АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Архитектура программы

1. Компьютерная архитектура
2. Сегменты программы
3. Компоненты программного кода

1

Компьютерная архитектура

Компоненты языка программирования

Компьютерная программа — один из способов реализации понятия алгоритма.

Язык программирования — средство описания алгоритмов.

Программа работает с *данными*.

Данные состоят из отдельных *переменных*, связанных между собой непосредственно или косвенно.

В языке программирования имеются средства *описания данных*.

Программист с помощью средств описания данных конструирует различные формы их представления — *типы данных*.

Программа использует *набор операций (систему команд),* которые можно выполнять над данными.

Зачем нужно знать компьютерную архитектуру

Ассемблер — это язык программирования, в котором система команд, способы адресации, машинные слова и их адреса обозначаются *символическими именами*.

Си — «*чистый компилятор*» — язык системного программирования, на котором пишутся ядра операционных систем, драйверы и т.п.

Язык Си и компьютерная архитектура

Имеет прямой выход на компьютерную архитектуру.

Базовые типы данных совпадают с основными формами представления данных в процессоре.

Имеется прямое соответствие между переменными и машинными словами.

Набор операций принадлежит системе команд.

Указатели интерпретируются как адреса.

Имеется возможность работы с памятью на низком (архитектурном) уровне.

Компьютерная архитектура

Прямо адресуемая память — основа компьютерной архитектуры.

Память представляет *собой множество элементарных ячеек (массив).* Номера ячеек — **адреса**.

Все данные (адрес памяти и ее содержимое) представляются в виде *машинных слов*. **Машинное слово** — набор двоичных разрядов. Для записи содержимого машинных слов используется *шестнадцатеричная система счисления*.

Минимальный размер
машинного слова для хранения, обработки, передачи данных —
8 разрядов (1 байт).

Стандартное машинное слово имеет разрядность, соответствующую разрядности процессора (например, 32 разряда или 64 разряда).

Числовые значения хранятся в памяти, начиная с младшего байта.

Сегментная организация памяти

Сегмент — непрерывная область памяти, хранящая данные одного вида, имеющая собственную систему относительной адресации и ограничения доступа.

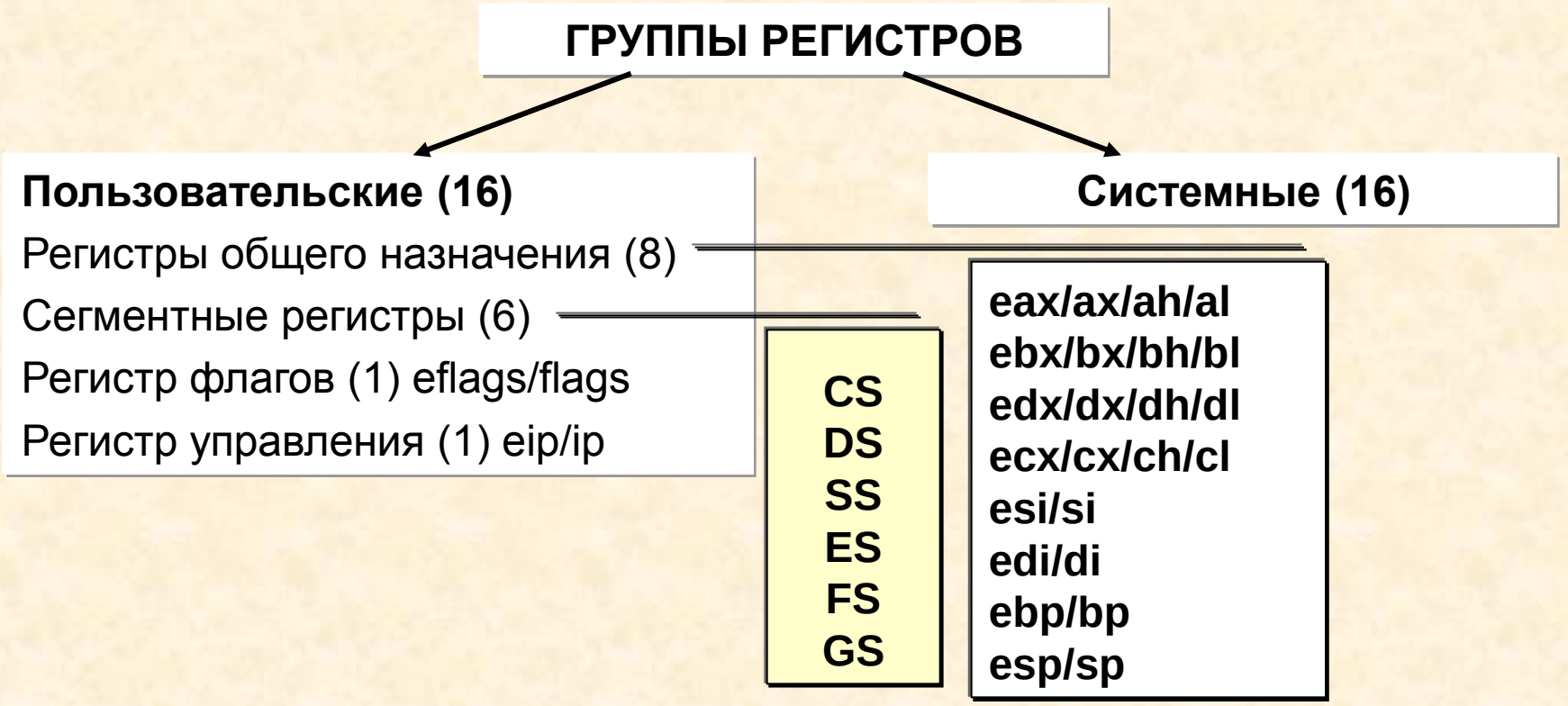
Выполняемая программа состоит из нескольких сегментов. При трансляции разные компоненты программы попадают в различные сегменты программного кода.

Процессор имеет в своем составе **регистры** — набор машинных слов**.** Регистры могут хранить как данные, так и адреса памяти.

Базовый регистр сегмента содержит его начальный адрес.

Регистры, работающие с данными сегмента, содержат относительный адрес данных от начала сегмента, или *смещение***.** Результирующий адрес получается путем сложения содержимого этих регистров.

Программная модель микропроцессора



Программная модель микропроцессора
Регистры общего назначения

eax/ax/ah/al — регистр-аккумулятор.
Применяется для хранения промежуточных данных.

ebx/bx/bh/bl — базовый регистр.
Применяется для хранения базового адреса некоторого объекта в памяти.

ecx/cx/ch/cl — регистр-счетчик.
Применяется в командах, производящих некоторые повторяющиеся действия.

edx/dx/dh/dl — регистр данных.
Применяется для хранения промежуточных данных.

Программная модель микропроцессора
Регистры общего назначения

esi/si — индекс источника.
В цепочечных операциях содержит текущий адрес элемента в цепочке-источнике (первоначальная строка).

edi/di — индекс приемника (получателя).
В цепочечных операциях содержит текущий адрес в цепочке-приемнике (результирующая строка).

esp/sp — регистр указателя стека.
Содержит указатель вершины стека в текущем сегменте стека.

ebp/bp — регистр указателя базы кадра стека.
Предназначен для организации произвольного доступа к данным внутри стека.

2

Сегменты программы

Компоненты программы и компьютерной архитектуры

Компоненты программы находятся в *памяти*.

Память логически разделяется на области — *сегменты*.

Сегмент — логическое образование, накладываемое на требуемые участки физического адресного пространства.

сегмент команд	**CS**
сегмент данных	**DS**
сегмент стека	**SS**
сегмент резерва	**ES**

---[Сегментные регистры

Набор операций в программе соответствует *системе команд* процессора, на котором она выполняется.

Размещение сегментов в адресном пространстве

Память, переменные, структуры данных

Переменные — единственный «представитель» памяти в программе.

Управление памятью определяется тем, каким образом работает с переменными и с образованными ими структурами данных язык программирования.

Большинство языков программирования компилирующего типа однозначно закрепляет за переменными их типы данных. Работа с памятью ограничена теми областями, в которых эти переменные размещены.

Задача эффективного размещения данных в памяти относится к области *системного программирования*.

Сегменты программы

Сегмент	**Регистры**
Сегмент команд	CS — сегментный, IP — адрес команды
Сегмент данных	DS — сегментный
Сегмент стека	SS — сегментный, SP — указатель стека
Динамическая память	DS — сегментный
Динамически связываемые библиотеки (DLL)	CS — сегментный, IP — адрес команды

Содержимое сегментов программы

Сегмент	**Что содержит**
Сегмент команд	Программный код (операции, операторы)
Сегмент данных	Глобальные (статические) данные
Сегмент стека	Локальные данные функций, «история» работы программы
Динамическая память	Динамические переменные, создаваемые при работе программы
Динамически связываемые библиотеки (DLL)	Программный код разделяемых библиотек

Создание сегментов программы

Вся программа (в том числе и алгоритм) рассматривается как данные для работы других программ, например, трансляторов.

Сегмент	**Когда создается**
Сегмент команд	Трансляция
Сегмент данных	Трансляция
Сегмент стека	При загрузке
Динамическая память	При загрузке, выполнении
Динамически связываемые библиотеки (DLL)	При загрузке

Виртуальное адресное пространство

Сегменты программы размещаются
в виртуальном адресном пространстве.

В процессоре имеется скрытая от программ система отображения виртуальных адресов на физические адреса, которая находится под управлением операционной системы и реализует

защиту памяти программы,

разделение памяти (сегменты динамически связываемых библиотек),

загрузку программы в память «по частям»,

иллюзию наличия неограниченной памяти.

Внутренне представление программы в виртуальной памяти
занимает все адресное пространство,
как будто в памяти нет других программ и операционной системы.

3

Компоненты программного кода

Компоненты программного кода

Программный код — последовательность команд**,** размещенная в сегменте команд (**CS**), на начало которого ссылается одноименный регистр процессора.

Каждая команда имеет в сегменте свой относительный адрес.

Регистр процессора **IP (Instruction Pointer)** содержит адрес очередной выполняемой команды.

Принцип хранимой программы

Программный код хранится в той же самой памяти, что и обрабатываемые данные

Программный код представляет собой специфические данные, с которыми работает процессор во время выполнения программы.

Компоненты программного кода

Команда представляет собой машинное слово, обычно переменной длины, и состоит из нескольких полей.

Поле **код операции** является обязательным и указывает на выполняемое командой действие.

Поле **операндов**. Это могут быть регистры процессора, содержащие данные и ячейки памяти. В команде в явном или неявном виде должны содержаться их адреса.

Переменные, используемые в программе, при трансляции «становятся» областями памяти в различных сегментах (сегменты данных, стека, динамической памяти), где получают свои адреса.

Виды операций в системе команд

Совокупность команд процессора называется системой команд.

Команды *перемещения и обработки данных* выполняют перенос данных между ячейками памяти и регистрами данных процессора, а также все действия по их обработке.

Команды *проверки состояний (условий)* проверяют результат выполнения операций, состояния регистров процессора, устройств ввода-вывода и т.п.

Команды *условного и безусловного переходов* определяют адрес следующей команды в зависимости от результата проверки некоторого условия. Если условие выполняется, то происходит переход, иначе — следующая команда.

Выполнение программы

ПРОЦЕССОР

Регистры **CS:IP**

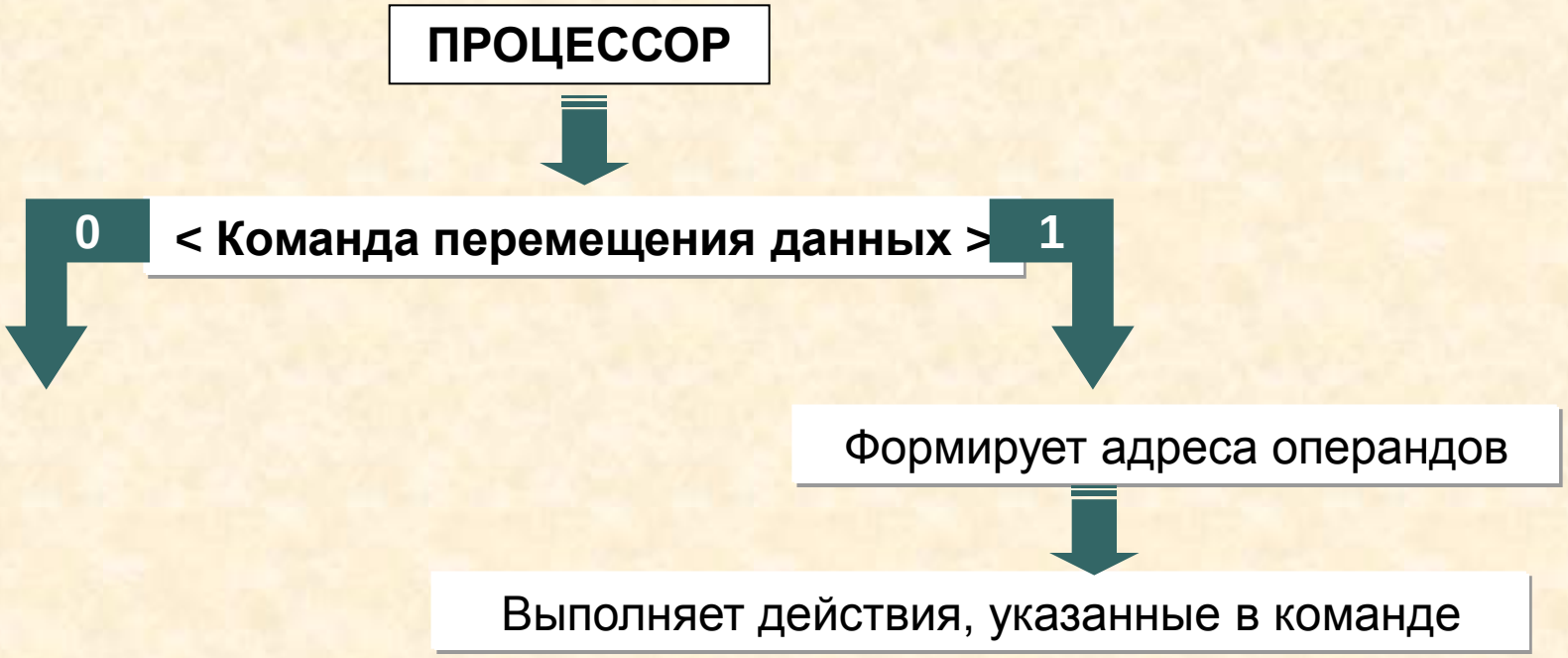
---[**1.** Выбор из памяти очередной команды и ее расшифровка

---[**2.** Увеличение **IP** в зависимости от длины текущей команды, например, **IP=IP+5**

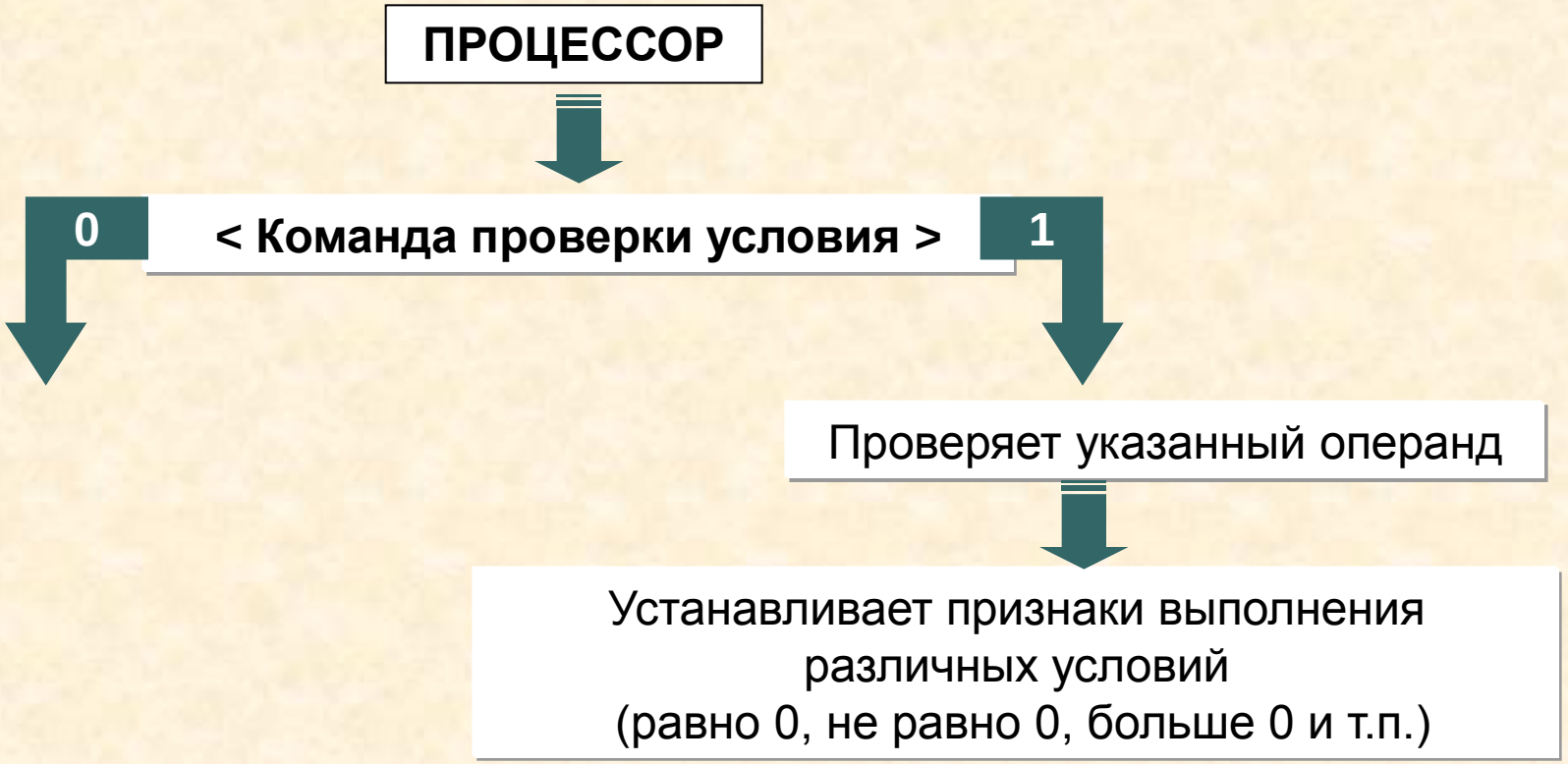
Ссылка на следующую команду

Внутреннее представление программы соответствует блок-схеме, имеющей те же самые составные элементы: ***действие, проверка условия, переход****.*

Расшифровка команды



Расшифровка команды



Расшифровка команды