Е.А. Бордаченкова, И.В. Горячая

Методические рекомендации для преподавателей практикума по курсу "Архитектура ЭВМ и язык ассемблера"

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 принципа 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.

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

во-первых, напоминать архитектурные особенности, изученные в теме Модельные ЭВМ (при изучении арифметических команд, команд переходов, массивов) и

во-вторых, 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 методических указаниях соответствующего параграфа задачника. Там же указаны наиболее существенные задачи. Как правило, на каждую особенность ассемблера, на каждый приём программирования в

задачнике даны несколько задач (либо несколько пунктов в одной задаче) - для решения на семинаре и для включения в домашнее задание.

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. Процесс выполнения директив ассемблером: заполнение байтов объектного кода, построение таблицы имён.
3. Понять и запомнить действие директив для разных видов операндов.
3. Запомнить синтаксис конструкций.
4. 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: 1.4, 1.9, 1.17, 1.18, 1.19, 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. Понять, как проверять правильность записи команд, научиться определять вид операнда: регистр, память (адресное выражение), непосредственный (константное выражение).
3. Повторение особенностей 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, 2.3, 2.5, 2.7, 2.8, 2.11, 2.15

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, 3.2 (3.3, 3.4), 3.6, 3.9 (3.10), 3.11, 3.12(3.13)
Семинар 2: 3.14, 3.15, 3.16, 3.17, 3.19 (3.20), 3.21, 3.23, 3.24, 3.27, 3.28, 3.30,3.31

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.

Важные задачи: 4.3, 4.5, 4.6, 4.9, 4.10, 4.19(а), 4.20

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.
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: 5.1, 5.2, 5.4, 5.19 (в), 5.20, 5.34, 5.30.

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 (для массивов разных типов).
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, 6.13, 6.15 (решить двумя способами: 1) без использования множителей при модификаторе; 2) используя множитель и адресуясь по ECX; сравнить два решения), 6.22, 6.27

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.

7. Структуры

Цель занятия
1. Разобрать синтаксис описания типов структур и описания переменных; инициализация переменных значениями по умолчанию.
2. Применение оператора точка для доступа к полю переменной. Трансляция адресного выражения *имя переменной*.*имя поля*. (Можно проанализировать листинг.)
3. 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(б)

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(г, д, е), 8.5(а, б), 8.7(в, г, ж, и, з, к), 8.9. 8.10, 8.21 (а, без переполнения), 8.13(а), 8.23, 8.24(а, б, д).
Домашнее задание: 8.6, 8.8, 8.11, 8.12, 8.20(а), 8.13(б), 8.18, 8.25.

Замечание. Можно потратить на тему 1,5 семинара: 1-й семинар – битовые команды, 2-й семинар – битовые команды и записи.

9. Записи

Цель занятия
1. Разобрать синтаксис описания типа записи и синтаксис описания переменной.
2. Научить студентов выделять поля записи.
3. Продемонстрировать значение порядка расположения полей для удобства работы с записями, показать аналогию с позиционными системами счисления.

Важные задачи: 9.1, 9.3, 9.4, 9.6.

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, 10.5, 10.3, 10.9, 10.11, 10.12, 10.13.

11. Процедуры

При изучении этой темы важно не только научить студентов средствам ассемблера, но и усилить опыт студентов осеннего семестра по нисходящему программированию.
Два семинара:

1. Передача параметров в регистрах.
2. Передача параметров в регистрах. 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 main, 11.9 main, 11.10, 11.11, 11.12.
Семинар 2: 11.3(б), 11.4, 11.10, 11.11, 11.12, 11.20, 11.25.

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: 12.3, 12.4(а,в), 12.8, 12.7(б), 12.9(а), 12.12(а), 12.18 и 12.19 (один-два пункта).
Домашнее задание: 12.2, 12.5(б), 12.9(б), 12.13(а), 12.16, 12.18 и 12.19 (один пункт).

13. Макросредства

Принципиальный момент при изучении темы: объяснить, что макросредства работают с текстом, что это механизм замены одного текста на другой текст. 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, 13.4, 13.14, 13.9, 13.10, 13.19, 13.20.

Семинар 2: 13.3 (в), 13.6, 13.8, 13.37, 13.16, 13.24. 13.34