Структура программы на ассемблере

Программа представляет собой последовательность операторов. Каждый *оператор* записывается на отдельной строке.
Операторы бывают четырех типов:

- *команды или инструкции.* Представляют собой символические аналоги машинных команд. В процессе трансляции инструкции ассемблера преобразуются в соответствующие коды системы команд микропроцессора;
- *макрокоманды*. Оформленные определенным образом предложения текста программы, замещаемые во время трансляции другими предложениями;
- *директивы*. Указания транслятору ассемблера на выполнение некоторых действий. У директив нет аналогов в машинном представлении;
- *строки комментариев.* Могут содержать любые символы, в том числе и буквы русского алфавита. Комментарии игнорируются транслятором. Начинаются с символа точка с запятой. Могут располагаться в конце рператорной строки или на отдельной строке.

Машинная команда делится на группы бит, называемые полями. Основным полем, которое всегда присутствует в команде, является КОД ОПЕРАЦИИ (КОП). КОП определяет что делать. Необязательными полями являются поля операндов. ***Операнды*** — это объекты, над которыми производятся действия. Операнды могут содержать данные, часть адреса, косвенный указатель.

Машинные команды бывают трех видов:

1. безоперандные КОП;
2. однооперандные КОП операнд;
3. двухоперандные КОП операнд, операнд.

Сочетание операндов в команде:
Регистр – регистр
Регистр – память
Регистр – непосредственный операнд
Память – регистр
Память – непосредственный операнд

Допустимыми символами при написании текста программ являются:

1. все латинские буквы: **A—Z**, **a—z**. При этом заглавные и строчные буквы считаются эквивалентными;
2. цифры от **0** до **9**;
3. знаки **?**, **@**, **$**, **_**, **&**;
4. разделители **, . [ ] ( )** < > **{ }** + / * **%** **!** ' " **?** \ = # ^.

(не используются ~ ` |)

Идентификаторы — последовательности допустимых символов, использующиеся для обозначения кодов операций, имен переменных и названий меток. Идентификатор может состоять из одного или нескольких символов. В качестве символов можно использовать буквы латинского алфавита, цифры и некоторые специальные знаки — _, ?, $, @. Идентификатор не может начинаться символом цифры. Длина идентификатора может быть до 255 символов, хотя транслятор воспринимает лишь первые 32, а остальные игнорирует. Регулировать длину возможных идентификаторов можно с использованием опции командной строки **mv**. Кроме этого существует возможность указать транслятору на то, чтобы он различал прописные и строчные буквы либо игнорировал их различие (что

и делается по умолчанию). Для этого применяются опции командной строки **/mu, /ml, /mx**;

Строки символов заключаются в одинарные или двойные кавычки;

Целые числа в двоичной, десятичной или шестнадцатеричной системах счисления.

- **Десятичные числа** не требуют для своего отождествления указания каких-либо дополнительных символов, например 25 или 139.
- Для **двоичных чисел** необходимо после записи нулей и единиц, входящих в их состав, поставить латинское "**b**", например 10010101**b**.
- Для **шестнадцатеричных чисел** на конце последовательности шестнадцатеричных цифр записывают латинскую букву "**h**". Если шестнадцатеричное число начинается с буквы, то перед ним записывается ведущий ноль: **0**ef15**h**.

Формат директив

[имя] директива [операнд1 , ... , операнд n] [; комментарий]

Формат команд

[имя метки :] команда [операнд1 [, операнд 2]] [; комментарий]

Типы данных

При программировании на языке ассемблера используются данные следующих типов:

1. ***Непосредственные*** данные, представляющие собой числовые или символьные значения, являющиеся частью команды.
2. Данные ***простого*** типа. Описываются с помощью директив резервирования памяти.

Эти два типа данных являются *элементарными*, или *базовыми.*

3. Данные ***сложного типа.*** Строятся на основе базовых типов. Введение сложных типов данных позволяет несколько сгладить различия между языками высокого уровня и ассемблером.

Микропроцессор аппаратно поддерживает следующие основные типы данных

- ***байт*** — восемь последовательно расположенных битов, пронумерованных от 0 до 7, при этом бит 0 является самым младшим значащим битом;
- ***слово*** — последовательность из двух байт, имеющих последовательные адреса. Размер слова — 16 бит; биты в слове нумеруются от 0 до 15. Байт, содержащий нулевой бит, называется младшим байтом, а байт, содержащий 15-й бит - старшим байтом. Микропроцессоры Intel имеют важную особенность — ***младший байт всегда хранится 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 байт (32 бита), расположенных по последовательным адресам. Нумерация этих бит производится от 0 до 31. Слово, содержащее нулевой бит, называется младшим словом, а слово, содержащее 31-й бит, - старшим словом. Младшее слово хранится по меньшему адресу.

- ***учетверенное слово*** — последовательность из восьми байт (64 бита), расположенных по последовательным адресам. Нумерация бит производится от 0 до 63. Двойное слово, содержащее нулевой бит, называется младшим двойным словом, а двойное слово, содержащее 63-й бит, — старшим двойным словом. Младшее двойное слово хранится по меньшему адресу.

Микропроцессор на уровне команд поддерживает ***логическую интерпретацию следущих*** 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-м бите соответственно. Ноль в этих битах в операндах соответствует положительному числу, а единица — отрицательному. Отрицательные числа представляются в дополнительном коде.
- ***Целый тип без знака — двоичное значение без знака, размером 8, 16 или 32 бита***.
- ***Указатель на память двух типов:***
  - **ближнего типа** — 32-разрядный логический адрес, представляющий собой относительное смещение в байтах от начала сегмента. Эти указатели могут также использоваться в сплошной (плоской) модели памяти, где сегментные составляющие одинаковы;
  - **дальнего типа** — 48-разрядный логический адрес, состоящий из двух частей: 16-разрядной сегментной части — селектора, и 32-разрядного смещения.
- ***Цепочка*** — представляющая собой некоторый непрерывный набор байтов, слов или двойных слов максимальной длины до 4 Гбайт.
- ***Битовое поле*** представляет собой непрерывную последовательность бит, в которой каждый бит является независимым и может рассматриваться как отдельная переменная. Битовое поле может начинаться с любого бита любого байта и содержать до 32 бит.
- ***Неупакованный двоично-десятичный тип*** — байтовое представление десятичной цифры от 0 до 9. Неупакованные десятичные числа хранятся как байтовые значения без знака по одной цифре в каждом байте. Значение цифры определяется младшим полубайтом.
- ***Упакованный двоично-десятичный тип*** представляет собой упакованное представление двух десятичных цифр от 0 до 9 в одном байте. Каждая цифра хранится в своем полубайте. Цифра в старшем полубайте (биты 4–7) является старшей.

Директивы резервирования и инициализации

Эти директивы указывают транслятору выделить определенный объем памяти. Если проводить аналогию с языками высокого уровня, то эти директивы определениями переменных. Транслятор, обрабатывая каждую такую директиву, выделяет необходимое количество байт памяти и при необходимости инициализирует эту область некоторым значением.

Директивы резервирования и инициализации данных простых типов имеют формат:

[имя] ДИРЕКТИВА выражение

ДИРЕКТИВА может быть

- **db** — резервирование памяти для данных размером 1 байт.
  Можно задавать
  1. значения: из диапазона:
     - для чисел со знаком –128...+127;
     - для чисел без знака 0...255;

2. символьную строку из одного или более символов. Строка заключается в кавычки. В этом случае определяется столько байт, сколько символов в строке.

- **dw** — резервирование памяти для данных размером 2 байта.
  Можно задавать
  1. значения: из диапазона:
     - для чисел со знаком –32 768...32 767;
     - для чисел без знака 0...65 535;
  2. смещение в 16-битовом сегменте или адрес сегмента;
- **dd** — резервирование памяти для данных размером 4 байта.
  Можно задавать
  1. значения из диапазона:
     - для чисел со знаком –2 147 483 648...+2 147 483 647;
     - для чисел без знака 0...4 294 967 295;
  2. адресное выражение, состоящее из 16-битового адреса сегмента и 16-битового смещения;
- **df** — резервирование памяти для данных размером 6 байт;
- **dp** — резервирование памяти для данных размером 6 байт.
  Директивами **df** и **dp** можно задавать значения:
  1. адресное выражение, состоящее из 16-битового сегмента и 32-битового смещения;
  2. константу со знаком из диапазона $-2^{47}...2^{47}-1$;
  3. константу без знака из диапазона $0...2^{48}-1$;
- **dq** — резервирование памяти для данных размером 8 байт.
  Можно задавать значения:
  1. константу со знаком из диапазона $-2^{63}...2^{63}-1$;
  2. константу без знака из диапазона $0...2^{64}-1$;
- **dt** — резервирование памяти для данных размером 10 байт.
  Можно задавать значения:
  1. константу со знаком из диапазона $-2^{79}...2^{79}-1$;
  2. константу без знака из диапазона $0...2^{80}-1$;
  3. строку длиной до 10 байт, заключенную в кавычки;
  4. упакованную десятичную константу в диапазоне 0...99 999 999 999 999 999 999.

В поле выражение может быть

- **?** показывает, что содержимое поля не определено, то есть при задании директивы с таким значением выражения содержимое выделенного участка физической памяти изменяться не будет. Фактически, создается неинициализированная переменная;
- **значение инициализации** — значение элемента данных, которое будет занесено в память после загрузки программы. Фактически, создается инициализированная переменная, в качестве которой могут выступать константы и строки символов.
- **имя** — некоторое символическое имя метки или ячейки памяти в сегменте данных, используемое в программе.
- **Количество повторений DUP (выражение)**

Примеры

DW 10 ; определить слово со значением 10
DD ? ; определить двойное слово с произвольным значением
DB ‘?’ ; определить байт со значением ASCII- кода вопроса
DB ‘8’ ; определить байт со значением ASCII- кода символа 8
DB 1 ; определить байт со значением 1
DB 1

DB 1

Следующие друг за другом операторы одной размерности можно записать в одну строку, разделив значения запятыми:
DB ‘?’, ‘8’, 1, 1, 1
Обозначения ASCII-символов в директивах DB можно сгруппировать, задав их одной строкой в кавычках:
DB ‘?8’, 1, 1, 1
Повторяющиеся элементы группируются при помощи конструкции DUP
DB ‘?8’, 3 DUP (1)
Конструкции DUP допускают вложенность.
Например,
DB 2 DUP (1, 3 DUP (0))
Означает
DB 1,0,0,0,1,0,0,0

Микропроцессоры Intel требуют следования данных в памяти по принципу: **младший байт по младшему адресу**. Это должно учитываться при определении данных.
Например, слово со значением 256 можно определить тремя способами:

1. **DW 256**
2. **DW 100h**
3. **DB 0,1**