1) Операционные системы

Оператор в компьютерных системах первых поколений занимался тем, что принимал от программистов программу и исходные данные, вводил и запускал её на исполнение. При этом дисплей и клавиатура появились у вычислительных машин не сразу, вместо них использовался пульт управления вычислительной машиной. С развитием вычислительной техники появилась естественная идея заменить оператора специальной программой, которая будет заменять его работу. Такие программы получили название операционных систем.

Операционная система загружает программы в память, контролирует их исполнение, предоставляет результаты работы в понятном для человека виде, обеспечивает возможность одновременного исполнения нескольких программ. В современном мире операционные системы можно условно разделить на пользовательские — где основной целью является предоставление конечному пользователю удобства работы с вычислительной системой, серверные, предназначенные для обработки большого количества запросов от других систем, и встроенные — целью которых является поддержка функционирования аппаратного обеспечения вычислительных систем, датчиков, преобразователей сигнала и другого оборудования.

Современные операционные системы: Windows, Unix, Linux, Android, IOS, DOS. Отдельно следует упомянуть гипервизоры операционные системы, которые предназначены для управления другими операционными системами.

Операционная система содержит ядро и окружение (библиотеки, файлы) пользовательских программ. В ядро обычно включаются подсистемы, позволяющие управлять и эффективно разграничивать данные и программы пользователя для исключения их неразрешенного взаимодействия между собой, драйверы аппаратного обеспечения, подсистемы обобщенного ввода и вывода информации.

2) История UNIX/Linux

Название Unix произошло от акронима UNICS (UNiplex Information and Computing System). Разработчики Кен Томпсон, Денис Риччи и Руд Канадей создали однопользовательскую (позднее двухпользовательскую), написанную на ассемблере операционную систему для популярной в то время вычислительной системы PDP-7. Позднее имя было заменено на UNIX. Позднее система была переписана на язык C и портирована на PDP-11.

Рождение UNIX (1969–1979):

Начало с Unnamed PDP-7 OS, перешедшей в UNIX версии 1–7.

BSD (Berkeley Software Distribution) появляется как первая ветка UNIX (версии 1.0–4.1).

Развитие BSD и коммерческих UNIX-систем (1980-е):

BSD становится основой для будущих открытых UNIX-систем.

Параллельно развиваются коммерческие закрытые варианты UNIX.

Открытые системы и переход к Linux (1990-е):

BSD-форки, такие как FreeBSD, NetBSD, OpenBSD, продолжают развитие в открытой среде.

Появляется Linux (1991), создавая новую открытую альтернативу UNIX, которая станет доминирующей в будущем.

Консолидация коммерческих UNIX-систем (1990-е–2000-е):

Solaris (Sun Microsystems) становится важной коммерческой платформой.

UNIXWare и OpenServer продолжают развиваться, но постепенно теряют позиции.

Доминирование открытых систем (2000-е):

Linux активно развивается и доминирует в серверной среде.

Mac OS X (на базе Darwin) объединяет элементы BSD и NeXTSTEP.

3) Современность

Базовой версией коммерческих операционных систем UNIX послужило ядро ОС UNIX System V. Изначально разработанное для поддержки масштабируемых вычислительных систем оно представляет собой ядро с уникальными для всего семейства UNIX функциями программной поддержки горячей замены аппаратных компонентов (включая память и процессоры) без перезагрузки ядра, высокой масштабируемостью (одновременно могут работать от 1 до сотен процессоров, поддерживаются терабайты ОЗУ), развитой архитектурой, предоставляемой пользовательским процессам. Наиболее известными, используемыми на предприятиях, в настоящее время являются Solaris 2.x+ (Sun/Oracle) и AIX (IBM).

BSD и ее производные часто используются для организации сетевой инфраструктуры, поскольку поддержка большого количества протоколов и средств, позволяющих управлять сетевым взаимодействием, в них широко развита. Особняком стоит MacOS прямой наследник ОС NextStep - предоставляющая удобную оболочку для пользователя и развитый пользовательский интерфейс.

Linux представлен богатым выбором операционных систем. Отличия в версиях, которые тем не менее базируются на одном программном ядре, состоит в управлении программным обеспечением (пакетные менеджеры) и способах распространения и поддержки ОС.

4) Ядро *NIX

Каждая операционная система выполняет служебные операции в ядре. Ядро представляет собой набор подсистем, которые управляют оборудованием и программами пользователя. Основными подсистемами, которые жизненно важны для любой ОС являются управление памятью, управление процессами и потоками (отвечает за выполнение нескольких программ одновременно на одной ОС), виртуальная файловая система (обмен данными организован с помощью файлов, с точки зрения ОС в UNIX все является файлами), сетевая подсистема, драйвера устройств и платформозависимый код (драйвера, которые поддерживают функционирования системы на различных аппаратных архитектурах). Взаимодействие программ пользователя с ядром происходит с помощью интерфейса системных вызовов.

С точки зрения пользователя любая исполняемая программа представляет собой адресное пространство, которое разбито логически и физически на несколько сегментов. Например, сегмент code, где выполняется собственно сама программа, сегмент data, где находятся необходимые для ее работы данные, сегмент stack, для хранения адресов возврата и передачи аргументов к подпрограммам. Кроме этих основных сегментов еще используются heap или куча, сегменты разделяемых библиотек, memory-maped файлы, т. е. файлы непосредственно отображенные в адресное пространство процесса и другие. Все эти сегменты вместе составляют образ процесса, который работает внутри операционной системы и взаимодействует с ней. Кроме этого, программы могут взаимодействовать между собой, используя ядро и интерфейс системных вызовов (и только через него).

5) Файловая система
ФС представлена в виде дерева. Верхнем элементом иерархии файловой системы является директория root (корень иерархии). Логические разделы дисковых накопителей могут подключаться в файловую систему в точках монтирования созданных администраторами директорий, в которых при подключении их файловая иерархия заменяется на расположенную на разделе диска. Каждая директория содержит два обязательных файла. Файл «.» - это ссылка на саму себя и файл «..» - это ссылка на директорию выше по файловой иерархии. В файловой иерархии есть каталоги, которые использует сама система. К ним, например, относится каталоги /usr/bin, где расположены основные утилиты и программы, поставляемые вместе с ОС.
В файловой системе можно адресоваться при помощи абсолютных и относительных путей. Абсолютный путь — это путь от корня ФС. Относительный путь – это путь от текущей директории, откуда была запущена программа пользователя.
В файловой системе есть специальные виды файлов, которые могут помочь организовать необходимую для пользователя или программ иерархию файлов ссылки на другие файлы. Ссылки бывают двух типов: жесткие и символические ссылки. В ОС UNIX каждому файлу внутри файловой системы для каждого накопителя присваивается свой уникальный номер inode. Директория это просто файл, где указаны имена файлов и их inode. Если два имени файла имеют один и тот же inode (внутри одной точки монтирования) значит это один и тот же файл с таким же содержимым. Или можно сказать, что один файл является жесткой ссылкой на другой. Файл существует в файловой системе до тех пор, пока не удалена последняя жесткая ссылка. Символическая ссылка – файл содержащий абсолютный или относительный путь до файла.

6) Права доступа к файлам
Вывод команды ls -l:
1) Тип файла и права доступа к нему
2) Кол-во жестких ссылок на файл
3) Имя владельца
4) Имя группы владельца
5) Длина файла в байтах
6) Время. По умолчанию показывается mtime, в зависимости от ключей может быть отображено atime (ls -u) или ctime (ls -c).
Права доступа разбиты на несколько групп и определяют права для разных категорий пользователей по отношению к текущему владельцу файла и группе владельца:
| Тип файла | Права для владельца | Права для группы | Права для остальных |
Все типы прав содержат три значения: право на чтение/право на запись/право на исполнение. Право на чтение (r) директории позволяет пользователю просматривать список файлов и поддиректорий, находящихся внутри этой директории. Право на запись (w) в директорию даёт возможность добавлять, удалять и переименовывать файлы в этой директории. Право на исполнение (x) директории даёт возможность "входить" в директорию, т. е. открывать её и перемещаться по её содержимому. Оно не связано с запуском программ, как для файлов, но означает, что пользователь может получить доступ к самой директории и выполнять действия над её содержимым, зная его точные имена.
Символ + (. - нет расширенных прав) в выводе команды ls -l указывает на то, что для файла или директории заданы расширенные списки контроля доступа (ACL, Access Control List). ACL позволяют задавать более детальные права на уровне пользователя и группы, которые выходят за рамки стандартной системы прав.

7) Способы задания прав
Права (для владельца, группы и остальных) принято обозначать в виде трех восьмеричных цифр. Каждая цифра отвечает свою тройку и высчитывается так rwx = $111_2 = 7_8$; r-x = $101_2 = 3_8$; - -x = $001_2 = 1_8$. Таким образом, рассчитывается десятичная цифра, соответствующая определенным правам для каждой тройки типов прав. Команда chmod 644 <имя файла> установит права rw - r- - r- - -.
Еще один способ изменять права это добавлять их или убирать с помощью буквенных обозначений. Сначала указывается, какие права будут изменены. u — права пользователя, g - права группы, o – права для остальных. Далее можно использовать знаки +, - или =.

8) Потоки stdin(0), stdout(1), stderr(2)
Файловый дескриптор — это целое число, которое операционная система использует для идентификации открытого файла или ресурса. Оно выступает как ссылка на структуру данных, содержащую информацию о текущем состоянии взаимодействия с файлом. Дескрипторы уникальны только внутри одного процесса. В каждом процессе в Unix автоматически открываются три стандартных файловых дескриптора (номера зарезервированы):

0 — Стандартный ввод (stdin)
1 — Стандартный вывод (stdout)
2 — Стандартный вывод ошибок (stderr).

9) Интерпретатор команд
Всеми командами (программами и приложениями) управляет интерпретатор команд - shell. Это программная оболочка позволяет организовывать выполнение команд, задавать им переменные окружения, связывать потоки ввода-вывода у различных команд между собой, а также анализировать коды возврата у команд и использовать простейший язык для написания новых исполняемых команд - скриптов.
sh, bash и ksh — это разные типы командных интерпретаторов (оболочек - shell), которые предоставляют интерфейс для взаимодействия с операционной системой через командную строку. Каждый из них имеет свои особенности, синтаксис, функции и возможности.
sh - это одна из первых оболочек, созданных в Unix в конце 1970-х. sh имеет простой синтаксис, базовый набор команд и операций управления потоком (циклы, условия и т.д.). Она довольно ограничена по функционалу по сравнению с современными оболочками, но все еще используется для написания скриптов из-за высокой совместимости с другими Unix-оболочками.
bash — это расширенная версия sh. Она является стандартной оболочкой в большинстве современных дистрибутивов Linux. bash добавляет множество новых возможностей, таких как автодополнение команд, история команд, поддержка работы с массивами и функций, расширенные конструкции для работы с циклами и условиями. bash обеспечивает обратную совместимость с sh, так что большинство скриптов, написанных для sh, работают в bash без изменений.
ksh (KornShell) направлен на работу с бОльшим количеством пользователей.

10) Перенаправление потоков stdin(0), stdout(1), stderr(2)
> (перенаправление стандартного вывода в файл): используется для записи стандартного вывода команды в файл (если файл уже существует, его содержимое будет перезаписано).
>> (добавление стандартного вывода в файл): используется для добавления (аппендирования) стандартного вывода команды в файл (если файл уже существует, содержимое сохраняется, а новые данные добавляются в конец файла).
< (перенаправление стандартного ввода из файла): используется для передачи содержимого файла как стандартного ввода в команду.
<< (here-document, встроенный ввод): позволяет передать текст напрямую в команду через стандартный ввод, используя специальный маркер конца ввода EOF (в начале и в конце передаваемого текста на отдельной строке пишем EOF).
<<< - аналогично <<, но работает только для одной строки (не требует EOF).
&>file или >&file - направить стандартный поток вывода и стандартный поток ошибок в файл. Другая форма записи: >file 2>&1.
Конвейер (|) — это механизм, позволяющий передавать вывод одной команды на вход другой. С помощью конвейера можно комбинировать несколько команд, чтобы выполнять обработку данных поэтапно.

11) Фильтры
Это команды (или программы), которые воспринимают входной поток данных, производят над ним некоторые преобразования и выдают результат на стандартный вывод (откуда его можно перенаправить куда-то еще по желанию пользователя). Фильтр-команды: cat, grep, sort, tail, head, wc, sed.

12) Регулярные выражения
Регулярное выражение определяет шаблон, которому должна соответствовать строка в тексте, в котором проводится поиск на совпадение. Простейший случай регулярного выражения - это последовательность символов, которые соответствуют сами себе.
. — любой одиночный символ.
* — ноль или более вхождений предыдущего символа.
^ — начало строки.
$ — конец строки.
[] — диапазон символов. Например, [a-z] — любой символ от a до z.
Экранирование — это способ использования специальных символов так, чтобы они воспринимались как обычные символы и не имели особого значения в строках или командах, выполняется с помощью символа обратного слэша (\).

13) Команды
mkdir - создание директории
echo - вывод символов на стандартный вывод
cat - конкатенация и вывод файлов на стандартный вывод
touch - создание (если файл отсутствует) или изменение mtime и atime файла
ls - вывод списка файлов в директории
pwd - вывод текущей директории в стандартный вывод
cd - смена текущей директории на указанную
more - постраничная распечатка файла или стандартного ввода
cp — копирование файлов и директорий

rm - удаление файлов
rmdir — удаление пустых директорий
mv - перемещение файлов
head - вывод указанного количества строк с начала файла или stdin
tail - вывод указанного количества строк с конца файлов или stdin
sort - сортировка строк файла или стандартного ввода
grep - выборка строк по шаблону
wc - подсчет количества символов, строк, слов в файле или стандартном вводе.

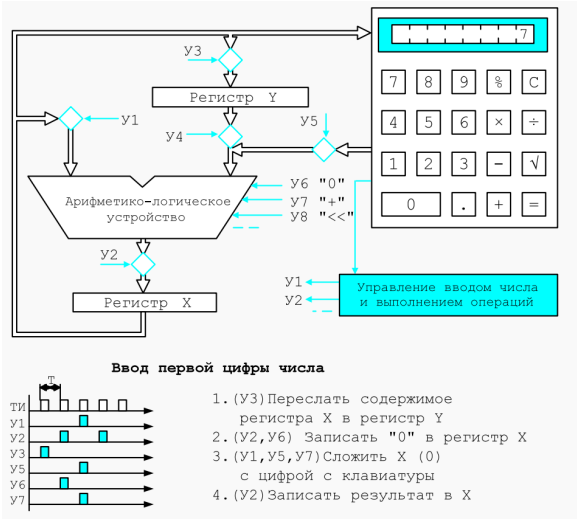
14) Аналоговые ЭВМ
С развитием техники и технологий людьми были сконструированы приборы, которые позволяли определять различные физические величины, такие как скорость, ускорение, расстояния. Например, если мы хотим измерить высоту полета самолета, то мы можем измерить давление атмосферы или послать электромагнитный импульс к земле и посчитать, когда придет отраженный сигнал. Но сами физические значения, такие как высота, сложно использовать в дальнейших вычислениях в их естественном виде, и люди придумали заменять их аналогичными величинам или просто аналогами. В современной вычислительной технике, в качестве аналоговых величин используются значения напряжения. При этом высоте, например, может ставится в соответствие напряжение от датчика высоты. Аналоговая величина всегда непрерывна, и ее поведение полностью копирует поведение соответствующей физической величины. Теперь с этой аналоговой величиной можно проводить вычисления - складывать, вычитать, дифференцировать. Например, при изменении высоты, можно рассчитать вертикальную скорость относительно поверхности. Аналоговые вычисления производится аналоговыми ЭВМ. К сожалению, у таких ЭВМ есть недостатки.
Основной из них - большие габариты таких вычислительных машин. Кроме того, не всегда просто преобразовать физическую величину в аналог и обратно. Аналоговые системы до сих пор широко используются в системах реального времени, в различных измерительных устройствах, в информационно-управляющих системах, потому что у них очень высокое быстродействие для определенного класса задач и удовлетворительная погрешность вычислений. Это происходит за счет того, что аналоговая величина непрерывна.

15) Цифровые ЭВМ
Вопрос о недостаточной точности вычисления аналоговых ЭВМ привел к созданию цифровых ЭВМ. Решили, что вместо сложного устройства, которое точно различает величину амплитуды напряжения в 0,001 вольта, проще использовать сигнал, у которого будет всего два состояния - либо 0, либо 1. Это существенно упростило конструкцию вычислительных машин, приведя к созданию цифровых вычислительных систем.
Отдельные импульсы в цифровых вычислительных машинах объединяются в группы, позволяя закодировать информацию кодовой посылкой в виде одного или нескольких чисел. Цифровое представление существенно упрощает необходимые элементы для построения вычислительной машины.

16) Функциональные элементы ЭВМ
На самом нижнем логическом уровне компьютеры состоят из логических элементов, которые выполняют логические функции. У каждого логического элемента есть один или несколько входов и один выход. На вход подаются значения (0 или 1), внутри элемента выполняется функция, результат которой попадает на выход. Простейшая микросхема объединяет несколько логических элементов и обычно содержит входы и выходы этих элементов, а также входы для подачи напряжения питания.
Для хранения информации и данных в ЭВМ используется оперативная память, информация в которой сохраняется, пока ЭВМ включена. В современных вычислительных системах существует два типа оперативной памяти: динамическая память (DRAM) и статическая память (SRAM).
В DRAM информация хранится на емкости, которая находится в цепи зарядов транзистора. Ячейка памяти состоит из одного транзистора и одного конденсатора. Недостаток такой памяти: емкость постепенно разряжается, соответственно информация теряется. Для того, что бы поддерживать заряд на емкости предусмотрено устройство регенерации памяти: компьютер раз в определенное время (64 мс) сканирует всю имеющуюся оперативную память и считывает каждую ячейку, возобновляя заряд на конденсаторе. Достоинством DRAM является ее дешевизна на единицу хранимой информации, т. к. для одного бита используется всего один транзистор.
Статическая память SRAM не требует регенерации за счет такой конструкции ячейки, где шесть транзисторов объединены в схему, которая самоподдерживает свое состояние за счет обратных цепей связи. Однако цена такой памяти будет больше, кроме того, ячейки данного типа являются более быстродействующими. На их основе строится память, которая обычно встроена в процессор. К такой памяти относятся, например, кэш память различных уровней.
С точки зрения схемы ЭВМ, построенной на логических элементах, в качестве хранения информации используются триггеры. Триггер = «защелка», с помощью которой можно хранить и управлять информацией.
Например, D-триггер представляет собой ячейку информации, хранящую один бит информации. У триггера есть 2 входа. Один из них D - информационный, на него подается значение, которое необходимо сохранить, и C синхронизации, который определяет, будет ли значение на информационном входе записано в триггер. Триггеры объединяются в регистры. Имеется общий вход C, который для всех триггеров один. Когда на него подается 1, одновременно вся информация поместится в ячейки памяти. Таким образом, регистры образуют промежуточное хранение информации.
Передачу информации обычно нужно осуществлять не непрерывно, а в строго заданное (тактовым генератором) время. Для этого предусмотрено логическое устройство, называемое вентилем, которое функционирует по принципу хорошо известного нам крана. Вентиль может быть в открытом состоянии, и тогда информация с выхода одного ре-

гистра поступает на другой, или в закрытом, тогда передача информации блокирована.
У вентиля есть управляющий и информационный сигналы. На его выходе единица будет только в том случае, если информационный и управляющий сигналы будут равны единице (кран открыт и течет вода). Таким образом управляющий сигнал разрешает или запрещает передачу информации.
Вычислительная машина не только передает и хранит данные. Она должна еще выполнять с ними различные арифметические или логические операции. Например, одни из простейших операций сложение и вычитание, осуществляются с помощью сумматора (входит в АЛУ). В общем случае, для сложения 2х разрядов многоразрядного двоичного числа у логического элемента должны присутствовать 3 входа два для значений разрядов слагаемых и один, дополнительный, для входящего переноса из предыдущего разряда, и два выхода Для значения суммы и выходящего переноса. Перенос передается следующему разряду после суммирования предыдущих (аналогично сложению в столбик).
Тактовые генераторы являются «сердцем» ЭВМ. Подобно человеческому сердцу они задают ритм работы вычислительной машины в целом. Все пересылки данных, арифметические или логические операции могут происходить только в строго заданное время, определяемое размером такта генератора. Одни операции могут выполняться по фронту сигнала ТГ, другие по значению, некоторые по спаду сигнала. К концу такта все операции должны быть завершены.

17) Первая ЭВМ: Калькулятор (1)



Рассмотрим принципы функционирования простейшей ЭВМ калькулятора. Он состоит из двух регистров X и Y, хранящих результаты ввода пользователя и промежуточных вычислений, АЛУ, которая может выполнять простейшие арифметические и логические операции, шин и управляющих вентилей, осуществляющих передачу данных между функциональными блоками калькулятора, устройства управления (УУ), клавиатуры и дисплея.
Дисплей постоянно отображает содержимое регистра X. Клавиатура передает значение нажатой клавиши на вентиль У5, каждое нажатие на клавишу запускает УУ, которое в зависимости от текущего состояния ЭВМ формирует последовательность импульсов для выполнения требуемой операции, которая называется циклом импульсов. Каждая группа импульсов выдается последовательно, в моменты, совпадающие с импульсами тактового генератора.
Предположим пользователь вводит первую цифру необходимого ему числа (7). Так как это новая операция, УУ, после своей активации нажатием кнопки 7, выдаст последовательность управляющих импульсов для первой цифры числа. В первую очередь необходимо сохранить предыдущее значение регистра X в регистре У. Для этого должен быть открыт вентиль, управляющий записью в регистр Y. Он открывается управляющим сигналом УЗ. После этого необходимо обнулить регистр X, подготовив его для новой цифры числа, которая была введена с клавиатуры. Для этого должны быть закрыты все вентили, кроме У2 - который осуществляет передачу данных из АЛУ в регистр Y и У6 — который сформирует в АЛУ значение 0. Далее необходимо сложить значение 0 с цифрой с клавиатуры. Для этого содержимое регистра X поступает на правый вход АЛУ (У1), цифра с клавиатуры на правый вход АЛУ (У5), и выбирается операция сложения (У7). В конце цикла необходимо передать результат сложения в регистр X (У2), отобразив его, при этом, на дисплее.

18) Архитектура ЭВМ, Гарвардская, Фон Неймана
Основным отличием между архитектурами является способ работы с памятью. В настоящее время в чистом виде архитектуры встречаются редко, обычно используется их комбинация в разных функциональных блоках ЭВМ.
В Гарвардской архитектуре центральным устройством является Control Unit - управляющее устройство в ЭВМ. Все остальные устройства ЭВМ подключены к управляющему устройству и взаимодействуют через него. Память для команд (instruction memory), данных (data memory) и устройств ввода-вывода физически отделена друг от друга.
В противоположность Гарвардской, архитектура фон Неймана содержит все в общей памяти. Нельзя определить, что находится в ячейке, данные или инструкции, без дополнительного анализа кода самой программы. При этом устройство обращения к памяти едино для данных и инструкций, что упрощает конструкцию ЭВМ. Отдельно выделены устройства ввода-вывода, которые являются внешними по отношению к процессору - объединению памяти, АЛУ и управляющего устройства.

19) Структура БЭВМ-NG

Память состоит из 2048 ячеек. Каждая ячейка занимает 16 разрядов. Для обращения к памяти существует два регистра: 11-разрядный регистр адреса (AR - Address Register), в который нужно поместить адрес прежде чем обратиться к памяти; 16-разрядный регистр данных (DR - Data Register), который предназначен для чтения или записи данных в/из ячеек памяти. Чтение данных и запись данных реализуется по шинам, которые подключаются к ячейке памяти.
11-разрядный счетчик команд (IP - Instruction Pointer). Хранит в себе адрес следующей исполняемой команды.
Арифметико-логическое устройство или АЛУ (ALU - Arithmetic-n-Logic Unit) может выполнять несколько операций: сложение, логическое умножение, инверсия и прибавление единицы. При операций «сложение» возможен выход за пределы разрядной сетки и формирование битов переполнения и переноса. Выход из АЛУ через коммутатор подключается к шине, по которой информация может быть передана в любой другой регистр БЭВМ.
Буферный регистр (BR - Buffer Register) это 16-разрядный регистр, который используется для организации промежуточного хранения данных во время работы.
Регистр команд (CR Command Register) используется для хранения кода команды и декодирования операций, происходящих во время работы.
Аккумулятор (AC - Accumulator). БЭВМ относится к ЭВМ, которые называются ЭВМ аккумуляторного типа, где все вычисления с данными производятся через этот регистр.
Указатель стека (SP - Stack Pointer), как и IP и AR 11-ти разрядный, и всегда указывает на вершину стека.
16-разрядный клавишный регистр (IR Input Register) находится в составе пульта оператора ЭВМ и предназначен для ввода адреса программы, кодов программы и данных, запуска программы на выполнение и управления режимами работы БЭВМ.
16-ти разрядный регистр состояния (PS - Program State) хранит биты управляющие работой БЭВМ (работа, прерывание и пр.) и признаки результата.

20) Устройство Управления

Устройство управления разработано в виде микропрограммного устройства управления (МПУ, Microprogram Control Unit) - простейшего компьютера, программа которого непосредственно состоит из микроопераций - т. е. по-тактного изменения значений вентилей БЭВМ, которые задают атомарные операции: вычисления в АЛУ, пересылки данных между регистрами и простейшие проверки. Код программы для МПУ называется микрокодом. МПУ выполняет все машинные команды БЭВМ. Исполнение в МПУ машинной команды называется циклом команды. Цикл команды логически разбит на пять циклов:
Цикл выборки команды. Осуществляет загрузку исполняемой команды в регистр команды и частичное ее декодирование. Выполнятся для каждой исполняемой команды.
Цикл выборки адреса. Предназначен для обработки адресных команд и выборки адреса операнда с учетом режимов адресации.
Цикл выборки операнда. Для тех команд, где это необходимо, размещает в DR второй операнд команды.
Цикл исполнения. Производится исполнение команды.
Цикл прерывания. Цикл выполняется в том случае, если разрешены прерывания.
Для обеспечения работы оператора БЭВМ в ней предусмотрена микропрограммная реализация циклов пультовых операций:
Ввод адреса - адрес из клавишного регистра вводится в счетчик команд.
Чтение — информация по адресу в IP читается из памяти в DR, IP увеличивается на единицу.
Запись - информация из IR записывается в память по адресу в IP, IP увеличивается на единицу. Используется для ввода программы и данных в режиме оператора.
Пуск - осуществляет сброс состояния БЭВМ и переход к выполнению программы.
На панели оператора, кроме того, расположены другие органы управления: переключатель «Работа/Останов», который вызывает останов программы после каждой команды; переключатель «Такт», который может выполнить микрокод по одному такту, кнопка «Продолжение» возобновляющая работу остановленной БЭВМ.