**ПЛАН ЗАНЯТИЯ**

**Дисциплина:** МДК.01.04 Системное программирование

**Преподаватель:** Галузин А.Б.

**Курс:** 4

**Группа:** П-40

**Специальность:** 09.02.07 Информационные системы и программирование

**Дата:** 19.09.24

**Время проведения:** 11.50-13.25, 3 пара

**Тема:** Синтаксис ассемблера

**Цель занятия:**

**дидактическая:** изучить синтаксис ассемблера

**развивающая**: развивать абстрактное мышление, логику

**Вид занятия** лекция

**Литература**

Юров В.И. Assembler. Учебник для вузов. - 2-ое изд. – СПб.: Питер, 2003, стр. 20.

**Интернет-ресурсы:**

<https://xakep.ru/2017/09/11/asm-course-1/>

**ЗАДАНИЕ**: законспектировать лекцию с учетом контрольных вопросов.

**КОНСПЕКТ ЛЕКЦИИ**

План

1. Синтаксис ассемблера.
2. Операнды.
3. Типы данных**.**
4. **Синтаксис ассемблера**

Общепринятого стандарта для синтаксиса языков ассемблера не существует. Однако большинство разработчиков языков ассемблера придерживаются общих традиционных подходов. Одним из основных таких стандартов — Intel-синтаксис. Программа на ассемблере является совокупностью блоков памяти - сегментов. Сегменты программы имеют определенное назначение: кода, данных и стека. Названия типов сегментов отображают их назначение. Распределение программы на сегменты отображает сегментную организацию памяти процессоров Intel. Каждый сегмент состоит из совокупности отдельных строк - предложений языка.. Для ассемблера предложения, составляя программу, могут быть синтаксическими конструкциями четырех типов.

* ***Команды*** (инструкции) - аналоги машинных команд. В процессе трансляции они превращаются в соответствующие команды системы команд процессора.
* ***Макрокоманды*** — это предложения текста программы, которые оформляются определенным образом, замещаются во время трансляции другими предложениями (строка, содержащая символическое имя, предназначенное для замещения одной или несколькими другими строками). Имя макрокоманды может сопровождаться параметрами.
* ***Директивы*** являются указанием транслятору ассемблера на выполнение некоторых действий. У директив нет аналогов в машинном представлении.
* ***Комментарии*** содержат любые символы, в том числе и буквы русского алфавита. Комментарии игнорируются транслятором.

**Алфавит ассемблера**

* АSCII\_символ\_буква — все латинские буквы А - Z, а - z, причем прописные и строчные буквы считаются эквивалентными;
* decdigit — цифры от 0 до 9;
* специальные знаки \_, ?, @, $, &;
* разделители: „ ., [, ], (,), <, >, {, }, +, /, \*, %,!," ?, \, =, #, ^.

***Лексемами*** языка ассемблера являются ключевые слова, идентификаторы, цепочки символов и целые числа.

***Ключевые слова*** — это служебные символы языка ассемблера. По умолчанию регистр символов ключевых слов не имеет значения. К ключевым словам относятся:

* названия регистров;
* операторы (BYTE, SBYTE, WORD, SWORD, DWORD, SDWORD, FWORD, QWORD, TBYTE REAL4, REAL8, REAL10, NEAR16, NEAR32, FAR16, FAR32, AND, NOT, HIGH, LOW, HIGHWORD, LOWWORD, OFFSET, SEG, LROFFSET, TYPE, THIS, PTR, WIDTH, MASK, SIZE, SIZEOF, LENGTH, LENGTHOF, ST, SHORT, TYPE, OPATTR, MOD, NEAR, FAR, OR, XOR, EQ, NE, LT, LE, GT, GE, SHR, SHL и др.);
* названия команд (КОП) ассемблера, префиксов.

***Идентификаторы*** — последовательности допустимых символов, которые используются для обозначения имен переменных и меток.

Идентификатор может состоять из одного или нескольких символов (буквы латинского алфавита, цифры и некоторые специальные знаки — \_,?,$, @). Идентификатор не может начинаться символом цифры. Длина идентификатора может составлять до 255 символов (247 в MASM), хотя транслятор воспринимает лишь первые 32, а другие игнорирует. Регулировать длину возможных идентификаторов можно с использованием ключа командной строки /mv. Кроме того, существует возможность указать транслятору на необходимость различать прописные и строчные буквы или игнорировать их отличие. Для этого (у TASM) применяются ключи командной строки /mu, /ml, /mx

***Цепочки символов*** — это последовательности символов, заключенные в одинарные или двойные кавычки.

***Целые числа*** могут указываться в двоичной, десятичной или шестнадцатилетней системах счисления. Отождествление чисел при записи их в программах на ассемблере проводится по определенным правилам. Десятичные числа не требуют для своего отождествления указания каких-либо дополнительных символов

Важно отметить наличие символов (h) и перед (0) записью шестнадцатилетнего числа. Это сделано для того, чтобы транслятор мог отличить в программе одинаковые по форме записи десятичные и шестнадцатилетние числа. Например числа 1578 и 1578h выглядят одинаково, но имеют разные значения. Для того, чтобы однозначно описать в тексте программы на ассемблере шестнадцатилетнее число, что начинается с буквы, его дополняют ведущим нулем «0» и в конце ставят символ «h».

Для двоичных чисел—после записи нулей и единиц необходимо поставить букву «b». 10010101b

***Комментарий*** - это самый простой элемент предложения ассемблера. Любая комбинация символов ASCII, расположенная в строке за символом точки с запятой (;), транслятором игнорируется, то есть является комментарием.

**2. Операнды**

Операнды — это объекты, над которыми или с помощью которых выполняются действия, которые задаются инструкциями или директивами. Машинные команды могут либо совсем не иметь операндов, либо имеют один или два операнда. Большинство команд требуют двух операндов, один из которых является источником, а другой — приемником (операндом назначения). В двухоперандной машинной команде возможные следующие сочетания операндов:

* регистр — регистр;
* регистр — память;
* память — регистр;
* непосредственный операнд — регистр;
* непосредственный операнд — память.

Один операнд может располагаться в регистре или памяти, а второй операнд обязательно должен находиться в регистре или непосредственно в команде. Непосредственный операнд может быть только источником. Для ***правил сочетания типов операндов есть исключения***, которые касаются:

* команд работы с цепочками, которые могут перемещать данные из памяти в память;
* команд работы со стеком, которые могут переносить данные из памяти в стек, что также находится в памяти;
* команд типа умножения, которые, кроме операнда, указанного в команде, неявно используют еще и второй операнд.

Операндами могут быть числа, регистры, ячейки памяти, символьные идентификаторы. При необходимости для расчета некоторого значения или определения ячейки памяти, на которую будет влиять данная команда или директива, используются выражения, то есть комбинации чисел, регистров, ячеек памяти, идентификаторов с арифметическими, логическими, побитовыми и атрибутивными операторами.

**3. Типы данных ассемблера**

С точки зрения размерности процессор аппаратно поддерживает следующие типы данных:

* ***Байт*** — восемь последовательно расположенных битов, пронумерованных от 0 до 7, при этом бит 0 является самым младшим значимым битом.
* ***Слово*** — последовательность из двух байтов, которые имеют последовательные адреса. Размер слова — 16 битов; биты в слове нумеруются от 0 до 15. Байт, который содержит нулевой бит, называется младшим байтом, а байт, который содержит 15-й бит, — старшим. Процессоры Intel имеют важную особенность — младший байт всегда сохраняется по меньшему адресу. Адресом слова считается адрес его младшего байта. Адрес старшего байта может быть использован для доступа к старшей половине слова.
* ***Двойное слово*** — последовательность из четырех байтов (32 биты), расположенных по последовательным адресам. Нумерация этих битов проводится от 0 до 31. Слово, содержащее нулевой бит, называется младшим словом, а слово, содержащее 31-й бит, — старшим словом. Адресом двойного слова считается адрес его младшего слова. Адрес старшего слова может быть использован для доступа к старшей половине двойного слова.
* ***Учетверенное слово*** — из восьми байтов (64 биты), расположенных по последовательным адресам. Нумерация битов проводится от 0 до 63. Двойное слово, содержащее нулевой бит, называется младшим двойным словом, а двойное слово, содержащее 63-й бит, — старшим двойным словом. Адрес старшего двойного слова может быть использован для доступа к старшей половине учетверенного слова.
* ***128-битовый упакованный тип данных***. Этот тип данных появился в процессоре Pentium III. Для работы с ним в процессор введенные специальные команды.

Кроме трактовки типов данных с точки зрения их разрядности, процессор на уровне команд поддерживает логическуюинтерпретацию этих типов.

***Целый тип со знаком*** — двоичное значение со знаком размером 8,16 или 32 бита. Знак в этом двоичном числе содержится в 7, 15 или 31 бите соответственно. Нуль в этих битах в операндах отвечает положительному числу, а единица — отрицательному.

Числовые диапазоны (ЧД) для этого типа данных следующие:

* 8-разрядное целое — от -128 до +127;
* 16-разрядное целое — от -32 768 до +32 767;
* 32-разрядное целое — от -231 до +231 – 1 (-2147483648 до +2147483647).

***Целый тип без знака*** — двоичное значение без знака размером 8,16 или 32 бита. ЧД:

* байт — от 0 до 255;
* слово — от 0 до 65 535;
* двойное слово— от 0 до 232 – 1 (4 294 967 295).

***Указатель на память*** бывает двух типов:

* ***ближний тип*** — 32-разрядный логический адрес, который является относительным смещением в байтах от начала сегмента;
* ***дальний тип*** — 48-разрядный логический адрес, который состоит из двух частей: 16-разрядной сегментной части (селектора) и 32-разрядного сдвига.

***Цепочка*** является некоторым непрерывным набором байтов, слов или двойных слов максимальной длиной до 4 Гбайт.

Описанные данные можно определить как данные простого типа. Описать их можно с помощью специального вида директив — ***директив резервирования и инициализации данных***. Они являются указаниями транслятору на выделение определенного объема памяти. Если проводить аналогию с языками высокого уровня, то они являются определениями переменных. Любой переменной, объявленной с их помощью, ассемблер присваивает три атрибута:

* сегмент (seg) — адрес начала сегмента, что содержит переменную;
* сдвиг (offset) в байтах от начала сегмента с переменной;
* тип (type) — объем памяти, что выделяется переменной в соответствии с директивой объявления переменной.

**Контрольные вопросы**

1. Что собой представляет программа на ассемблере?
2. Какие синтаксические конструкции используются в ассемблере? Поясните каждую из перечисленных конструкций.
3. Что включает алфавит ассемблера?
4. Что такое операнд?
5. Какие сочетания операндов возможны в двухоперандной машинной команде?
6. Какие типы данных поддерживает ассемблер?