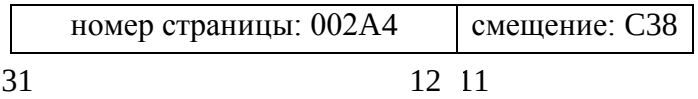
7. Страничная организация памяти

Страничная организация памяти основана на том, что физическое и виртуальное адресные пространства разбиваются на **фрагменты постоянной длины.** Размер страницы выбирается **кратным степени 2,** т.к. это позволяет максимально просто выполнять преобразование ВА в ФА на аппаратном уровне. Наиболее распространенный размер страницы равен 2^{12} = 4096 байт. Все страницы нумеруются от 0 до N. Тем самым ВАП состоит из $2^{32} / 2^{12} = 2^{20}$ = 1 Мбайт страниц, а число реальных физических страниц зависит от размера установленной памяти.

Так, для объема основной памяти 128 Мб получим 32 Кбайт страниц. Можно сказать, что страница – это просто некоторый диапазон адресов: страница 0 определяет адреса от 0 до 4095 (или 0FFFh), страница 1 – от 4096 до 8191 (или от 1000h до 1FFFh), страница 2 – от 8192 до 12287 (от 2000h до 2FFFh) и т.д. Видно, что номер страницы однозначно определяет соответствующий этой странице диапазон адресов.

При создании транслятором исполняемого машинного кода назначение виртуальных адресов командам и элементам данных начинается с нулевой страницы и идет в порядке следования страниц. Каждый виртуальный адрес задается **номером** страницы N_V и **смещением** S_V относительно начала страницы: ВА = (N_V , S_V).

Если размер страницы равен 4096 байт, то смещение для любой страницы будет принимать значения от 0 до 4095. Тем самым полный 32-х разрядный виртуальный адрес разбивается на две составляющие: младшие 12 разрядов (т.е. 3 полубайта) определяют смещение на странице, а старшие 20 разрядов (5 полубайтов) определяют номер страницы. Например, виртуальный адрес 002A4C38 разбивается следующим образом



Поскольку ВАП и ФАП разбиты на страницы совершенно одинаково, замена ВА на ФА сводится только к **замене номера страницы,** смещение при этом остается **неизменным!** Это является важнейшим преимуществом страничной организации и одной из основных причин столь высокой популярности этого способа организации памяти в современных ОС. Замена номера страницы в виртуальном адресе сводится к действиям с отдельными битами и чрезвычайно эффективно выполняется на **аппаратном** уровне.

Такой подход позволяет отображать **любую** виртуальную страницу на **любую** физическую. Это приводит к исключительно эффективному использованию основной памяти и эффективной реализации механизма виртуальной памяти. Для того чтобы процессор мог выполнить преобразование адресов, ему необходима информация о соответствии виртуальных и физических страниц: в какую физическую страницу отображена данная виртуальная страница. Для этого система для каждого процесса строит специальную **таблицу страниц.** Каждая запись в этой таблице называется **дескриптором** страницы и обычно содержит следующие данные:

- номер назначенной физической страницы;
- признак присутствия данной страницы в основной памяти;
- признак модификации страницы, отслеживающий наличие изменений в данной странице за время ее присутствия в основной памяти;

- признак обращения к странице, необходимый для отслеживания частоты обращений к странице;
- код доступа к странице для организации механизма защиты.

Входом в таблицу страниц является номер виртуальной страницы, содержащийся в виртуальном адресе. Таблица строится при запуске программы, т.е. в момент создания процесса, и является одной из важнейших структур, связанных с процессом. Если процесс реализует внутри себя дополнительные потоки, то все они используют **одну и ту же** таблицу страниц, разделяя тем самым одно и то же адресное пространство.

Поскольку для каждого процесса создается **своя** таблица страниц, появляется возможность организовать **защиту** адресных пространств процессов. Процесс может манипулировать только своими страницами. Если нескольким процессам надо использовать общие данные, то для этого система создает **общие** или **разделяемые** страницы. Начальный адрес размещения таблицы страниц активного процесса содержится в специальном системном регистре процессора, недоступном для прикладных программ. При переключении процессов система заменяет содержимое этого регистра, тем самым включая в работу новую таблицу страниц.

Если при запуске процесса в основной памяти нет достаточного количества свободных страниц, чтобы разместить весь код и данные, то загружается только **часть** страниц, остальные сохраняются на диске и в таблице страниц помечаются как **отсутствующие.** При необходимости любая отсутствующая страница может быть загружена в память с назначением ей свободной физической страницы. Наоборот, система может вытеснить на диск часть или даже все страницы процесса. Такой обмен страницами между основной и дисковой памятью называют **страничным обменом,** или **своппингом** (swap – обмен). Часто перемещаемые на диск страницы сохраняются в специальном файле, называемом **страничным файлом,** или файлом **подкачки.** Каждая такая операция сопровождается соответствующими изменениями в таблице страниц.

Алгоритм преобразования ВА в ФА можно представить следующим образом. Пусть при выполнении некоторой команды процессору необходимо обратиться к памяти по виртуальному адресу. Тогда система, используя реализованную в процессоре аппаратную поддержку, выполняет следующее:

- из виртуального адреса выделяется номер виртуальной страницы;
- с помощью системного регистра производится обращение к текущей таблице страниц;
- из таблицы по номеру виртуальной страницы выбирается запись-дескриптор;
- анализируется признак присутствия данной страницы в памяти;
- если страница отображена в память, то из дескриптора извлекается номер назначенной физической страницы, который заменяет номер виртуальной страницы, и тем самым формируется реальный физический адрес;
- если страницы в памяти нет, то генерируется так называемое **страничное прерывание,** обработчик которого должен найти на диске и загрузить в память необходимую страницу;
- на время выполнения этих действий процессор может переключиться на обработку другого потока;
- если при загрузке нужной виртуальной страницы для нее есть свободная физическая страница, то загрузка выполняется, в таблицу страниц заносится соответствующий номер, устанавливается признак присутствия страницы и вычисляется физический адрес;
- если ни одной свободной страницы нет, то система вынуждена:
    - по определенному правилу найти страницу, которую надо вытеснить на диск (правила описаны ниже);
    - проверить ее признак модификации и либо сохранить страницу на диске, либо просто отметить ее как отсутствующую;

- освободившуюся физическую страницу назначить затребованной виртуальной странице.

Для определения вытесняемой на диск страницы существуют разные правила. Наиболее простым с точки зрения реализации является правило **случайного** выбора. Более сложными, но и более эффективными являются следующие правила:

- выбрать страницу, к которой было меньше всего обращений;
- выбрать страницу, которая не использовалась дольше всех.

Для правильной работы страничного механизма система должна вести учет свободных и занятых физических страниц, отмечая каждую соответствующим признаком.

При всех своих достоинствах страничная организация памяти имеет ряд следующих недостатков:

- большие накладные расходы на поддержку таблиц страниц: одна таблица может требовать нескольких мегабайт памяти для своего хранения;
- разбиение кода и данных на виртуальные страницы носит **механический** характер и никак не учитывает **логическую** структуру построения программы; это может приводить, например, к тому, что начало кода некоторой подпрограммы будет находиться на одной странице, а конец – на другой, что приведет к лишним страничным прерываниям при выполнении единой подпрограммы.