МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное

обеспечение информационных технологий

**РЕФЕРАТ**

на тему:

«Модели памяти и их связь с модификаторами near, far, huge»

Выполнил:

студент I курса 6 группы

специальности ПОИТ Шимко А.А

Преподаватель:

Белодед Николай Иванович

**Оглавление**

Введение………………………………………………………………………...…3

Модификатор near, far, huge…………………………………………...………....5

**Введение**

**Память -** способность объекта обеспечивать хранение данных.  
Все объекты, над которыми выполняются команды, как и сами команды, хранятся в памяти компьютера.

Память состоит из ячеек, в каждой из которых содержится 1 ***бит*** информации, принимающий одно из двух значений: 0 или 1. Биты обрабатывают группами фиксированного размера. Для этого группы бит могут записываться и считываться за одну базовую операцию.  Группа из 8 бит называется.

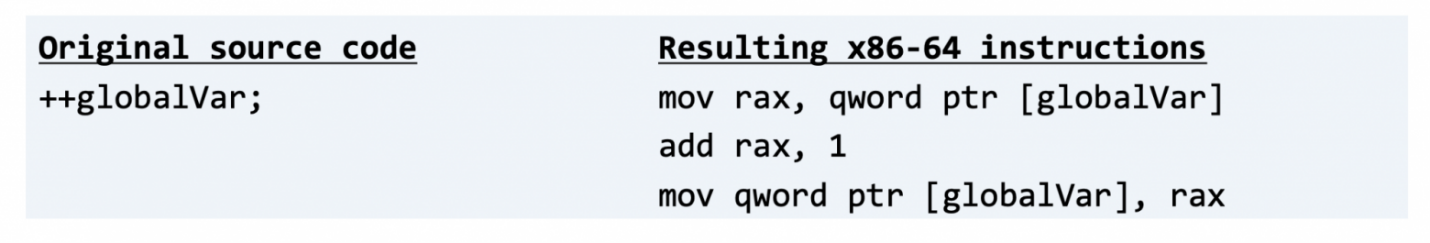


***Байты последовательно располагаются в памяти компьютера.***

* 1 килобайт (Кбайт) = 210 = 1 024 байт
* 1 мегабайт (Мбайт) = 210 Кбайт = 220 байт = 1 048 576 байт
* 1 гигабайт (Гбайт) = 210 Мбайт = 230 байт = 1 073 741 824 байт

Для доступа к памяти с целью записи или чтения отдельных элементов информации используются ***идентификаторы***, определяющие их расположение в памяти. Каждому идентификатору в соответствие ставится ***адрес***. В качестве адресов используются числа из диапазона от 0 до 2k-1  со значением k, достаточным для адресации всей памяти компьютера. Все 2k адресов составляют ***адресное пространство компьютера***.

Итак, начнем с трех фундаментальных концепций — атомарность, эксклюзивность и изменение порядка. Когда мы говорим, что операция атомарна, мы имеем в виду, что она не может быть прервана. Это означает, что во время выполнения операции не может произойти переключение потока, операция не может частично завершиться. На оборудовании с 64-битными процессорами Intel можно быть уверенными, что операции чтения и записи значений меньше 64 бит являются атомарными. Эти операции не могут быть прерваны и не могут частично завершиться. Проблема в том, что большинство на первый взгляд простых операций, которые мы пишем на языках высокого уровня, на самом деле не являются атомарными. Очевидный пример, с которым многие из вас наверняка знакомы — увеличение значения на единицу. На C++, компилятор сгенерирует код наподобие такого:



Это последовательность инструкций, выполняющая считывание из памяти, добавление единицы и запись обратно в память. Каждая операция из этой последовательности атомарная, а вся последовательность, очевидно, не является атомарной операцией. Важно понимать, что многие операции кажутся атомарными на языках высокого уровня, но на самом деле таковыми не являются. В этом можно убедиться, посмотрев на сгенерированный машинный код.

Применение *моделей памяти* позволяет контролировать ее сегментное распределение и делать его более эффективным или адекватным решаемой задаче. По умолчанию при компиляции и редактировании связей генерируется код для работы в малой (small) модели. Если программа удовлетворяет хотя бы одному из двух следующих условий, следует использовать другую модель памяти:

* размер кода программы превышает 64 Кб;
* размер статических данных программы превышает 64 Кб.
* имеется два варианта выбора модели памяти для программы:
* назначить нужную модель в опциях компилятора;
* использовать в объявлении объектов программы модификаторы near**,** far и huge.

Можно комбинировать эти способы.

Архитектура процессоров, основанных на базе 8086/8088, предусматривает разбиение оперативной памяти на физические *сегменты*, способные содержать информацию объемом до 64 Кб. Минимальное количество сегментов, выделяемое программе, равно двум: сегмент кода и сегмент статических данных. К статическим данным при этом относятся все объекты, объявленные с классом памяти extern или static. Формальные параметры функций и локальные переменные не являются статическими. Они хранятся не в сегменте данных, а в *стеке* (однако при этом стек может быть совмещен со стандартным сегментом данных физически).

Программа на Си может работать с динамической памятью с помощью библиотечных функций семейства malloc. При этом память может выделяться как в отдельном сегменте (*дальняя* динамическая память), так и в стандартном сегменте данных между концом занятой данными области и стеком (*ближняя* динамическая память).

Адрес оперативной памяти состоит из двух частей:

· *базовый адрес сегмента* — 16‑битовое число;

· *смещение* относительно начала сегмента — также 16-битовое число.

**Модификаторы near, far, huge**

Эти модификаторы оказывают воздействие на работу с адресами объектов.

Компилятор языка Си позволяет использовать при компиляции одну из нескольких моделей памяти. Виды моделей памяти и методы их применения рассмотрены в разделе 8 "Модели памяти".

Модель, которую вы используете, определяет размещение в оперативной памяти вашей программы и данных, а также внутренний формат указателей. Однако при использовании какой-либо модели памяти можно объявить указатель с форматом, отличным от действующего по умолчанию. Это делается с помощью модификаторов **near**, **far** и **huge**.

Указатель типа **near** — 16-битовый; для определения адреса объекта он использует смещение относительно текущего содержимого сегментного регистра. Для указателя типа **near** доступная память ограничена размером текущего 64-килобайтного сегмента данных.

Например:

*double near cdecl calc(double, double);*

*double cdecl near calc(double, double);*

В примере 3 показано два эквивалентных объявления. В них объявляется **calc** как функция с модификаторами near и **cdecl**.

Указатель типа **far** — 32-битовый; он содержит как адрес сегмента, так и смещение. При использовании указателей типа **far** допустимы обращения к памяти в пределах 1-мегабайтного адресного пространства процессора Intel 8086/8088, однако значение указателя типа **far** циклически изменяется в пределах одного 64-килобайтного сегмента.

Например:

char far pascal initlist[INITSIZE];

char far nextchar, far \*prevchar, far \*currentchar;

В примере специальное ключевое слово *far* модифицирует расположенную справа от него звездочку, делая х указателем на **far** указатель на значение типа **char**. Это объявление можно для ясности записать и так:

Указатель типа **huge** — 32-битовый; он также содержит адрес сегмента и смещение. Значение указателя типа **huge** может быть изменено в пределах всего 1-мегабайтного адресного пространства. В СП ТС указатель типа huge всегда хранится в нормализованном формате. Это имеет следующие следствия:

—операции отношения ==, !=, <, >, <=, >= выполняются корректно и предсказуемо над указателями типа **huge**, но не над указателями типа **far**;

—при использовании указателей типа **huge** требуется дополнительное время, т. к. программы нормализации должны вызываться при выполнении любой арифметической операции над этими указателями. Объем кода программы также возрастает.

В СП MSC модификатор **huge** применяется только к массивам, размер которых превышает 64 К. В СП ТС недопустимы массивы больше 64 К, а модификатор **huge** применяется к функциям и указателям для спецификации того, что адрес функции или указываемого объекта имеет тип **huge**.

Для вызова функции типа **near** используются машинные инструкции ближнего вызова, для типов **far** и **huge** — дальнего.

Например:

int huge database [65000];

В этом примере объявляется массив с именем *database,* содержащий 65000 элементов типа **int**. Поскольку размер массива превышает 64 Кбайта, его описатель должен быть модифицирован специальным ключевым словом **huge.**

**Замечание:** Модификатор **interrupt** предназначен для объявления функций, работающих с векторами прерываний процессора 8086/8088. Для функции типа **interrupt** при компиляции генерируется дополнительный код в точке входа и выхода из функции, для сохранения и восстановления регистров микропроцессора АХ, ВХ, СХ, DX, SI, DI, ES и DS. Остальные регистры — ВР, SP, SS, CS и IP сохраняются всегда как часть вызывающей последовательности языка Си или часть самой системы обработки прерывания.

Функции прерываний следует объявлять с типом возвращаемого значения **void**.

Функции прерываний поддерживаются для всех моделей памяти. В СП MSC, в малой и средней модели в регистр DS заносится при входе в функцию адрес сегмента данных всей программы, а в компактной, большой и максимальной модели в регистр DS заносится адрес сегмента данных программного модуля. В СП ТС только в максимальной модели в регистр DS заносится адрес сегмента данных программного модуля, а в остальных моделях—адрес сегмента данных всей программы.

Модификатор **interrupt** не может использоваться совместно с модификаторами **near**, **far**, **huge**.