ASSEMBLER

Параметры директивы SEGMENT

Выравнивание

■ BYTE

■ WORD

■ DWORD

■ PARA

■ PAGE

Тип

■ PUBLIC

■ STACK

■ COMMON

■ AT

■ PRIVATE

Класс - любая метка, взятая в одинарные кавычки. Сегменты одного класса расположатся в памяти друг за другом.

***Атрибут выравнивания сегмента*** (тип выравнивания) align сообщает компоновщику о том, что нужно обеспечить размещение начала сегмента на заданной границе. Это важно, поскольку при правильном выравнивании доступ к данным в процессорах, совместимых с базовым i8086, выполняется быстрее. Допустимые значения этого атрибута следующие:

* BYTE — выравнивание не выполняется. Сегмент может начинаться с любого адреса памяти;
* WORD — сегмент начинается по адресу, кратному двум, то есть последний (младший) значащий бит физического адреса равен 0 (выравнивание на границу слова);
* DWORD — сегмент начинается по адресу, кратному четырем, то есть два последних (младших) значащих бита равны 0 (выравнивание по границе двойного слова);
* PARA — сегмент начинается по адресу, кратному 16, то есть последняя шестнадцатеричная цифра адреса должна быть 0h (выравнивание по границе параграфа);
* PAGE — сегмент начинается по адресу, кратному 256, то есть две последние шестнадцатеричные цифры должны быть 00h (выравнивание по границе страницы размером 256 байт);
* MEMPAGE — сегмент начинается по адресу, кратному 4 Кбайт, то есть три последние шестнадцатеричные цифры должны быть 000h (адрес следующей страницы памяти размером 4 Кбайт);

По умолчанию тип выравнивания имеет значение PARA.

***Атрибут комбинирования сегментов*** (комбинаторный тип) combine сообщает компоновщику, как нужно комбинировать сегменты различных модулей, имеющие одно и то же имя. По умолчанию атрибут комбинирования принимает значение PRIVATE. Значениями атрибута комбинирования сегмента могут быть:

* PRIVATE — сегмент не будет объединяться с другими сегментами с тем же именем вне данного модуля;
* PUBLIC — заставляет компоновщик соединить все сегменты с одинаковым именем. Новый объединенный сегмент будет целым и непрерывным. Все адреса (смещения) объектов, а это могут быть, в зависимости от типа сегмента, команды или данные, будут вычисляться относительно начала этого нового сегмента;
* COMMON — располагает все сегменты с одним и тем же именем по одному адресу. Все сегменты с данным именем будут перекрываться и совместно использовать память. Размер полученного в результате сегмента будет равен размеру самого большого сегмента;
* AT xxxx — располагает сегмент по абсолютному адресу параграфа (параграф — объем памяти, кратный 16, поэтому последняя шестнадцатеричная цифра адреса параграфа равна 0). Абсолютный адрес параграфа задается выражением хххx. Компоновщик располагает сегмент по заданному адресу памяти (это можно использовать, например, для доступа к видеопамяти или области ПЗУ), учитывая атрибут комбинирования. Физически это означает, что сегмент при загрузке в память будет расположен, начиная с этого абсолютного адреса параграфа, но для доступа к нему в соответствующий сегментный регистр должно быть загружено заданное в атрибуте значение. Все метки и адреса в определенном таким образом сегменте отсчитываются относительно заданного абсолютного адреса;
* STACK — определение сегмента стека. Заставляет компоновщик соединить все одноименные сегменты и вычислять адреса в этих сегментах относительно регистра SS. Комбинированный тип STACK (стек) аналогичен комбинированному типу PUBLIC, за исключением того, что регистр SS является стандартным сегментным регистром для сегментов стека. Регистр SP устанавливается на конец объединенного сегмента стека. Если не указано ни одного сегмента стека, компоновщик выдаст предупреждение, что стековый сегмент не найден. Если сегмент стека создан, а комбинированный тип STACK не используется, программист должен явно загрузить в регистр SS адрес сегмента (подобно тому, как это делается для регистра DS).

Директивы выделения памяти

■ Псевдокоманда - директива ассемблера, которая приводит к включению данных или кода в программу, но не соответствует никакой команде процессора.

■ Псевдокоманды определения данных указывают, что в соответствующем месте располагается переменная, резервируют под неё место заданного типа, заполняют значением и ставят в соответствие метку.

■ Виды: DB (1), DW (2), DD (4), DF (6), DQ (8), DT (10).

Таблица 4.1. Директивы выделения памяти

|  |  |  |
| --- | --- | --- |
| Код директивы | Размер данного (байт) | Тип соответствующего данного в языке С |
| DB | 1 | char |
| DW | 2 | short |
| DD | 4 | int, unsigned, float |
| DF, DP | 6 | - |
| DQ | 8 | double, int64 |
| DT | 10 | long double |

■ Примеры: – a DB 1 – float\_number DD 3.5e7 – text\_string DB 'Hello, world!'

■ DUP - заполнение повторяющимися данными.

■ ? - неинициализированное значение.

■ uninitialized DW 512 DUP(?)

**Инструкция LOOP** в Ассемблере уменьшает значение в регистре СХ в реальном режиме или ECX в защищённом. Если после этого значение в СХ не равно нулю, то команда LOOP выполняет переход на МЕТКУ. Синтаксис:

LOOP МЕТКА

Состояние [**флагов**](http://av-assembler.ru/asm/afd/asm-flags-register.htm) не изменяется.

МЕТКА - это допустимый в Ассемблере идентификатор.

**Алгоритм работы команды LOOP:**

* CX = CX - 1
* Если CX не равен 0, то выполнить [**переход**](http://av-assembler.ru/instructions/jb.php#uslovie)
* Иначе не выполнять переход, продолжить цикл

То есть команда LOOP выполняется в два этапа. Сначала из [**регистра**](http://av-assembler.ru/asm/afd/asm-cpu-registers.htm) СХ вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной МЕТКЕ. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды LOOP.

**Как выполнить цикл в Ассемблере**