**Экзаменационный лист**

**18 января 2021 г.** по дисциплине **«Элементы управления в АСОИУ»**

Начало **09:00**  билет № **4** группа **ИУ5-71б**

Окончание **10:05** студент **Белоусов Евгений Александрович**

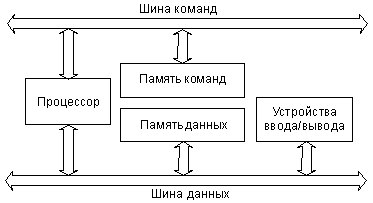
Оценка \_\_\_\_\_\_\_\_\_ экзаменатор **Нестеров Юрий Григорьевич**

**Вопрос 1.** **Принстонская и гарвардская архитектуры: достоинства и недостатки.**

Наиболее распространенный тип *архитектуры*   *микропроцессорных систем* — *архитектура* с общей, единой *шиной* для данных и команд (одношинная, или **принстонская**, **фон-неймановская**   *архитектура* ). Соответственно, в составе системы в этом случае присутствует одна общая *память*, как для данных, так и для команд.



Но существует также и альтернативный тип *архитектуры* *микропроцессорной системы* — это *архитектура* с раздельными *шинами* данных и команд (двухшинная, или гарвардская, *архитектура* ). Эта *архитектура* предполагает наличие в системе отдельной *памяти* для данных и отдельной *памяти* для команд. Обмен процессора с каждым из двух типов *памяти* происходит *по* своей *шине*.



*Архитектура* с общей *шиной* распространена гораздо больше, она применяется, например, в персональных компьютерах и в сложных *микрокомпьютерах*. *Архитектура* с раздельными *шинами* применяется в основном в однокристальных *микроконтроллерах*.

*Архитектура* с общей *шиной* (принстонская, фон-неймановская) проще, она не требует от процессора одновременного обслуживания двух *шин*, контроля обмена *по* двум *шинам* сразу. Наличие единой *памяти* данных и команд позволяет гибко распределять ее объем между кодами данных и команд. Например, в некоторых случаях нужна большая и сложная *программа*, а данных в *памяти* надо хранить не слишком много. В других случаях, наоборот, *программа* требуется простая, но необходимы большие объемы хранимых данных. Перераспределение *памяти* не вызывает никаких проблем, главное — чтобы *программа* и данные вместе помещались в *памяти* системы. Как правило, в системах с такой *архитектурой**память* бывает довольно большого объема (до десятков и сотен *мегабайт*). Это позволяет решать самые сложные задачи.

*Архитектура* с раздельными *шинами* данных и команд сложнее, она заставляет *процессор* работать одновременно с двумя потоками кодов, обслуживать обмен *по* двум *шинам* одновременно. *Программа* может размещаться только в *памяти* команд, данные — только в *памяти* данных. Такая узкая специализация ограничивает круг задач, решаемых системой, так как не дает возможности гибкого перераспределения *памяти*. *Память* данных и *память* команд в этом случае имеют не слишком большой объем, поэтому применение систем с данной *архитектурой* ограничивается обычно не слишком сложными задачами.

**Основное преимущество гарвардской архитектуры заключается в быстродействии.**

Дело в том, что при единственной *шине* команд и данных *процессор* вынужден *по* одной этой *шине* принимать данные (из *памяти* или *устройства ввода/вывода* ) и передавать данные (в *память* или в *устройство ввода/вывода* ), а также читать команды из *памяти*. Естественно, одновременно эти пересылки кодов *по* магистрали происходить не могут, они должны производиться *по* очереди. Современные процессоры способны совместить во времени выполнение команд и проведение циклов обмена *по* системной *шине*. Использование конвейерных технологий и быстрой кэш-памяти позволяет им ускорить процесс взаимодействия со сравнительно медленной системной *памятью*. Повышение тактовой частоты и совершенствование структуры процессоров дают возможность сократить *время выполнения* команд. Но дальнейшее увеличение быстродействия системы возможно только при совмещении пересылки данных и чтения команд, то есть при переходе к *архитектуре* с двумя *шинами*.

В случае двухшинной *архитектуры* обмен *по* обеим *шинам* может быть независимым, параллельным во времени. Соответственно, структуры *шин* (количество разрядов кода адреса и кода данных, порядок и скорость обмена информацией и т.д.) могут быть выбраны оптимально для той задачи, которая решается каждой *шиной*. Поэтому при прочих равных условиях переход на двухшинную *архитектуру* ускоряет работу *микропроцессорной системы*, хотя и требует дополнительных затрат на аппаратуру, усложнения структуры процессора. *Память* данных в этом случае имеет свое распределение адресов, а *память* команд — свое.

Проще всего преимущества двухшинной *архитектуры* реализуются внутри одной микросхемы. В этом случае можно также существенно уменьшить влияние недостатков этой *архитектуры*. Поэтому основное ее применение — в *микроконтроллерах*, от которых не требуется решения слишком сложных задач, но зато необходимо максимальное *быстродействие* при заданной тактовой частоте.

**Вопрос 2. Микропроцессоры IA-32: организация логического адресного пространства.**

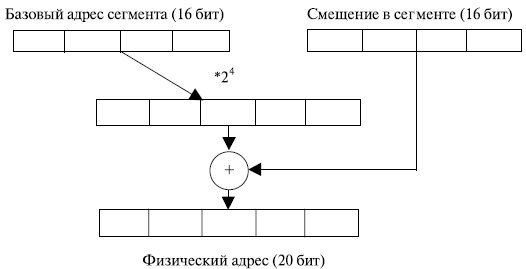
*Архитектура компьютера* различает физическое *адресное пространство* (ФАП) и логическое *адресное пространство* (ЛАП). **Физическое адресное пространство**представляет собой простой *одномерный массив* байтов, *доступ* к которому реализуется аппаратурой памяти *по* адресу, присутствующему на *шине адреса* *микропроцессорной системы*. **Логическое адресное пространство** организуется самим программистом исходя из конкретных потребностей. Трансляцию логических адресов в физические осуществляет блок управления памятью *MMU*.

В архитектуре современных *микропроцессоров* ЛАП представляется в виде набора элементарных структур: байтов, сегментов и страниц. В микропроцессорах используются следующие варианты организации **логического адресного пространства**:

* **плоское (линейное) ЛАП**: состоит из массива байтов, не имеющего определенной структуры; трансляция адреса не требуется, так как *логический адрес* совпадает с физическим;
* **сегментированное ЛАП**: состоит из сегментов - непрерывных областей памяти, содержащих в общем случае переменное число байтов; *логический адрес* содержит 2 части: идентификатор сегмента и смещение внутри сегмента; *трансляцию адреса* проводит блок *сегментации* *MMU*;
* **страничное ЛАП**: состоит из страниц - непрерывных областей памяти, каждая из которых содержит фиксированное число байтов. *Логический адрес* состоит из номера (идентификатора) страницы и смещения внутри страницы; *трансляция логического адреса в физический* проводится блоком *страничного преобразования* *MMU*;
* **сегментно-страничное ЛАП**: состоит из **сегментов**, которые, в свою очередь, состоят из страниц; *логический адрес* состоит из идентификатора сегмента и смещения внутри сегмента. Блок сегментного преобразования *MMU* проводит трансляцию *логического адреса* в номер страницы и смещение в ней, которые затем транслируются в *физический адрес* блоком *страничного преобразования* *MMU*.

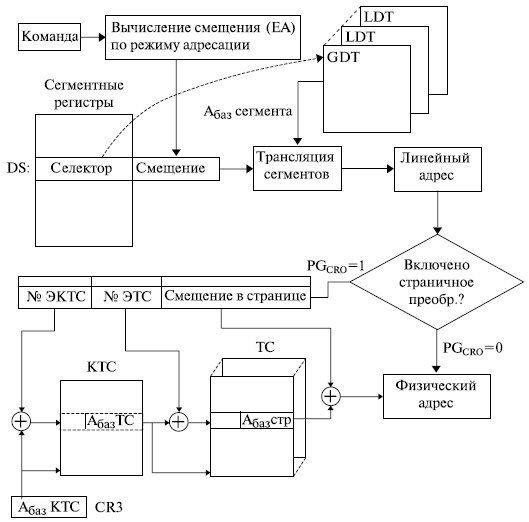
*Микропроцессор* способен работать в двух режимах: реальном и защищенном.

При работе в **реальном режиме**возможности процессора ограничены: емкость адресуемой памяти составляет 1 Мбайт, отсутствует *страничная организация* памяти, **сегменты** имеют фиксированную длину 216 *байт*. Этот режим обычно используется на начальном этапе загрузки компьютера для перехода в **защищенный режим**.



В **реальном режиме***сегментные регистры* процессора содержат старшие 16 *бит* *физического адреса* начала **сегмента**. Сдвинутый на 4 разряда влево **селектор**дает 20-разрядный базовый *адрес* сегмента. *Физический адрес* получается путем сложения этого адреса с 16-разрядным значением смещения в сегменте, формируемого *по* заданному режиму адресации для операнда или извлекаемому из регистра *EIP* для команды. *По* полученному адресу происходит *выборка* информации из памяти.

Наиболее полно возможности *микропроцессора* *по* адресации памяти реализуются при работе в **защищенном режиме**. Объем адресуемой памяти увеличивается до 4 Гбайт, появляется возможность страничного режима адресации. **Сегменты** могут иметь переменную длину от 1 байта до 4 Гбайт.



Как уже отмечалось, основой формирования *физического адреса* служит *логический адрес*. Он состоит из двух частей: **селектора**и **смещения в сегменте**.

**Селектор** содержится в *сегментном регистре* *микропроцессора* и позволяет найти описание сегмента **(дескриптор)** в специальной таблице дескрипторов. **Дескрипторы** сегментов хранятся в *системных объектах* глобальной (*GDT*) и локальных (*LDT*) таблицах дескрипторов. **Дескриптор**играет очень важную роль в функционировании *микропроцессора*, от формирования *физического адреса* при различной организации адресного пространства и до организации мультипрограммного режима работы.

**Сегменты** *микропроцессора*, работающего в **защищенном режиме**, характеризуются большим количеством параметров. Поэтому в универсальных 32-разрядных микропроцессорах *информация* о сегменте хранится в специальной 8-байтной структуре данных, называемой **дескриптором**, а за *сегментными регистрами* закреплена основная *функция* - *определение* местоположения дескриптора.