавливают флаги; объяснить, в каких ситуациях команду сравнения можно не использовать.
4. Запомнить названия и особенности команд переходов.

Важные задачи: переходы 5.1, 5.2, 5.4, 5.5, 5.8, 5.12, 5.18; циклы 5.6, 5.14, 5.34, 5.20, 5.19, 5.29, 5.41, 5.48.

Самостоятельная работа

Задача 1. Определить, на какую метку будет сделан переход

```
; .data
R DB 250
; .code
  SUB R, 20
  jGE M1
  jAE M2
  JMP M3
```

Ответ: M2

Задача 2.

`N = 50`

Написать *фрагмент программы* (необходимые описания и команды) решения следующей задачи: Дана последовательность N чисел. Напечатать первое отрицательное число последовательности. Если все числа в последовательности неотрицательные, напечатать «Нет отрицательных чисел». *Ограничение:* печать должна быть вне цикла.

6. Массивы

Цель занятия

1. Выучить синтаксис описания массивов и операторы ассемблера, облегчающие работу с массивами.
2. Вспомнить понятие исполнительного адреса из темы Модельные ЭВМ; разобраться с тем, как транслируются адресные выражения с модификаторами (в машинной команде, кроме поля адреса операнда, есть поля для модификатора и множителя).
3. Разобраться в правилах записи адресных выражений на ассемблере.
4. Понять соотношение между индексом элемента массива и смещением элемента относительно начала массива (влияние типа элементов и способа нумерации элементов).
5. Освоить приёмы программирования:
   1) просмотр массива от начала к концу, перемещая модификатор на размер элемента
   2) просмотр массива от конца к началу, адресуясь с помощью ECX
   3) работа с двух концов массива с помощью двух модификаторов
   4) реализация аналога массива вида array[‘a’..’z’] of byte.

Важные задачи: 6.5, 6.1, 6.2, 6.3, 6.4, 6.7, 6.9, 6.10, 6.11, 6.13, 6.15, 6.16 (решить двумя способами: без использования множителей при модификаторе; используя множитель и адресуясь по ECX; сравнить два решения), 6.20, 6.23, 6.28, 6.32

Опыт показывает, что желательно провести **два** семинара по теме «массивы». **На первом** разобрать теоретический материал, команду LEA, решить алгоритмически простые задачи. **На втором** семинаре решить алгоритмически сложные задачи и задачи на применение массивов, работа с матрицами.

Самостоятельная работа

Задача 1. Есть следующие описания, требуется обнулить последний элемент массива X и последний элемент массива Y. Описать текстовые константы L_X и L_Y (вставить тексты в уголки вместо **???**) так, чтобы команды верно решали эту задачу. *Ограничение:* вставленные тексты должны отличаться не более, чем на два символа.

```
;.DATA
   N = 100
   X DB  4  DUP(?)
   Y DD  N  DUP (?) ; Y[0..N-1]
 L_X EQU <   ???   >
 L_Y EQU <   ???   >
;.CODE
   MOV X + L_X, 0
   MOV Y + L_Y, 0
```

Задача 2. Есть следующие описания:

```
Z DW  100 dup(?)    ;  Z: array [3..102] of 3..102
i DB ?              ;  i∈[3,102]
```

1) Выписать формулу вычисления адреса (в сегменте данных) элемента Z[i]
2) Реализовать присваивание: **Z[Z[i]] := i**. *Ограничение*: использовать не более двух команд.

7. Структуры

Цель занятия

1. Разобрать синтаксис описания типов структур и описания переменных; инициализация переменных значениями по умолчанию.
2. Разобрать хранение структур в объектном коде, значение имени поля.
3. Отработка применения оператора точка для доступа к полю переменной: синтаксис записи адресных выражений и трансляция адресных выражений *имя переменной.имя поля* и *имя переменной* [*модификатор*].*имя поля*. (Можно проанализировать листинг.)
4. Обратить внимание студентов, что для массивов структур не подходит адресация с множителем при модификаторе.

Важные задачи: 7.1, 7.2(а), 7.3, 7.4(в), 7.5, 7.6(в)

Самостоятельная работа

Даны описания:

```
TS STRUC
    A DB ?
    B DW -2
    C DB '1234567'
    D DD 8
TS ENDS
Arr  TS  100  dup(< >)         ; Arr[0..99]  of  TS
```

Выполнить следующие задания:

№1. Указать значения выражений:

`type Arr`	`type Arr.C`	`length Arr`	`size Arr`

№2. Пусть требуется реализовать оператор AX:=Arr[5].B. Определить правильные варианты решения задачи, для неверных объяснить ошибку:

1	`MOV AX, Arr[5].B`
2	`MOV AX, Arr[5*(type Arr)].B`

3	`LEA ESI, Arr[5*(type Arr)].B` `MOV AX, [ESI]`
4	`LEA EDI, Arr[5*(type Arr)]` `MOV AX, [EDI].B`

8. Битовые команды

Тема важная, поскольку в этих командах и выражается преимущество программирования на ассемблере – быстрые вычисления, упаковка данных, представление множеств. Для студентов тема является сложной, т.к. нет ей аналога в материале прошлого семестра.

Цель занятия

В результате изучения материала и решения задач студенты должны

1. Выучить битовые команды; уяснить, что битовые команды работают быстро (т.к. процессор построен на 2-ной системе).
2. Научиться выделять группы разрядов логическим умножением на маску.
3. Научиться сдвигать несколько ячеек как единый объект: записать разряд, выдвинувшийся из одной ячейки, в другую.
4. Работать с CF (через ADC).
5. Использовать битовые команды вместо арифметических умножений и делений.

Важные задачи: 8.1, 8.4, 8.5, 8.6, 8.7, 8.10. 8.11, 8.20, 8.21, 8.25.

Самостоятельная работа

Задача 1.

Особенностью **ASCII**-кодировки *латинских букв* является то, что буквы *верхнего* и *нижнего регистров* различаются в своём двоичном представлении *только одним битом*. Достигается это за счёт того, что данные буквы представляются *своими порядковыми номерами в алфавите* (при нумерации букв *от единицы*), записанными **пятью двоичными цифрами**, перед которыми стоит $\mathbf{010_2}$ (для букв *верхнего регистра*) или $\mathbf{011_2}$ (для букв *нижнего регистра*), например, код *большой* буквы **'A'** равен **01000001b**, а код *маленькой* буквы **'a'** равен **01100001b**. С учётом указанной особенности выполнить следующее преобразования:

пусть в регистре **AL** хранится некоторая *большая* латинская буква, требуется заменить её на одноимённую *маленькую*.

Задача 2. *Фрагмент программы*. Есть описание:

```
X DD ?  ; число без знака
```

Реализовать оператор присваивания: X := X div 16 + X mod 16

9. Записи

Цель занятия

1. Разобрать синтаксис описания типа записи и синтаксис описания переменной.
2. Научить студентов выделять поля записи, печатать и заполнять поля.
3. Продемонстрировать значение порядка расположения полей для удобства работы с записями (например, сравнить год.месяц.день и день.месяц.год с точки зрения сравнения дат), показать аналогию записей с позиционными системами счисления.

Важные задачи: 9.1, 9.2, 9.3, 9.4 (б, в, д), 9.5 (б, в), 9.7.

Самостоятельная работа

Даны описания:

```
TR RECORD A:1, B:2, C:3
R TR  <>
```

№1. Указать значения выражений:

`type R`	`mask C`	`width A`	`A`	`B+C`

№2. *Фрагмент программы*. Напечатать значения полей записи R в виде чисел без знака.

10. Стек

Цель занятия

1. Разобрать семантику команд push и pop.
2. Научить студентов работать со стеком с адресацией по EBP.
3. Порешать задачи на использование стека в разных задачах (печать в обратном порядке, вычисление формул и т.п.)

Важные задачи: 10.1, 10.3, 10.5, 10.9, 10.10, 10.11, 10.12, 10.13.

Самостоятельная работа

Задача 1. *Фрагмент программы*. Реализовать последовательность операторов

```
read(N); {dword, N>0}
for i:= N downto 1 do
begin
    for j:= i downto 1 do write (j);
    writeln
end
```

Для реализации вложенных циклов использовать стек.

Задача 2. *Фрагмент программы*. В стеке 35 элементов (двойных слов). Записать в EAX количество элементов – чётных чисел. Исходное состояние стека и регистры не портить. Для адресации по стеку использовать EBP (сохранив его значение).

11. Процедуры

При изучении этой темы важно не только научить студентов средствам ассемблера, но и усилить опыт студентов осеннего семестра по программированию сверху вниз.

Два семинара:

1. Передача параметров в регистрах.
2. Передача параметров в стеке. Рекурсия.

Цель занятий

1. Объяснить семантику команд CALL и RET.
2. Натренировать студентов изображать кадр стека, соответствующий процедуре, особенно при передаче параметров в стеке, отмечать базу кадра EBP.
3. Научить передавать параметры по ссылке и по значению (в регистрах; в стеке); обрабатывать в процедуре параметры, переданные по ссылке и по значению.
4. Научить грамотно оформлять процедуры: указание интерфейса процедуры в виде комментария, сохранение используемых регистров, пролог и эпилог.
5. Потренировать студентов в написании ведущей части программы, использующей процедуры с фиксированным интерфейсом, без написания самих процедур. (Передача параметров в регистрах; в стеке.) Хороший приём: один студент программирует ведущую часть, другой описывает процедуру.

Важные задачи

Семинар 1: 11.1, 11, 2, 11.3(а), 11.6, 11.7, 11.12, 11.14
Семинар 2: 11.3(б) (+ нарисовать кадр стека), 11.4, 11, 5, 11.10, 11.12, 11.15, 11.20, 11.28

Самостоятельная работа

```
A DD ?   ;  со знаком
B DD ?   ;  со знаком
```

Требуется реализовать процедуру

```
procedure absolute(var x: dword; y: dword);
begin x:= abs(y) end;
```

Выписать последовательность команд, реализующую вызов `absolute(B,A+2)`.

№1. Параметры передать *в регистрах*.
№2. Параметры передать *в стеке*.

12. Строковые команды

Строковые команды важны для развития алгоритмических навыков студентов, для улучшения программистской квалификации. Строковые команды дают возможность формулировать алгоритм обобщённо, в терминах работы с целым массивом (фрагментом массива), а не в терминах перебора отдельных элементов. Такой подход позволяет также сфокусировать внимание студентов на постусловии цикла (результате его работы).

В начале параграфа даны тренировочные задачи на освоение строковых команд: понимание работы команд, применение пересылок, применение сравнений. Далее приведены задачи на использование строковых команд при решении более общих задач. В конце параграфа рассматриваются строки переменной длины.

Важные задачи: 12.1, 12.3, 12.4, 12.6, 12.7, 12.8, 12.9, 12.11, 12.16, 12.17 и 12.18 (один-два пункта).

Самостоятельная работа

```
Len = 15
  A DB Len DUP (' ')
  B DB Len DUP (' ')
  K DB ?
```

Используя строковые команды *с префиксами повторения*, решить задачи.

Задача 1. Реализовать оператор B:= A.

Задача 2. Массив A содержит слово из маленьких латинских букв. Если в слове меньше, чем Len букв, в конце стоят пробелы, например, A='forest·····' (здесь · обозначает пробел). Записать в K реальную длину слова A (количество букв в слове). *Подсказка*: учесть ситуацию, когда в A нет пробелов.

13. Макросредства

Принципиальный момент при изучении темы: объяснить, что макросредства работают с текстом, что это механизм замены одного текста на другой текст. Акцентировать внимание студентов на то, что в процесс трансляции программы вводится этап макрогенерации. Тема сложная, т.к. не имеет аналогии с осенним семестром.

Два семинара:

1. IF, блоки повторения, макросы, LOCAL; логика работы макрогенератора.
2. Тонкости: поиск формальных параметров, виды фактических параметров, вложенные макросы (блоки повторения).

Цель занятия

1. Сформировать у студентов понимание, какие действия (вычисления) может выполнить макрогенератор (а какие не может), в чём суть этапа макрогенерации.
2. Разобрать синтаксис и семантику конструкций макросредств.
3. Научить студентов посмотреть результат работы макрогенератора. (Создать листинг с раскрытыми макросредствами.)
4. Важный момент: студенты должны понимать, что прежде, чем писать макрос (блок повторения) нужно чётко понимать, какой текст должен получиться в результате работы макрогенератора. Метод: сначала написать ассемблерный текст, затем сократить его, используя макросредства.

Важные задачи

Семинар 1: 13.1, 13.3(без (в)), 13.4, 13.14, 13.9, 13.10, 13.15.
Семинар 2: 13.3(в), 13.6, 13.8, 13.37, 13.16, 13.19, 13.20 (рекурсивно и нерекурсивно), 13.24. 13.34.

Самостоятельная работа

Задача 1.

Описать макрос IsEq X,L (где X – имя *знаковой* переменной типа byte, word, dword или qword, L – метка), реализующий действие **if** x=-5 **then goto** L **else** x:= 10. Макрос должен генерировать минимальное число команд.