6. Задача управления основной памятью

Основная память является одним из наиболее важных ресурсов вычислительной системы. В соответствии с фундаментальными принципами, сформулированными Джоном фон Нейманом более 60 лет назад, выполняемые процессором команды и обрабатываемые данные должны размещаться в основной памяти, занимая выделенные им области. Каждая команда и каждый элемент данных располагаются в памяти на строго фиксированных местах, описываемых в машинных командах в виде адресов.

Назначение адресов командам и данным выполняется при **трансляции** программы. При разработке программы вместо адресов используются символические **имена,** замена которых адресами является одной из основных задач транслирующих программ (к ним можно отнести и программы-ассемблеры, переводящие символический код на языке ассемблера в машинный код). После трансляции адресные взаимосвязи между командами и данными становятся весьма жесткими. Многие команды в качестве своих операндов содержат адреса обрабатываемых элементов данных или адреса других команд. При этом, как правило, важным является **не абсолютное** расположение команд и данных в памяти, а их размещение **относительно** друг друга. Отсюда возникают две разновидности исполняемого кода, формируемого трансляторами: **абсолютный** и **перемещаемый** код.

Код в абсолютном формате жестко привязан к его будущему расположению в основной памяти. Такой код вместе со своими данными может размещаться в памяти, только начиная со строго заданного адреса. Данный адрес называют **адресом загрузки,** и меняться он для однажды созданного кода не может. В этом случае все команды и элементы данных имеют четко определенные неизменяемые адреса, раз и навсегда (до следующей перетрансляции) прописанные в операндных частях машинных команд. Большим достоинством такого кода является высокая скорость его загрузки в память и выполнения процессором, а большим недостатком – невозможность изменения адреса загрузки. Поэтому данная разновидность

применяется только в специальных целях, например – при создании важнейших системных модулей, размещение которых в памяти всегда неизменно и выполняется только один раз при загрузке системы.

Код в перемещаемом формате, как следует из его названия, может размещаться в памяти, начиная с **любого** адреса. В таком коде важнейшим понятием является расположение команды или элемента данных относительно начального нулевого адреса – так называемое **смещение.** Именно использование смещений вместо абсолютных адресов в операндах машинных команд позволяет размещать код в **любом** месте памяти, поскольку величина смещения **НЕ** изменяется при изменении адреса загрузки. Платой за такую гибкость является необходимость выполнения дополнительной работы при загрузке, связанной с настройкой адресных операндов на текущий адрес загрузки. Поскольку создаваемые при трансляции адреса **не соответствуют** реальному размещению кода в памяти, их можно назвать **виртуальными.** Именно этот способ является основным в настоящее время при создании подавляющего числа программ, особенно прикладных.

Преобразование виртуальных адресов (ВА) в физические адреса (ФА) может выполняться в **разные моменты времени:** либо при **загрузке** кода в память, либо непосредственно при **выполнении** команд процессором. Первый подход используется, если система не поддерживает механизм виртуальной памяти. В этом случае специальная системная программа “**перемещающий загрузчик**” размещает запускаемый код в памяти по заданному текущему адресу загрузки и заменяет в командах смещения реальными физическими адресами. После такой настройки размещенный в памяти код может выполняться процессором, поскольку все связи между командами и данными будут иметь свои “законные” значения. Это приводит к более медленной загрузке, но зато выполнение команд происходит также быстро, как и для абсолютного кода.

К сожалению, данный метод плохо подходит для многозадачных систем. Это связано с тем, что современные массовые приложения требуют больших объемов памяти и за этими запросами не могут угнаться даже постоянно возрастающие доступные объемы основной памяти. Поэтому одновременное размещение в памяти кода и данных сразу нескольких таких “монстров” (в числе которых и сами операционные системы), как правило, невозможно. Часть кода и данных запущенных программ (как прикладных, так и системных) **хранится на диске,** загружаясь в основную память лишь по необходимости. Поскольку пользователь в условиях “вседозволенной” многозадачности может переключаться между работающими приложениями в любом нужном ему порядке, за время существования процесса некоторые части его кода и данных могут **многократно** загружаться в память с диска и перемещаться обратно на диск.

Самое интересное в том, что повторная загрузка кода с диска в память может выполняться в **совершенно разные** области! Этот механизм, известный как **виртуальная память**, требует хранения кода на диске в своем “родном” виртуальном виде, т.е. с использованием виртуальных адресов. Преобразование ВА в ФА в этом случае выполняется в момент непосредственного обращения процессора по виртуальному адресу, т.е. при выполнении потока команд. Ясно, что для максимально быстрого выполнения кода такие преобразования должны выполняться как можно быстрее, и без **аппаратной поддержки** здесь обойтись нельзя. Именно поэтому все основные современные процессоры в своей архитектуре так или иначе поддерживают механизм преобразования адресов.

Использование виртуальных адресов и виртуальной памяти является одной из основ построения современных многозадачных (многопоточных) ОС. Концепция виртуальных адресов в некотором смысле упрощает работу трансляторов, т.к. уже не надо “думать” о проблеме нехватки памяти. Решение этой проблемы переходит на уровень ОС. Транслятор создает машинный код, используя **весь** возможный диапазон изменения виртуальных

адресов, а какая реальная физическая память будет установлена на компьютере, где данный код будет выполняться, транслятор “не волнует”.

В связи с этим вводится понятие **виртуального адресного пространства** (ВАП) как **потенциально** возможного диапазона изменения адресов. Размерность ВАП определяется архитектурой базового процессора (в первую очередь разрядностью шины адреса) и возможностями ОС. Так, все 32-х разрядные процессоры обеспечивают ВАП в объеме 2^{32} = 4 Гбайт, а процессоры Pentium IV – и того больше – 2^{36} = 64 Гбайт. Процессоры типа IA-64 обеспечивают уже 2^{64} байт. Правда, из этого огромного (пока!) объема часть адресов отводится на системные нужды (например, около 2 Гб в системах Windows), но и оставшихся хватит самым “прожорливым” программам. Ясно, что ВАП значительно больше физического адресного пространства (ФАП), определяемого объемом установленной на компьютере памяти, и эта тенденция сохранится в обозримом будущем.

Что касается принципов управления основной памятью, то за время существования ОС их было создано достаточно много. В настоящее время наиболее часто используется организация памяти в виде отдельных фрагментов **постоянной** (страницы) или **переменной** (сегменты) длины, а также их комбинации. При этом в число основных задач управления памятью включают следующие:

- выделение памяти процессам при их создании и освобождение памяти при завершении процессов;
- отслеживание свободной и занятой памяти и динамическое выделение памяти процессам по их запросам (например, для поддержки динамических структур данных);
- преобразование ВА в ФА при загрузке или исполнении кода;
- поддержка механизма виртуальной памяти при взаимодействии с диском;

- защита адресных пространств процессов от воздействия со стороны других процессов и в то же время поддержка общей разделяемой памяти.