Типы процессорных архитектур

Основные типы процессорных архитектур

- **Классическая фон-Неймановская архитектура.**

Большинство микропроцессоров и микроконтроллеров малой и средней производительности. Универсальные шины адреса и данных для доступа к памяти программ, данных и устройствам ввода-вывода.

- **(Модифицированная) Гарвардская архитектура.**

Изолированные шины для записи и чтения операндов в память данных, считывания из и записи в память программ. Резкое повышение производительности за счет использования многоуровневого конвейера команд и совмещения операций доступа к данным. Аппаратная поддержка умножения с накоплением для задач цифровой фильтрации и управления.

- **Мультипроцессорные архитектуры** современных процессоров и микроконтроллеров с общей памятью и периферийными устройствами.

Основные типы процессорных архитектур

- **CISC – архитектура** *(Complex Instruction Set Computing)* - Универсальный набор команд разной длины
- **RISC – архитектура** *(Reduced Instruction Set Computing)* - архитектура. Поддержка сокращенного набора команд, каждая из которых имеет фиксированный формат и выполняется за один машинный цикл. Расширение вычислительных возможностей возлагается на трансляторы и компиляторы с языков высокого уровня.
- **ARM - архитектура** *(Advanced RISC Machine)* – усовершенствованная RISC-машина.

Семейство процессоров ARM-Cortex

2000-е годы – Фирма ARM создала несколько линеек процессоров, среди которых ARM11 и семейство **Cortex**, ряд многоядерных процессоров, а также специализированных высоконадежных процессоров для ответственных применений.

Все процессоры, ориентированные на встраиваемые применения, делятся на три группы в зависимости от требований конечных потребителей:

- Cortex-A
- Cortex-R
- Cortex-M

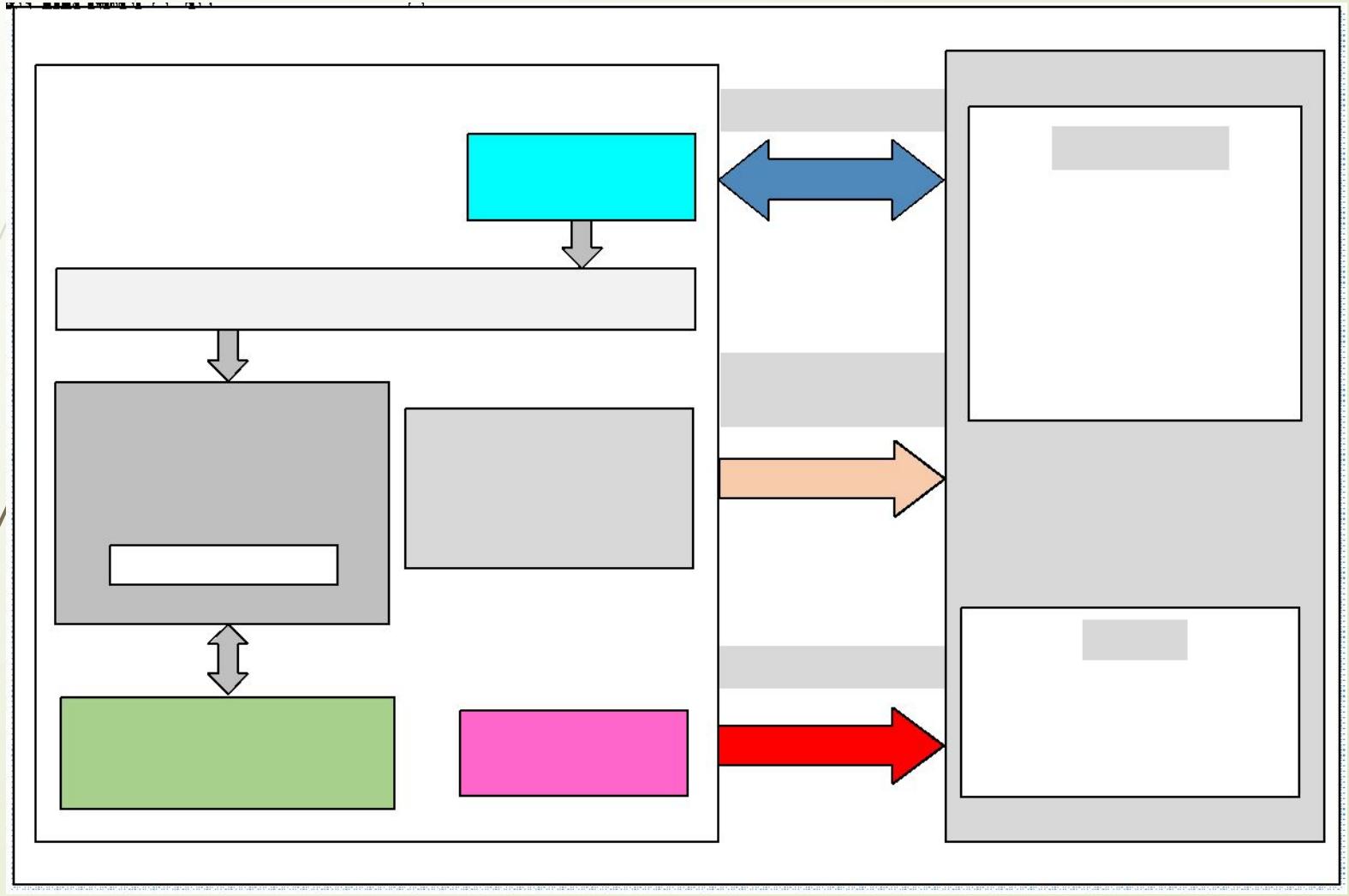
Семейство процессоров ARM-Cortex

Семейство **Cortex-A** предназначено для встраивания в изделия, требующие большой вычислительной мощности (тактовая частота до 3 – 4 ГГц), в том числе, в компьютеры и устройства связи.

Процессоры **Cortex-R4/5/7** имеют архитектуру, предназначенную для работы в очень ответственных устройствах, где надежность и предсказуемость имеет первостепенное значение: авиация, транспорт, энергетика и т.д. Это избыточные системы, реализованные на больше чем одном ядре. В них имеется система контроля и предупреждения программного сбоя.

Контроллер управления приводом подачи станка не требует огромной вычислительной мощности и может быть реализован на более дешевом процессоре из семейства **Cortex-M**. Именно это семейство ориентировано на микроконтроллерные применения, связанные с управлением оборудованием в реальном времени.

Упрощенная фон-Неймановская архитектура



Упрощенная фон-Неймановская архитектура

- ***Программное управление***. Все действия, которые должен выполнить процессор описаны в программе, расположенной в памяти. Программа представляет собой набор управляющих слов (кодов команд), которые «понятны» данному процессору, то есть могут быть декодированы и выполнены.
- 2) ***Последовательное выполнение команд***. Команды считываются из памяти, расшифровываются и выполняются последовательно. За порядком выполнения команд следит специальный регистр процессора – *счетчик команд PC*, содержимое которого автоматически модифицируется процессором в зависимости от длины текущей команды. Он всегда содержит *адрес очередной команды, подлежащей выполнению*. Последовательное выполнение команд может нарушаться специальными командами условной или безусловной передачи управления, суть которых сводится к загрузке в счетчик команд PC нового адреса.
- 3) ***Память адресуется исключительно процессором***. Каждая ячейка памяти имеет свой персональный адрес, по которому процессор может обратиться к ней по чтению или записи, выставляя адрес этой ячейки на шину адреса. В памяти хранятся *слова информации в двоичном коде*, значения которых может интерпретировать только процессор. Никакого признака типа хранимой информации в памяти нет.

Упрощенная фон-Неймановская архитектура

- 4) ***Направление передачи данных*** от памяти к процессору (чтение) или от процессора к памяти (запись) определяет только процессор, выставляя *на шину управления* либо сигнал чтения, либо - записи данных.
- 5) ***Память является однородной***. «С точки зрения» процессора нет возможности отличить, хранится ли в данной ячейке памяти код команды (оптокод) или данные. Первый раз обращаясь к памяти процессор «по умолчанию» считает, что расположенные там данные – код операции. Код операции автоматически попадает в *регистр команд* и подвергается расшифровке. Все последующие обращения процессора к памяти зависят от типа текущей команды. Если эта команда требует получения данных из памяти, то в процессе ее выполнения следует дополнительный цикл обращения к памяти, но уже за данными. Адрес этой ячейки памяти генерирует процессор и выставляет на шину адреса. Считанные из памяти данные попадают не в регистр команд, а в один из внутренних регистров процессора.

Недостатки фон-Неймановской архитектуры

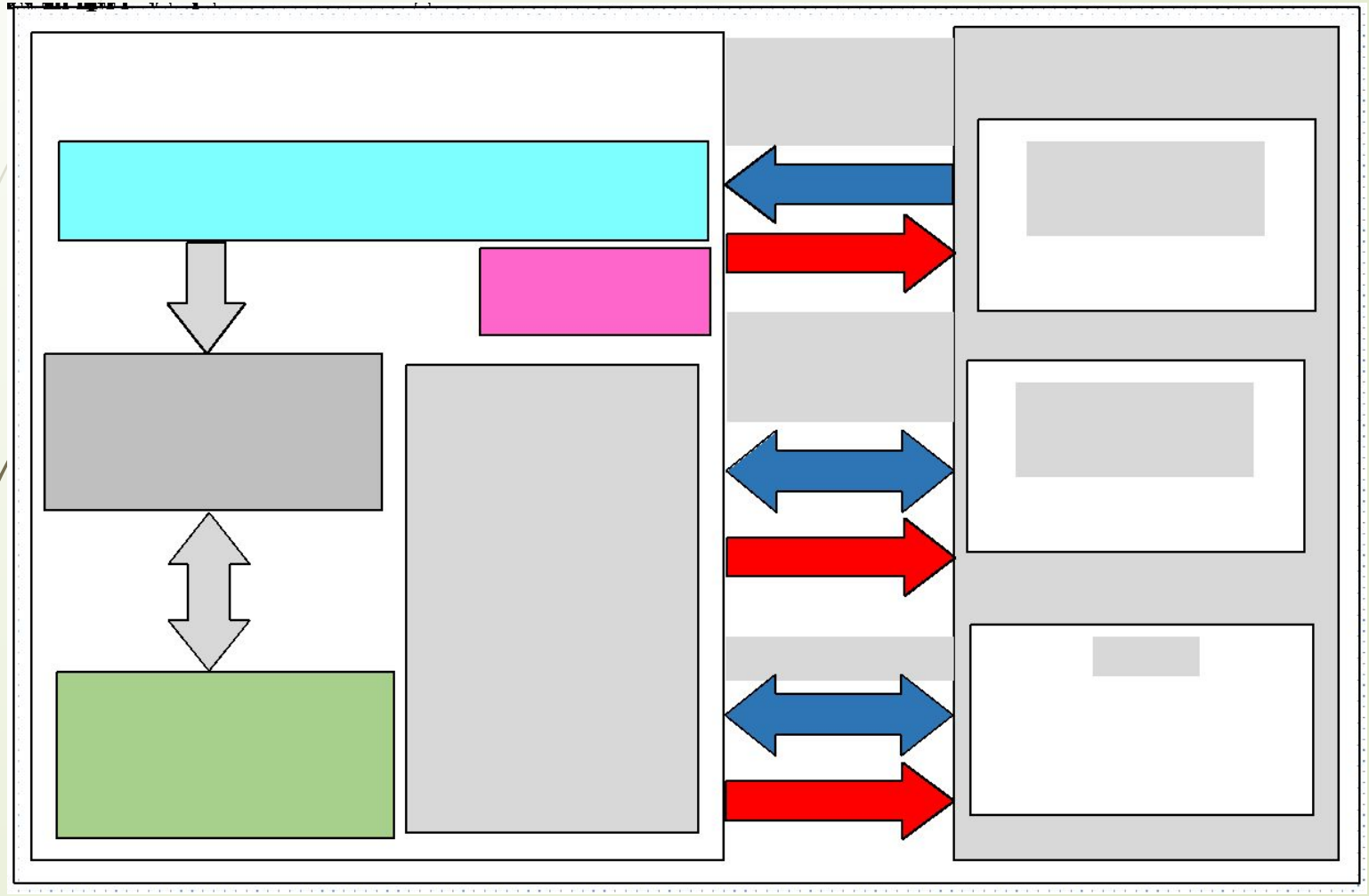
- 1. **Наличие общих шин** для обращения к памяти программ и памяти данных

Наличие в процессорах с фон-Неймановской архитектурой общих шин для обращения и к памяти программ, и к памяти данных делает одновременный, параллельный доступ к этим областям памяти невозможным. Это означает, что считывать очередной код команды из памяти и одновременно получать операнд из памяти для уже считанной команды, находящейся на этапе выполнения, невозможно.

- 2. **Низкая пропускная способность канала связи между памятью и процессором** (интерфейса «Процессор» – «Память»)
- 3. **Память является однородной.**

Память программ, и память данных находятся в общем адресном пространстве. Программа может располагаться в общем случае как в ПЗУ, так и в ОЗУ. При этом архитектура процессора не предполагает никаких аппаратных средств защиты области кодовой памяти с расположенной там программой от преднамеренного или непреднамеренного доступа по записи.

Гарвардская архитектура процессоров



Гарвардская архитектура процессоров

Отличительная особенность Гарвардской архитектуры:

- Физически разная память для хранения команд и данных (кодовая память и память данных).
- Физически разные интерфейсы *«Процессор»* – *«Кодовая память»* и *«Процессор»* – *«Память данных»*.
- Физически разные интерфейсы *«Процессор»* – *«Кодовая память»* и *«Процессор»* –*«УВВ»*.

Гарвардская архитектура процессоров

Отличительная особенность Гарвардской архитектуры:

- Возможность параллельного выполнения нескольких действий сразу: считывания кода очередной команды из кодовой памяти; чтения значений операндов из памяти данных или сохранения в ней результата предыдущей операции.
- Параллельно могут выполняться также операции получения очередной команды из кодовой памяти и чтения/записи в устройства ввода/вывода.
- Если память данных является двух-портовой, то возможен еще больший параллелизм: считывание очередного операнда и одновременное сохранение результата предыдущей операции (так и делается в сигнальных процессорах)

Гарвардская архитектура процессоров

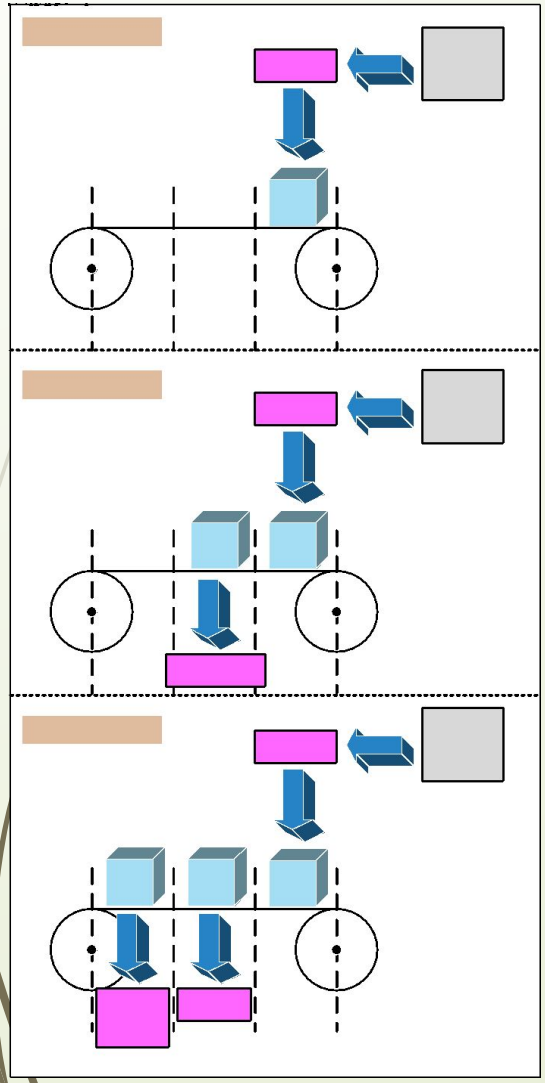
Недостатки Гарвардской архитектуры:

- **Значительное усложнение аппаратной реализации процессора.**

Большое число шин означает также большое число выводов процессора, если элементы памяти – внешние, а также наличие для каждой из них своего собственного устройства управления и синхронизации.

Именно усложнение аппаратуры задержало разработку процессоров с Гарвардской архитектурой на десятилетия. Она стала возможной только при резком повышении уровня интеграции транзисторов на кристалле и удешевлении процессорных БИС.

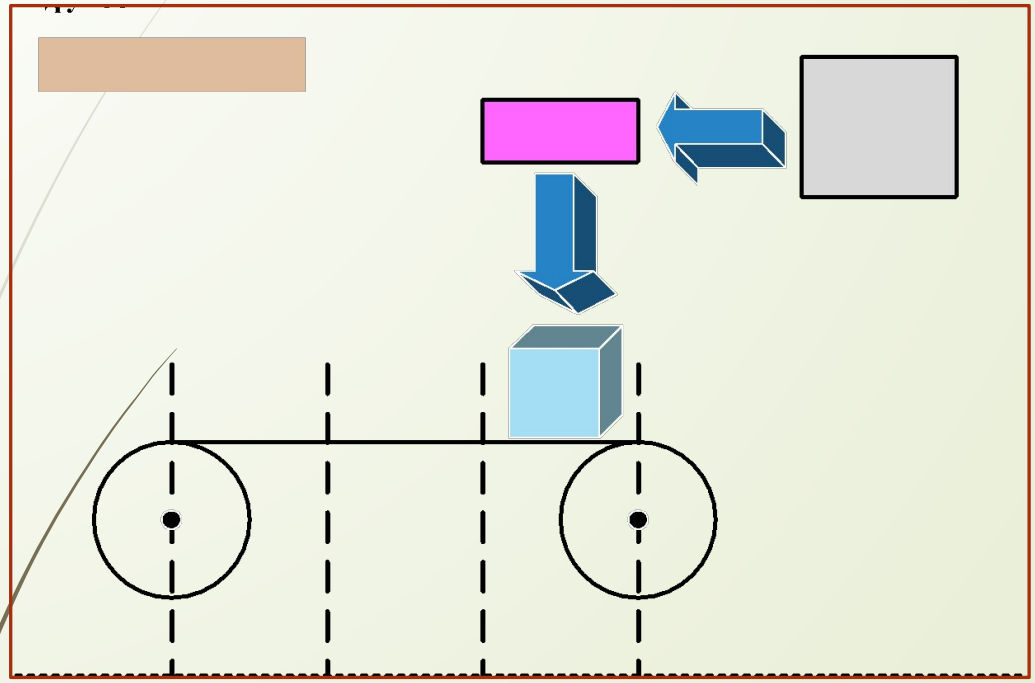
3-уровневый конвейер команд RISC-процессоров ARM-Cortex-M3/M4/M4F.



RISC-архитектура - Reduced Instruction Set Computer – процессор с сокращенным набором команд. Главной отличительной особенностью таких процессоров является выполнение большинства команд за ***один такт процессора***. Это возможно только в *конвейерных архитектурах.*

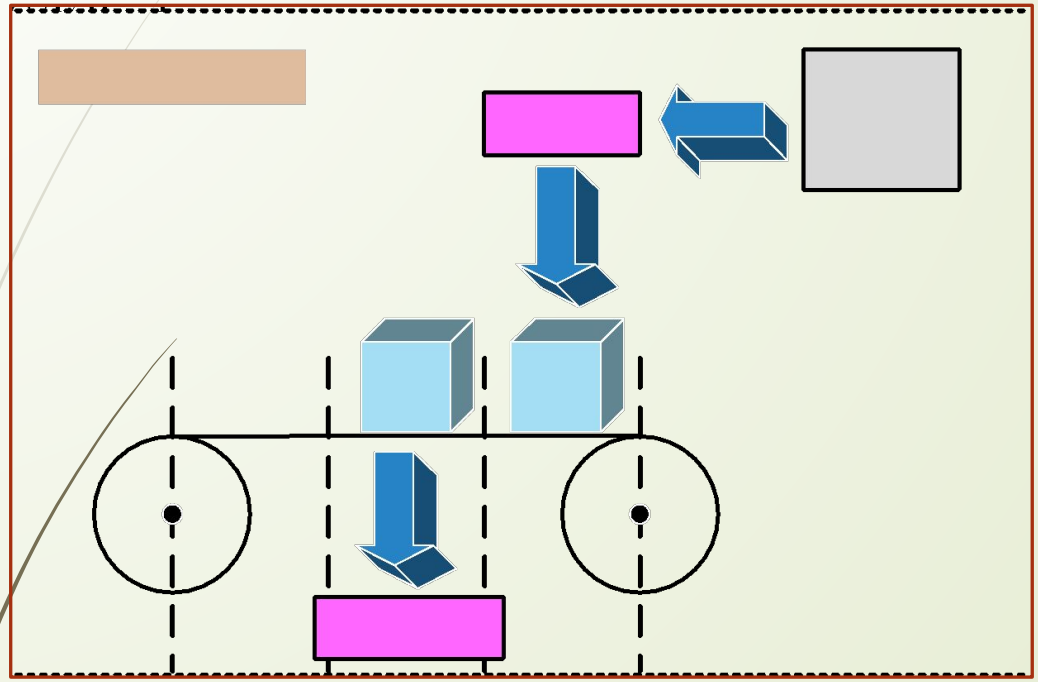
Практически все современные процессоры, имеющие Гарвардскую архитектуру, обрабатывают команды на так называемом «*конвейере команд*» и являются RISC-процессорами. Задача обработки команды разбивается на несколько этапов, на каждом из которых над командой выполняются строго определенные действия. Эти действия зависят от места расположения команды на конвейере. Каждое место (каскад конвейера) имеет свой собственный обработчик команды.

Конвейерная обработка команд



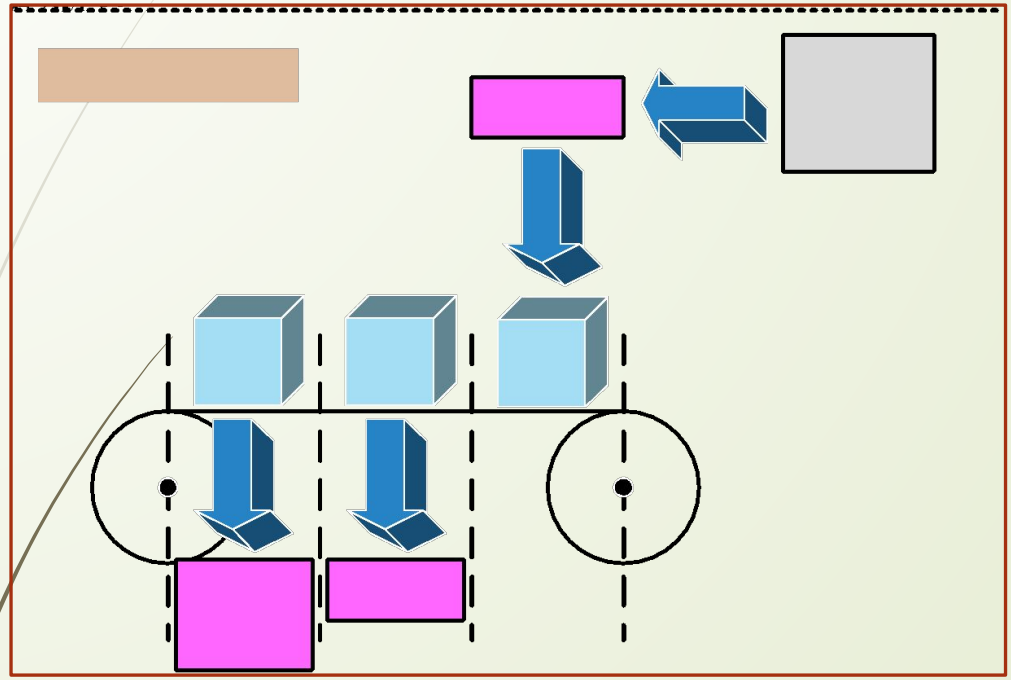
1 такт – выбрать команду из памяти по интерфейсу «Процессор» - «Кодовая память». Обработчик этого каскада извлекает очередную команду из памяти и размещает ее на конвейере – *загружает конвейер* новой командой.

Конвейерная обработка команд



2 такт – декодировать код операции уже находящейся на конвейере команды - определить, что же должен делать операционный блок процессора, какая микропрограмма обработки данных должна быть запущена.

Конвейерная обработка команд



3 такт – выполнить команду.
Обработчиком этого каскада конвейера является операционный блок процессора (включая АЛУ, умножитель, делитель, сдвиговый регистр, сопроцессор).