Структурная и функциональная организация ЭВМ (Computer Organization and Design)

БГУИР
кафедра ЭВМ

доцент Воронов Александр Анатольевич
т. 217-74-02, sash_v_oo@mail.ru

Лекция 2
«Введение»

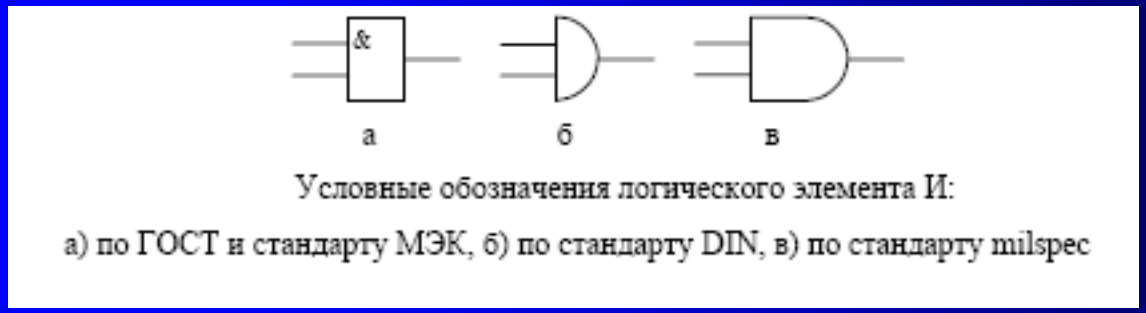
2018

План лекции

1. Обозначения элементной базы
2. Определения
3. Уровни абстракции ЭВМ
4. Концепция фон Неймана
5. Принципы организации ЭВМ по фон Нейману
6. Вопросы к лекции

Обозначения элементной базы

Советское и зарубежное обозначение элементной базы:



Условные обозначения логического элемента И:

а) по ГОСТ и стандарту МЭК, б) по стандарту DIN, в) по стандарту milspec

- И
- AND

MilSpec - военный стандарт США,
IEC 60617-12 - стандарт международной электротехнической комиссии.
ANSI/IEEE Std 91-1984 и последующий ANSI/IEEE Std 91a-1991

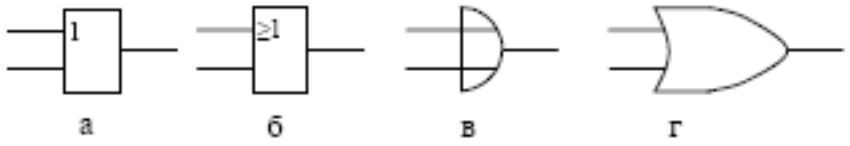
Условные обозначения логического элемента И-НЕ:

а) по ГОСТ и стандарту МЭК, б) по стандарту DIN, в) по стандарту milspec

- И-НЕ
- NAND

Обозначения элементной базы

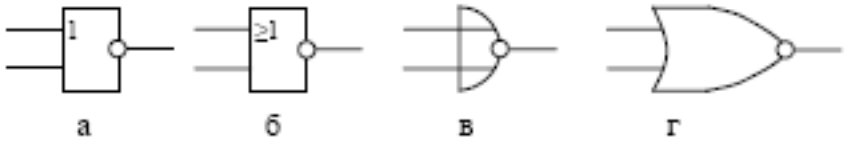
Советское и зарубежное обозначение элементной базы:



Условные обозначения логического элемента ИЛИ:

а) по ГОСТ, б) по стандарту МЭК, в) по стандарту DIN, г) по стандарту milspec

- ИЛИ
- OR



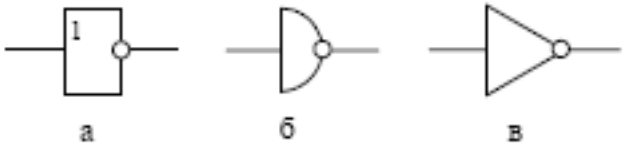
Условные обозначения логического элемента ИЛИ-НЕ:

а) по ГОСТ, б) по стандарту МЭК, в) по стандарту DIN, г) по стандарту milspec

- ИЛИ-НЕ
- NOR

Обозначения элементной базы

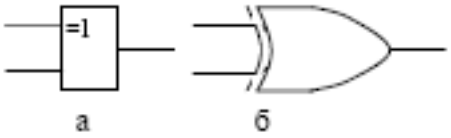
Советское и зарубежное обозначение элементной базы:



Условные обозначения логического элемента НЕ:

а) по ГОСТ и стандарту МЭК, б) по стандарту DIN, в) по стандарту milspec

- НЕ
- NOT



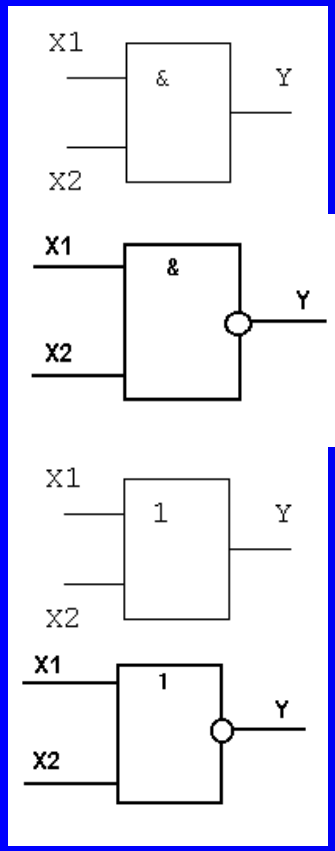
Условные обозначения логического элемента ИСКЛЮЧАЮЩЕЕ ИЛИ:

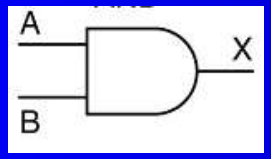
а) по ГОСТ и по стандарту МЭК, б) по стандарту milspec

- XOR

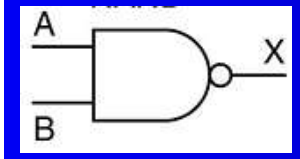
Обозначения элементной базы

Советское и зарубежное обозначение элементной базы:

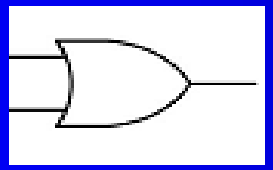




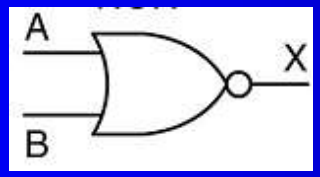
❑ И
❑ AND



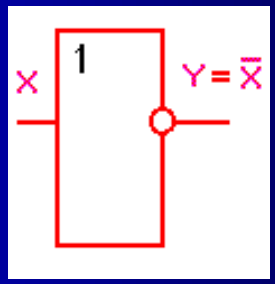
❑ И-НЕ
❑ NAND

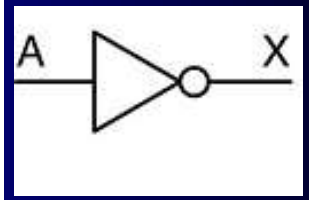


❑ ИЛИ
❑ OR

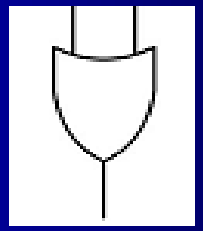


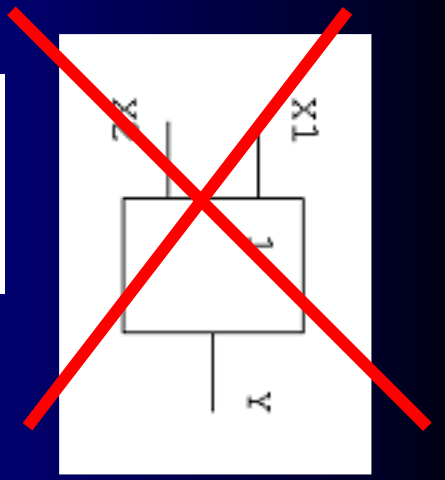
❑ ИЛИ-НЕ
❑ NOR





❑ НЕ ❑ NOT





Алгоритмы. Программы. Команды.

«Алгоритм – конечный набор предписаний, определяющий решение задачи посредством конечного количества операций» (ISO 2382/1-84)

Доп. определения:

– способ преобразования информации, задаваемый с помощью конечной системы правил.

– совокупность правил, определяющих эффективную процедуру решения любой задачи из некоторого заданного класса задач.

– точно определённое правило действий, для которого задано указание, как и в какой последовательности это правило следует применять к исходным данным задачи, чтобы получить её решение.

Алгоритмы. Программы. Команды.

Свойства алгоритма:

- Дискретность
  - действия над дискретной информацией,
  - действия сами дискретны
- Определённость
  - в алгоритме указано всё, что необходимо
  - ни одно из действий не должно трактоваться двояко
- Массовость
  - применимость алгоритма к множеству значений исходных данных
- Результативность
  - возможность получения результата за конечное число шагов

Алгоритмы. Программы. Команды.

Свойства алгоритмов позволяют осуществлять их выполнение на вычислительной машине (ВМ). Процесс, порождаемый алгоритмом – *«вычислительный процесс»*.

Основа архитектуры современных ВМ – представление алгоритма решения задачи в виде программы последовательных вычислений.

« Программа для ВМ –упорядоченная последовательность команд, подлежащая обработке». (ISO 2382/1-84)

Алгоритмы. Программы. Команды.

Вычислительная машина – это:

- «устройство, которое принимает данные, обрабатывает их в соответствии с программой, генерирует результаты и обычно состоит из ...»
- «функциональный блок, способный выполнять вычисления (ариф. и логич.) без участия человека»
- «устройство, способное:
  - хранить программу (-мы) и информацию, необх. для её выполнения,
  - быть свободно перепрограммируемым,
  - выполнять арифм. вычисления, задаваемые пользователем,
  - выполнять без вмешательства человека программу обработки, требующую изменения действий путём принятия логических решений в процессе обработки.»

Дилемма

Вычислительная машина намного проще человека

(на данный момент).

Алгоритм решения задачи для человека ≠ алгоритму решения задачи для ВМ

(по уровню детализации).

Электронные схемы могут распознавать и выполнять ограниченный набор простых команд. Все программы перед выполнением должны быть превращены в последовательность таких команд, которые обычно не сложнее, чем:

- сложить 2 числа
- проверить, не является ли число нулём
- скопировать данные из одной части памяти ВМ в другую

Дилемма

Набор примитивных команд в совокупности составляют *машинный язык*.

Чем проще команды – тем проще ВМ, дешевле электроника и т.п.

Соответственно, большинство машинных языков очень примитивны и использовать их трудно и утомительно.

Выход: построение ряда уровней абстракций, каждая из которых надстраивается над абстракцией более низкого уровня.

Языки, уровни и виртуальные машины
Languages, Levels, Virtual Machines

2 способа решения – **трансляция** и **интерпретация**.

Оба преследуют одну цель - разработка новых команд, которые более удобны для человека, чем встроенные машинные команды.

- Машинные команды- язык L0. Новые команды формируют язык L1.
- Трансляция и интерпретация сходны (по результату).

•Различие - хранение всей программы L0 (из L1) в памяти ВМ и затем выполнение, во втором случае - каждая команда программы на L1 перекодируется в L0 и сразу же выполняется.

Языки, уровни и виртуальные машины
Languages, Levels, Virtual Machines

- **Виртуальная машина** M1 - машинный язык L1, а машина (физ. или вирт) с языком L0 — M0.
- Должна быть разница между M1 и M0, иначе нет смысла.
- Стоимость или сложность M1 виртуальной машины - не ограничение. Программы можно писать и интерпретировать (или трансл.) на нижний уровень.

Языки, уровни и виртуальные машины
Languages, Levels, Virtual Machines

- То есть можно писать программы для виртуальных машин, как будто они действительно существуют.
- Д. б. не сильный разрыв между L0 и L1 не должны сильно различаться...
- Очевидное решение этой проблемы — создание еще одного набора команд, которые в большей степени ориентированы на человека и в меньшей степени на компьютер, чем L1. Этот третий набор команд также формирует язык, который можно назвать L2, а соответствующую виртуальную машину — M2.

Языки, уровни и виртуальные машины
Languages, Levels, Virtual Machines

- Верхний уровень L2 – для программиста, все нижние – по цепочке…
- Изобретение целого ряда языков, каждый из которых более удобен для человека, чем предыдущий, может продолжаться до тех пор, пока мы не дойдем до подходящего нам языка.
- Каждый такой язык использует своего предшественника как основу, поэтому мы можем рассматривать компьютер в виде ряда уровней.
- Язык, находящийся в самом низу иерархической структуры — самый примитивный, а находящийся на самом верху — самый сложный.

Языки, уровни и виртуальные машины
Languages, Levels, Virtual Machines

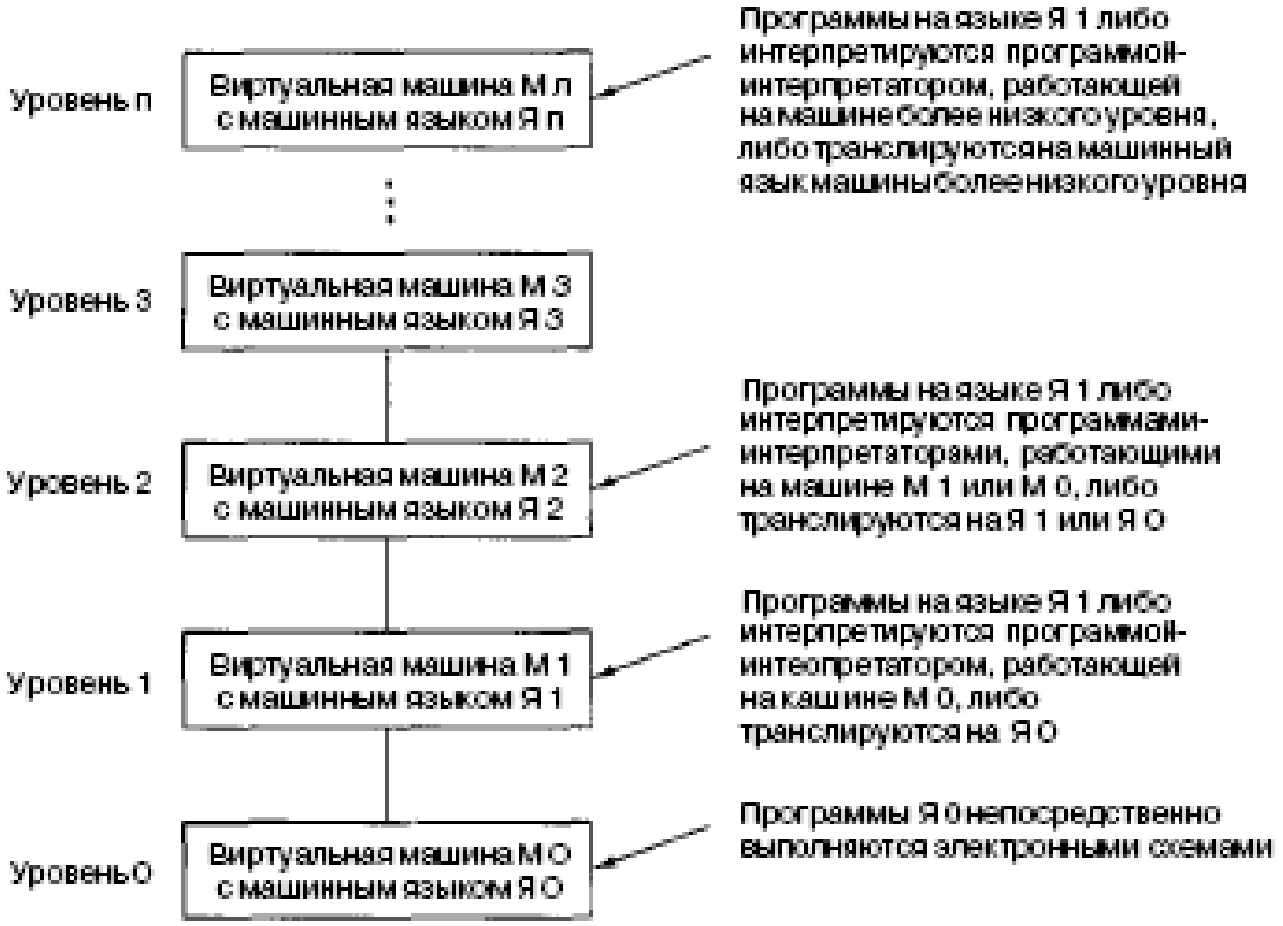
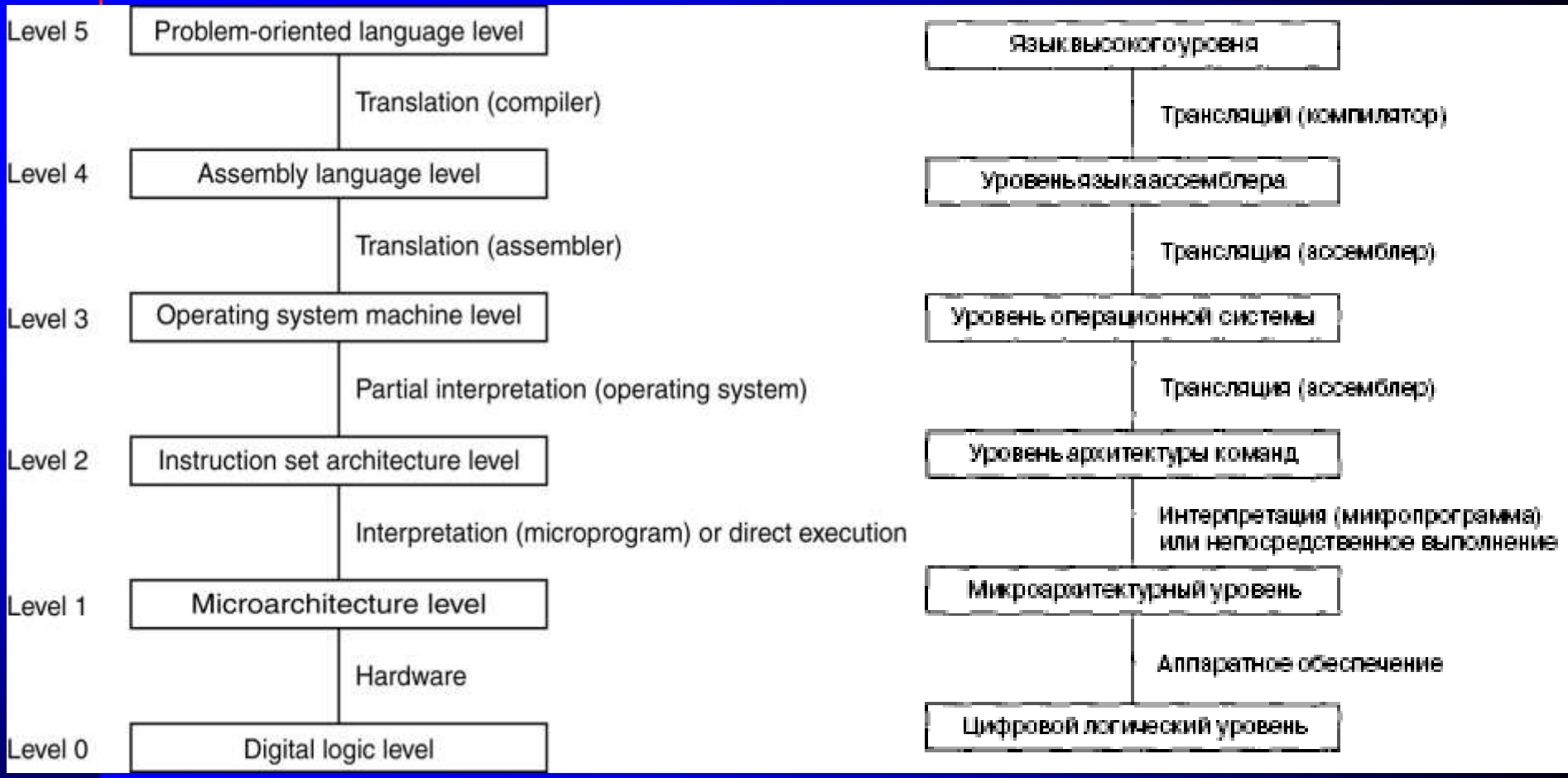


Рис. Многоуровневая машина

A multilevel machine

Современные многоуровневые машины
Contemporary Multilevel Machines



A six-level computer.

The support method for each level is indicated below it .

Современные многоуровневые машины
Contemporary Multilevel Machines

- -1 уровень (не показан) - **уровень физических устройств.** На нём находятся транзисторы, которые являются примитивами для разработчиков компьютеров. Объяснять, как работают транзисторы,— задача физики.

- 0 уровень - самый нижний - **цифровой логический уровень содержит** объекты, называемые **вентилями.** Вентиль вычисляет простые функции двоичных сигналов, такие как И или ИЛИ. Каждый вентиль формируется из нескольких транзисторов. Несколько вентилей формируют 1 бит памяти. Биты памяти, объединенные в группы, например, по 16,32 или 64, формируют регистры.

Современные многоуровневые машины Contemporary Multilevel Machines

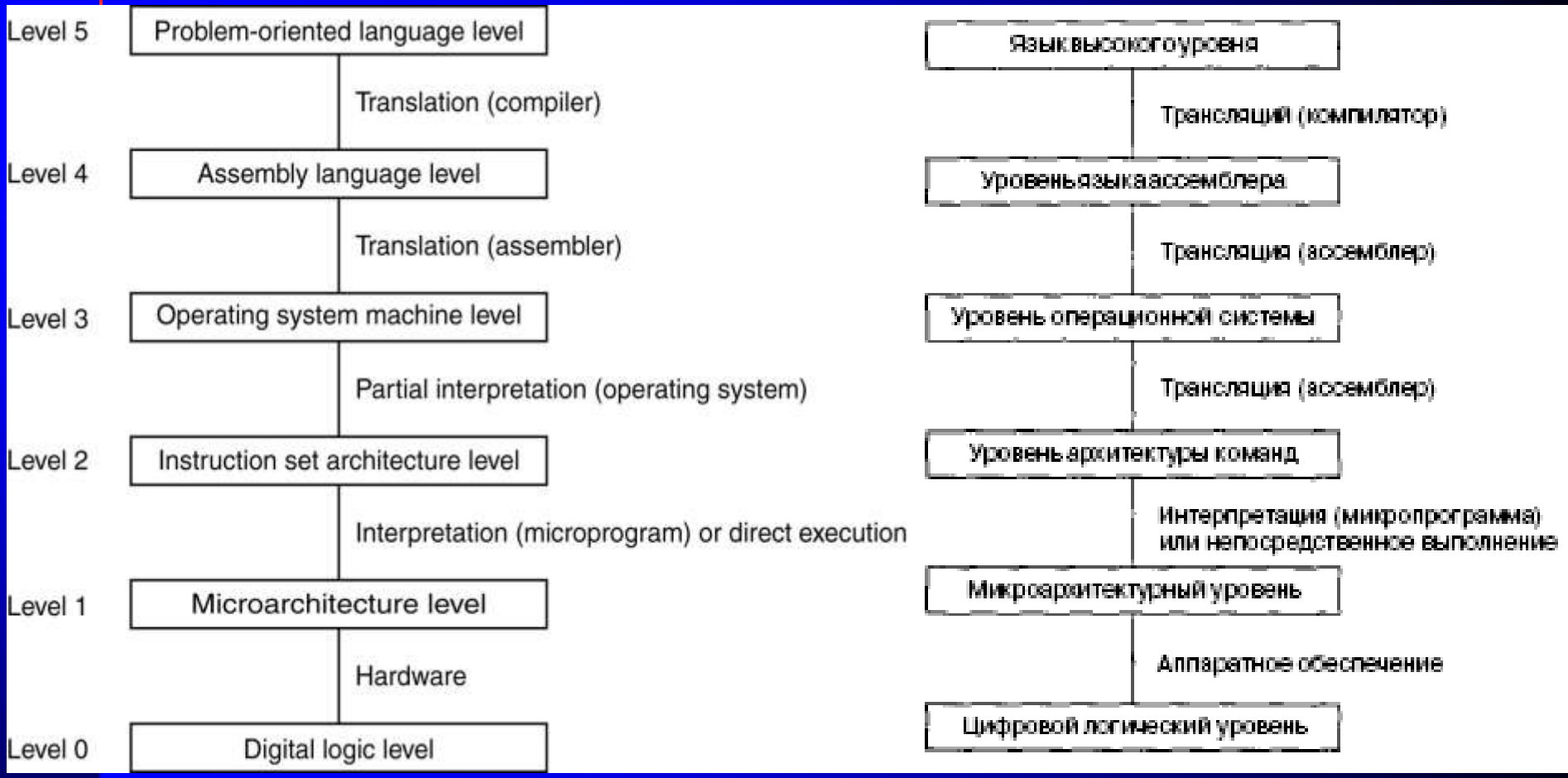
- 1 уровень - **микроархитектурный уровень.** На этом уровне можно видеть совокупности 8 или 32 регистров, которые формируют локальную память и схему, называемую **АЛУ (арифметико-логическое устройство).** АЛУ выполняет простые арифметические операции. Регистры вместе с АЛУ формируют **тракт данных,** по которому поступают данные. Основная операция тракта данных:
  - выбирается один или два регистра, АЛУ производит над ними какую-либо операцию, например сложения, а результат помещается в один из этих регистров.

- На некоторых машинах работа тракта данных контролируется особой программой, которая называется **микропрограммой.** На других машинах тракт данных контролируется аппаратными средствами.

Современные многоуровневые машины Contemporary Multilevel Machines

- На машинах, где тракт данных контролируется программным обеспечением, микропрограмма — это интерпретатор для команд на уровне 2.
- Микропрограмма вызывает команды из памяти и выполняет их одну за другой, используя при этом тракт данных. Например, для того чтобы выполнить команду ADD, эта команда вызывается из памяти, ее операнды помещаются в регистры, АЛУ вычисляет сумму, а затем результат переправляется обратно.
- На компьютере с аппаратным контролем тракта данных происходит такая же процедура, но при этом нет программы, которая контролирует интерпретацию команд уровня 2.

Современные многоуровневые машины
Contemporary Multilevel Machines



A six-level computer.

The support method for each level is indicated below it .

Современные многоуровневые машины Contemporary Multilevel Machines

- 2-й уровень - уровень архитектуры системы команд- Instruction Set Architecture (ISA).
- Каждый производитель семейств процессоров публикует руководство под названием «Руководство по машинному языку » или «Принципы работы компьютера Western Wombat Model 100X » и т.п.
- Такие руководства содержат информацию именно об этом уровне. Когда они описывают набор машинных команд, они в действительности описывают команды, которые выполняются микропрограммой-интерпретатором или аппаратным обеспечением. Если производитель поставляет два интерпретатора для одной машины, он должен издать два руководства по машинному языку, отдельно для каждого интерпретатора.

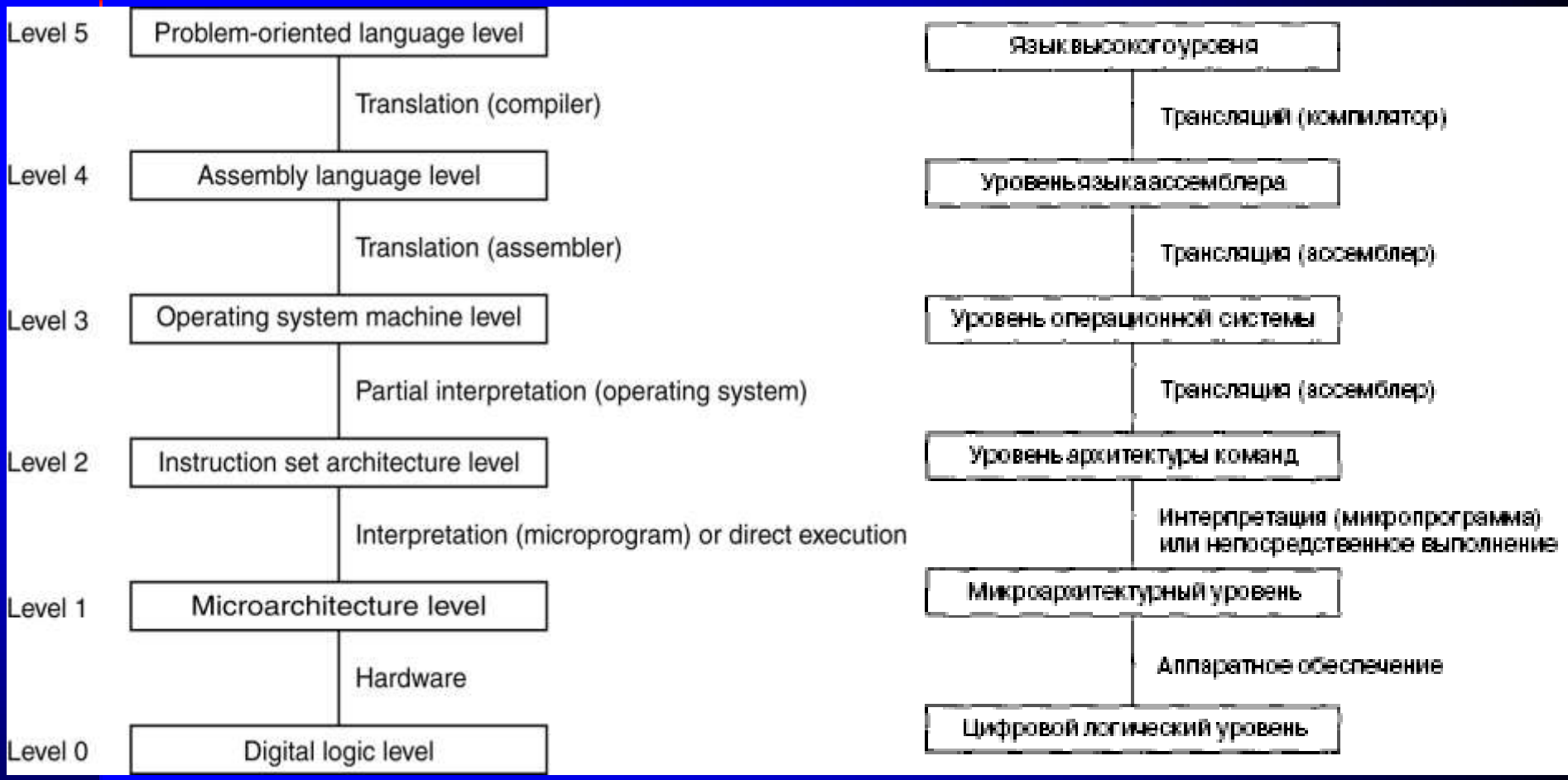
Современные многоуровневые машины Contemporary Multilevel Machines

- 3-й - уровень операционной системы - обычно гибридный. Большинство команд в его языке есть также и на уровне архитектуры системы команд (команды, имеющиеся на одном из уровней, вполне могут находиться на других уровнях). Особенности уровня: набор новых команд, другая организация памяти, способность выполнять две и более программ одновременно и т.п. При построении третьего уровня возможно больше вариантов, чем при построении первого и второго.

Современные многоуровневые машины Contemporary Multilevel Machines

- Новые средства, 3-го уровня, выполняются интерпретатором, написанном на языке второго уровня. Этот интерпретатор - операционная система. Команды 3-го уровня, идентичные командам 2-го уровня, выполняются микропрограммой или аппаратным обеспечением, но не ОС. Иными словами, одна часть команд третьего уровня интерпретируется ОС, а другая часть — микропрограммой.

Современные многоуровневые машины
Contemporary Multilevel Machines



A six-level computer.

The support method for each level is indicated below it .

Современные многоуровневые машины Contemporary Multilevel Machines

- Уровни с четвертого и выше предназначены для прикладных программистов, решающих конкретные задачи.
- Уровни 2 и 3 обычно интерпретируются, а уровни 4, 5 и выше обычно, хотя и не всегда, поддерживаются транслятором.
- Другое различие между уровнями 1,2,3 и уровнями 4,5 и выше — особенность языка. Машинные языки уровней 1,2 и 3 — цифровые. Начиная с четвертого уровня, языки содержат слова и сокращения, понятные человеку.

Современные многоуровневые машины Contemporary Multilevel Machines

- Четвертый уровень представляет собой символическую форму одного из языков более низкого уровня. На этом уровне можно писать программы в приемлемой для человека форме. Эти программы сначала транслируются на язык уровня 3, 2 или 1,а затем интерпретируются соответствующей виртуальной или фактически существующей машиной. Программа, которая выполняет трансляцию, называется **ассемблером.**

- Пятый уровень - языки прикладных программистов (ЯВУ). Программы, написанные на этих языках, обычно транслируются на уровень 3 или 4. Трансляторы, которые обрабатывают эти программы, называются компиляторами. Иногда также используется метод интерпретации. (например - Java).

Развитие многоуровневых машин
Evolution of Multilevel Machines

Уровень	Характеристика, примечание
Проблемно-ориентированные языки	Использование готовой прикладной программы (например, MS Word)
Процедурно-ориентированные языки	Программирование прикладной задачи
Уровень языка макроассемблера	Позволяет поименовать часто повторяющиеся фрагменты кода и в дальнейшем использовать это имя
Уровень функций ОС (BIOS)	Можем пользоваться готовыми типовыми подпрограммами для часто используемых или аппаратно зависимых действий
Уровень машинных команд	Если пишем свои аналоги системных функций
Микропрограммный уровень	Программно невидим, но знание его свойств позволяет улучшать эффективность кода
Уровень аппаратуры	Доступен во встроенных применениях

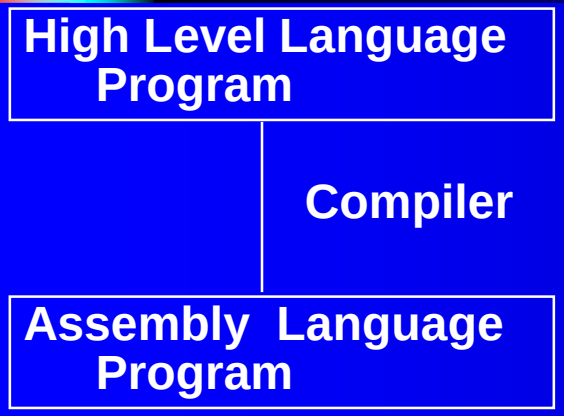
Вывод: компьютер проектируется как иерархическая структура уровней, каждый из которых надстраивается над предыдущим. Каждый уровень представляет собой определенную абстракцию с различными объектами и операциями.

How to Speak Computer

High Level Language Program

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

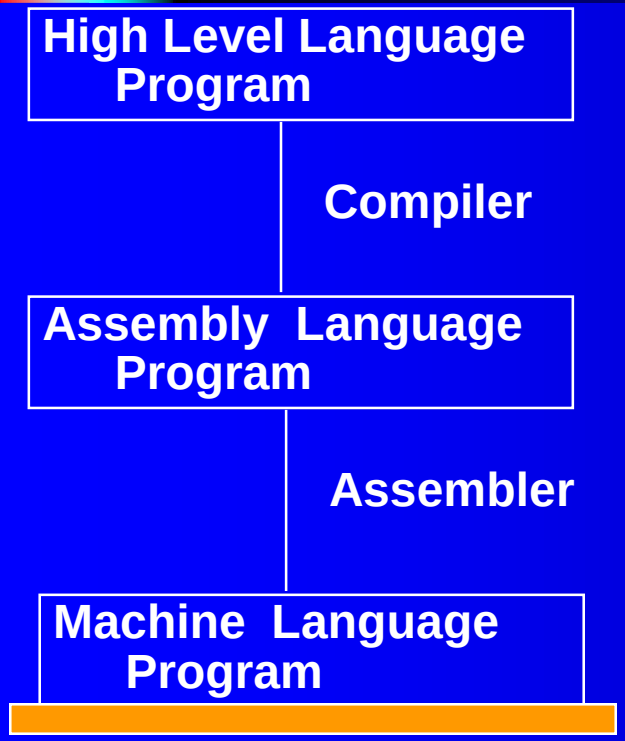
How to Speak Computer




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw  $15,   0($2)
lw  $16,   4($2)
sw $16,   0($2)
sw $15,   4($2)
```

How to Speak Computer

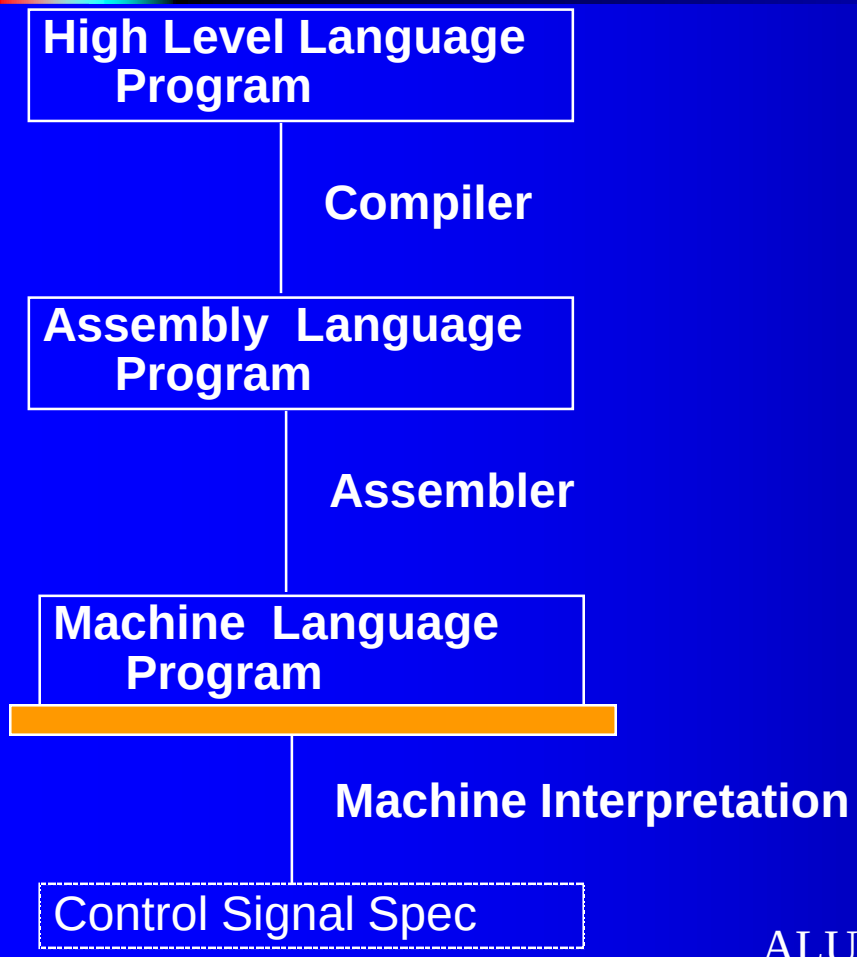



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw  $15,   0($2)
lw  $16,   4($2)
sw $16,   0($2)
sw $15,   4($2)
```

```
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
```

How to Speak Computer




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw  $15,   0($2)
lw  $16,   4($2)
sw $16,   0($2)
sw $15,   4($2)
```

```
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
```

ALUOP[0:3] <= InstReg[9:11] & MASK

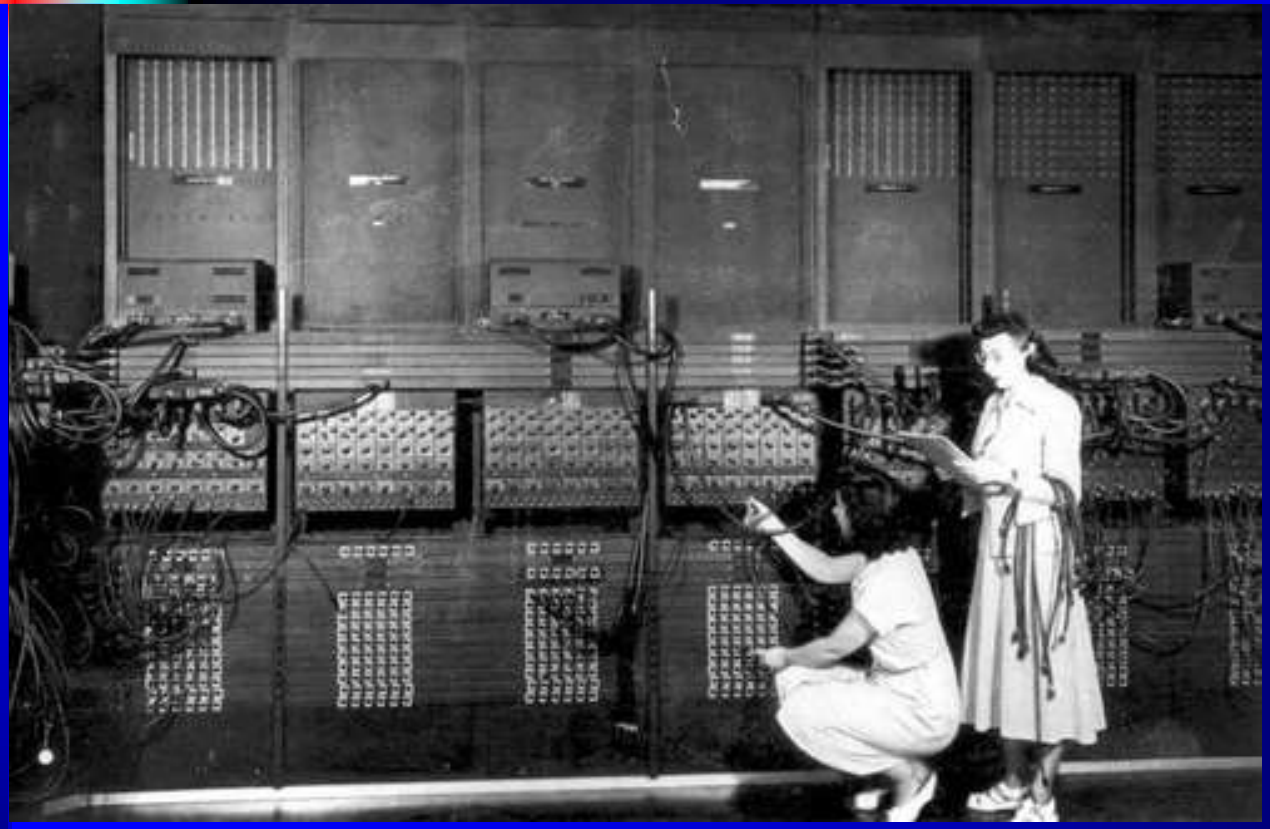
The Instruction Set Architecture

Архитектура системы команд – интерфейс между всем выполняемым на машине ПО и аппаратным обеспечением.

Развитие многоуровневых машин
Evolution of Multilevel Machines

- Изобретение микропрограммирования
- Изобретение ОС
- Перемещение функциональности системы на уровень микрокода
- Устранение микропрограммирования

Развитие многоуровневых машин
Evolution of Multilevel Machines



ENIAC,
14.02.1946

У первых цифровых компьютеров в 1940-х годах было только 2 уровня: уровень архитектуры набора команд, на котором осуществлялось программирование, и цифровой логический уровень, который выполнял программы. Схемы цифрового логического уровня были сложны для производства и понимания и ненадежны.

Развитие многоуровневых машин
Evolution of Multilevel Machines

Изобретение микропрограммирования

- В 1951 году Морис Уилкс, (Кембриджский университет), предложил идею разработки трехуровневого компьютера для того, чтобы упростить аппаратное обеспечение. Эта машина должна была иметь встроенный **неизменяемый** интерпретатор (микропрограмму), функция которого заключалась в выполнении программ посредством интерпретации.

Развитие многоуровневых машин
Evolution of Multilevel Machines

- Соответственно, так как аппаратное обеспечение стало вместо программ уровня архитектуры команд выполнять только микропрограммы с ограниченным набором команд, то потребовалось меньшее количество электронных схем. Поскольку электронные схемы были из эл. ламп, то данное изобретение сокращало количество ламп и увеличивало надежность ВМ.

- В 50-е годы было построено несколько трехуровневых машин. В 60-х годах число таких машин значительно увеличилось. К 70-м годам идея о том, что написанная программа сначала должна интерпретироваться микропрограммой, а не выполняться непосредственно электроникой, стала преобладающей.

Развитие многоуровневых машин
Evolution of Multilevel Machines

Изобретение ОС

- В 60-е годы - первая попытка сократить количество потерянного времени на загрузку, выполнение и выгрузку ОЗУ (в случае ошибки), автоматизировав работу оператора. Программа под названием **«операционная система »** теперь содержалась в компьютере все время.
- В дальнейшем ОС все больше и больше усложнялись. К уровню ISA добавлялись новые команды, приспособления и особенности, и в итоге сформировался новый уровень. Некоторые команды нового уровня были идентичны командам предыдущего, но другие (в частности команды ввода-вывода) полностью отличались. Они тогда назывались **макросами операционной системы** или **вызовами супервизора.** Сейчас обычно используется термин **«системный вызов ».**

Развитие многоуровневых машин
Evolution of Multilevel Machines

Перемещение функциональности системы на уровень микрокода

- С 1970 года, когда микропрограммирование стало обычным, у производителей появилась возможность вводить новые машинные команды путем расширения микропрограммы, то есть с помощью программирования. Это открытие привело к виртуальному взрыву в производстве программ машинных команд, поскольку производители начали конкурировать друг с другом, стараясь выпустить лучшие программы. Эти команды не представляли особой ценности, поскольку те же задачи можно было легко решить, используя уже существующие программы, но обычно они работали немного быстрее.

Развитие многоуровневых машин
Evolution of Multilevel Machines

- Например, команда INC (INCrement) при наличии общей команды сложения ADD. INC работала чуть быстрее, чем ADD, поэтому она была включена в набор команд.

Многие программы были добавлены в микропрограмму по той же причине:

1.Команды для умножения и деления целых чисел.

2.Команды для арифметических действий над числами с плавающей точкой.

3.Команды для вызова и прекращения действия процедур.

4.Команды для ускорения циклов.

5.Команды для работы со строкой символов.

Развитие многоуровневых машин Evolution of Multilevel Machines

- Как только производители поняли, что добавлять новые команды очень легко, они начали добавлять доп. тех. характеристики к микропрограмме:

- 1.Ускорение работы с массивами (индексная и косвенная адресация).
- 2.Перемещение программы из одного раздела памяти в другой после запуска программы (настройка).
- 3.Системы прерывания, которые дают сигнал процессору, как только закончена операция ввода или вывода
- 4.Способность приостановить одну программу и начать другую, используя небольшое число команд (переключение процесса).
- Доп. команды для ускорения работы ВМ.

Развитие многоуровневых машин
Evolution of Multilevel Machines

Устранение микропрограммирования

- В 60-х-70-х годах количество микропрограмм сильно увеличилось. Однако они работали все медленнее и медленнее, поскольку требовали большого объема памяти.
- Обнаружилось, что доля доп. команд, эквивалентных ЯВУ в общем объёме программ не превышает 10-20%, а для наиболее сложн. команд – даже 0.2%. В то же время объём аппаратных средств вырастает существенно, например ёмкость микропрограммной памяти увеличивается до 60%.

- В конце концов исследователи осознали, что с устранением микропрограммы резко сократится количество команд и компьютеры станут работать быстрее.

Вехи в развитии ЭВМ
Milestones in Computer Architecture

Year	Name	Made by	Comments
1834	Analytical Engine	Babbage	First attempt to build a digital computer
1936	Z1	Zuse	First working relay calculating machine
1943	COLOSSUS	British gov't	First electronic computer
1944	Mark I	Aiken	First American general-purpose computer
1946	ENIAC I	Eckert/Mauchley	Modern computer history starts here
1949	EDSAC	Wilkes	First stored-program computer
1951	Whirlwind I	M.I.T.	First real-time computer
1952	IAS	Von Neumann	Most current machines use this design
1960	PDP-1	DEC	First minicomputer (50 sold)
1961	1401	IBM	Enormously popular small business machine
1962	7094	IBM	Dominated scientific computing in the early 1960s
1963	B5000	Burroughs	First machine designed for a high-level language
1964	360	IBM	First product line designed as a family

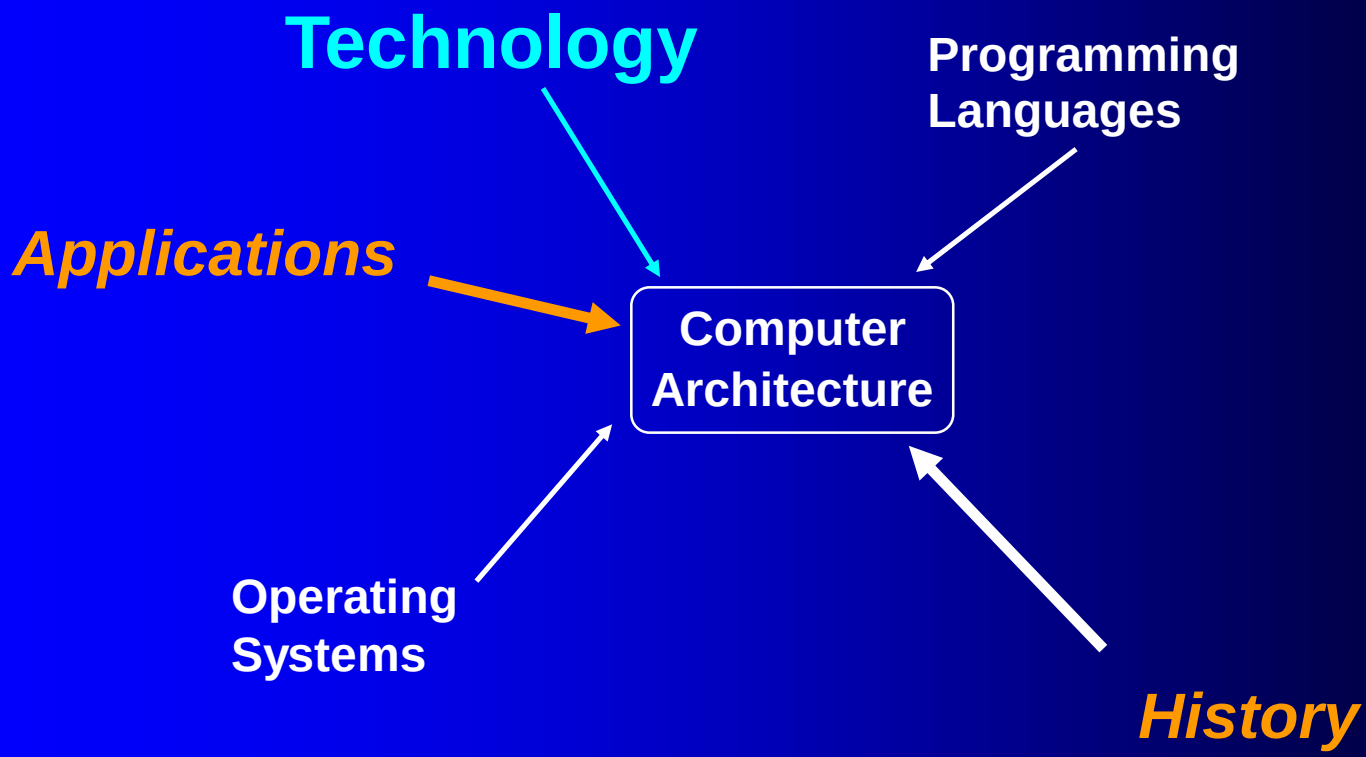
Вехи в развитии ЭВМ
Milestones in Computer Architecture

Year	Name	Made by	Comments
1965	PDP-8	DEC	First mass-market minicomputer (50,000 sold)
1970	PDP-11	DEC	Dominated minicomputers in the 1970s
1974	8080	Intel	First general-purpose 8-bit computer on a chip
1974	CRAY-1	Cray	First vector supercomputer
1978	VAX	DEC	First 32-bit superminicomputer
1981	IBM PC	IBM	Started the modern personal computer era
1981	Osborne-1	Osborne	First portable computer
1983	Lisa	Apple	First personal computer with a GUI
1985	386	Intel	First 32-bit ancestor of the Pentium line
1985	MIPS	MIPS	First commercial RISC machine
1987	SPARC	Sun	First SPARC-based RISC workstation
1990	RS6000	IBM	First superscalar machine
1992	Alpha	DEC	First 64-bit personal computer
1993	Newton	Apple	First palmtop computer

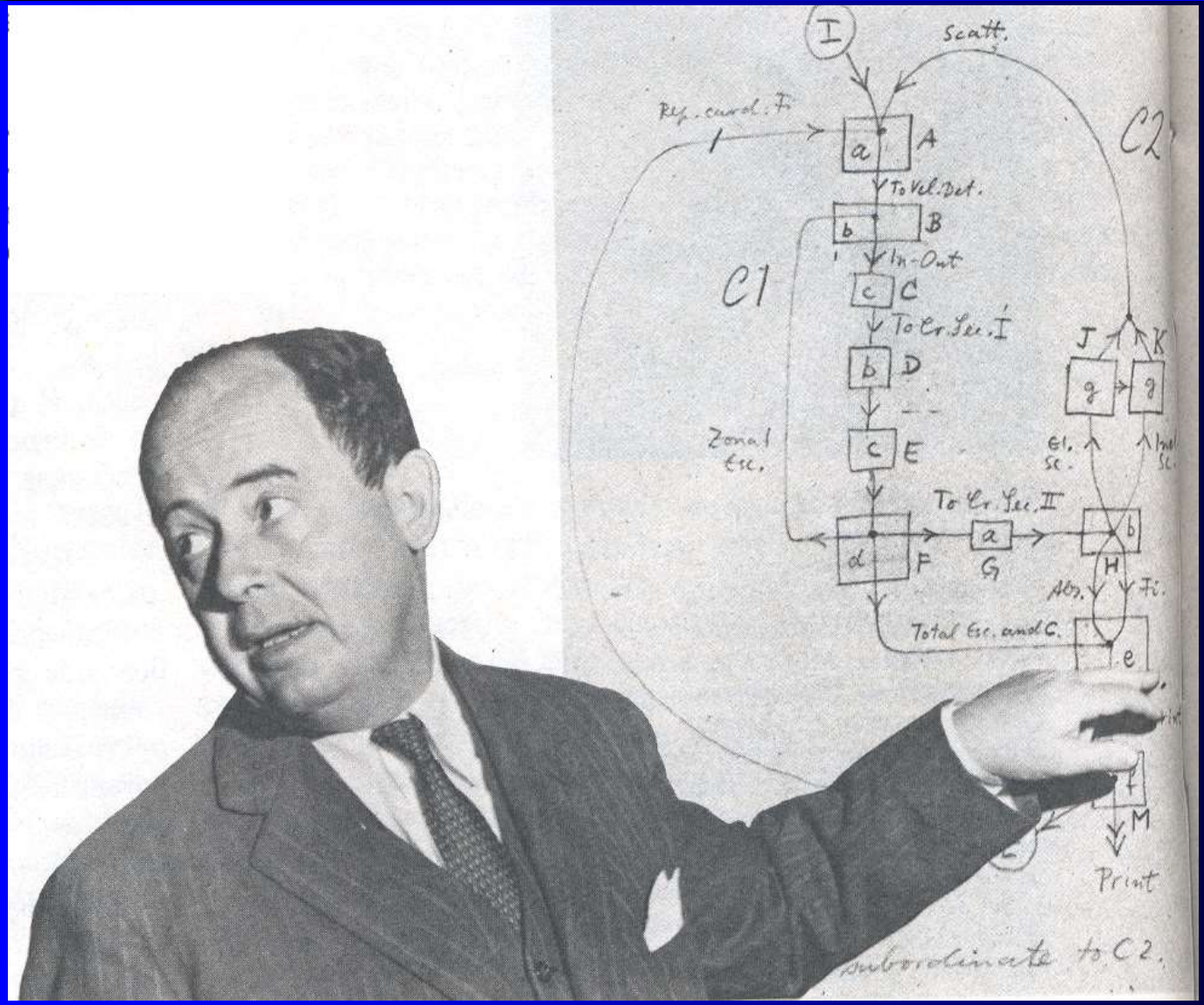
Computer Generations

- Zeroth Generation
  Mechanical Computers (1642 – 1945)
- First Generation
  Vacuum Tubes (1945 – 1955)
- Second Generation
  Transistors (1955 – 1965)
- Third Generation
  Integrated Circuits (1965 – 1980)
- Fourth Generation
  Very Large Scale Integration (1980 – ?)

Факторы, влияющие на СиФО ЭВМ
Forces on Computer Architecture



John von Neumann

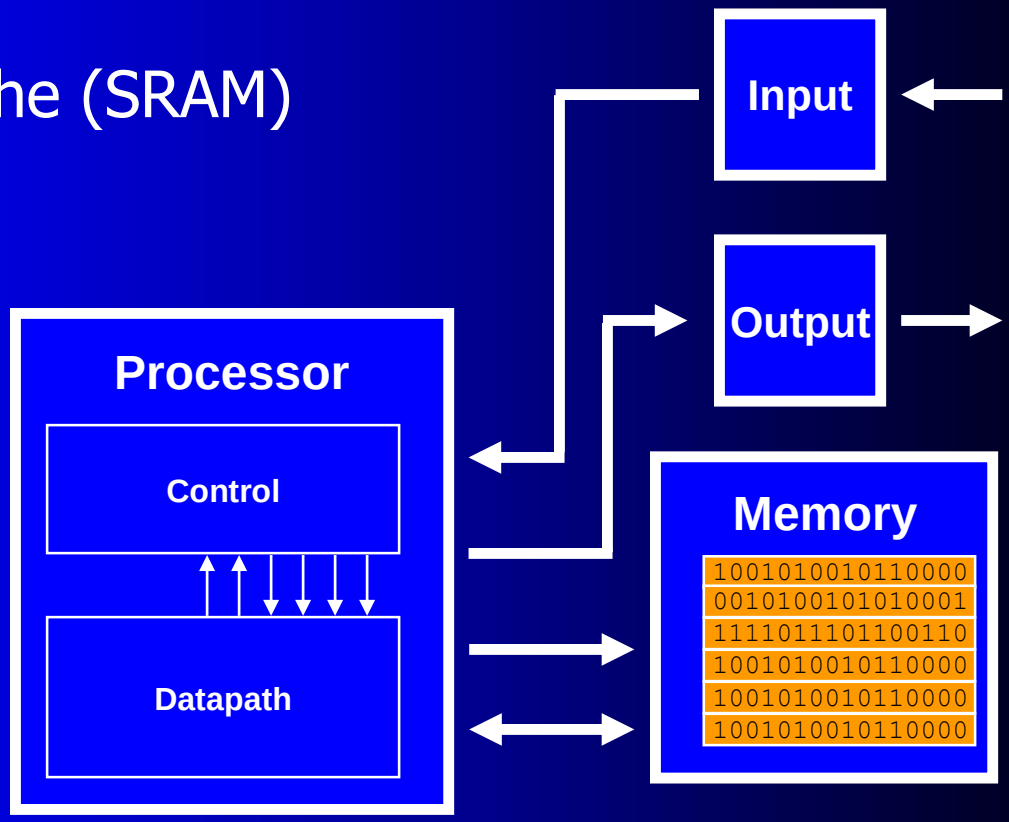


Пять классических компонентов любой ЭВМ

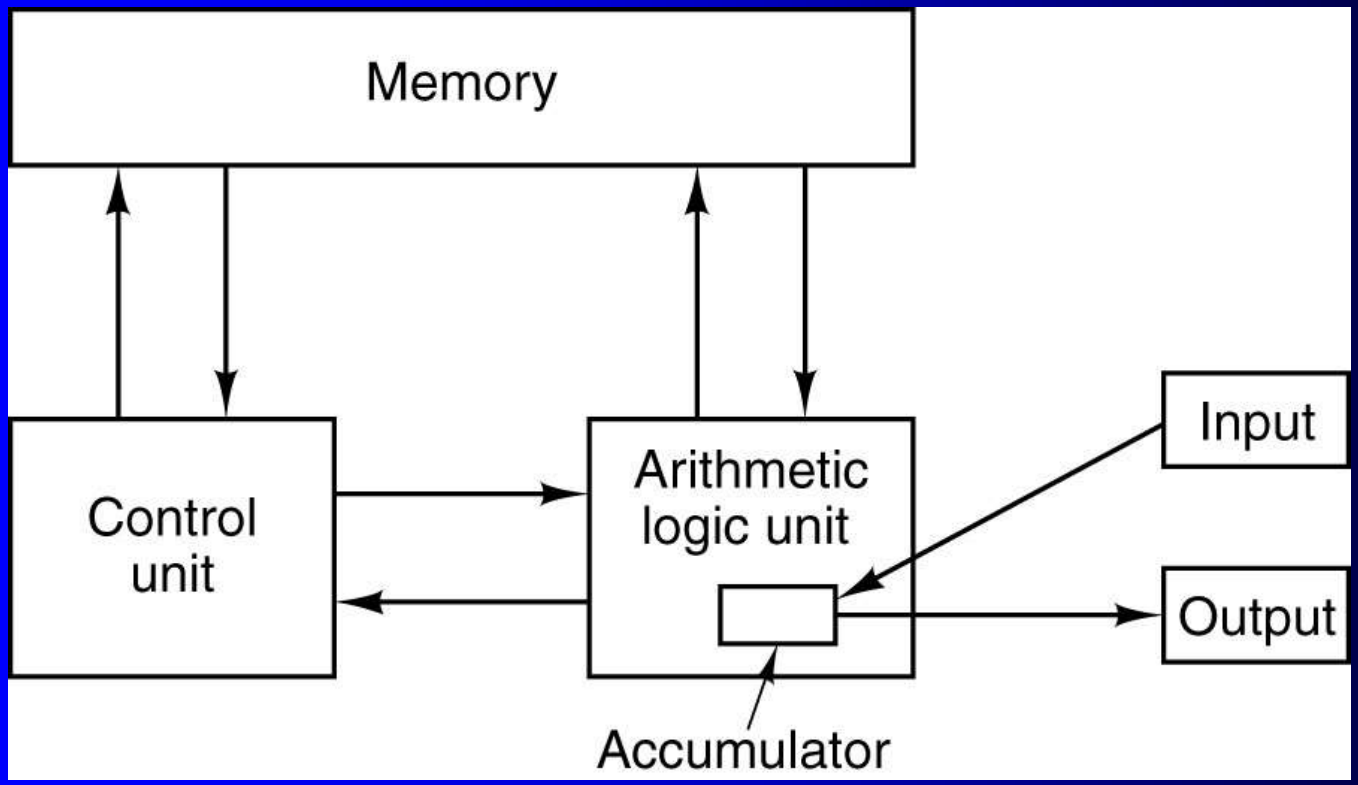
Input (mouse, keyboard, …)

- Output (display, printer, …)
- Memory
  - main (DRAM), cache (SRAM)
  - secondary (disk, CD, DVD, …)
- Datapath
- Control

Processor (CPU)

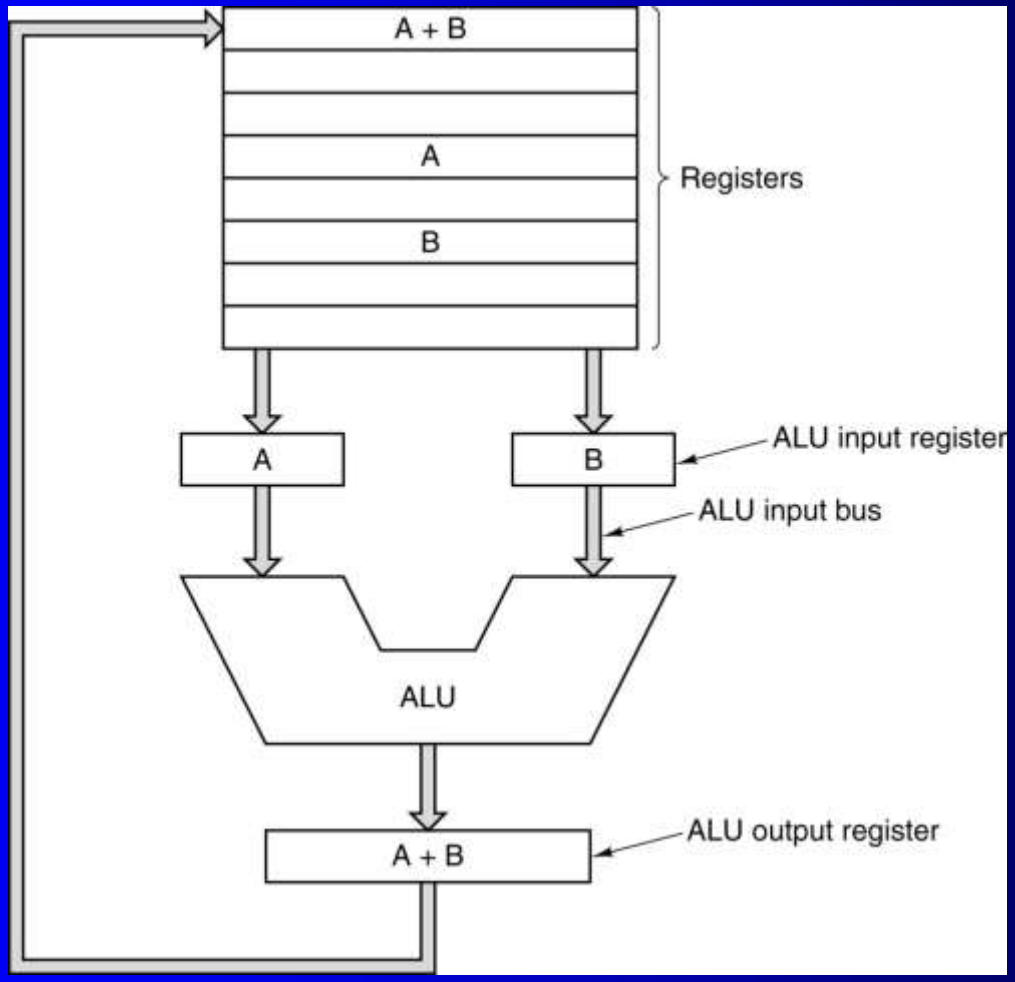


Von Neumann Machine



The original Von Neumann machine.

CPU Organization



Тракт данных (datapath) типичной ЭВМ, построенной по принципам фон Неймана (Von Neumann machine).

Принципы фон-Неймана (1945)

- принцип хранимой в памяти программы
- двоичное кодирование
- программное управление (последовательное выполнение команд)
- однородность памяти (безразличие к целевому назначению данных) Принстон
- линейное пространство памяти (адресность)

Принципы фон-Неймана (1945)

- Принцип хранимой в памяти программы
- ENIAC – (1946 -1955), программа задавалось схемой коммутации триггеров на 40 наборных полях.
- EDVAC – (1949-…) первая машина с хранимой в памяти программой

Принципы фон-Неймана (1945)

- Принцип двоичного кодирования

- Вся информация (команды, данные) – в бинарном виде
- Каждый тип информации имеет свой формат
- Последовательность битов, связанных одним смыслом, называется полем:
  - для данных – поле знака и поле значащих разрядов
  - для команды – поле кода операции и поле адресов

Принципы фон-Неймана (1945)

- Принцип программного управления

- Все вычисления алгоритма должны быть представлены в виде программы
- Программа состоит из последовательности управляющих слов - команд
- Команды хранятся в последовательных ячейках памяти и выполняются в естественной последовательности
- Последовательность может быть изменена – условно или безусловно

Принципы фон-Неймана (1945)

- Принцип однородности памяти

- Команды и данные хранятся в одной памяти и внешне не различимы
- Возможность производить над командами те же операции, что и над данными (модификация команд)
- Модификация команд современным программированием не приветствуется
- Самый полезный эффект данного принципа – команды одной программы могут быть получены как результат работы другой программы.

Возможность **трансляции** – главное следствие.

Принципы фон-Неймана (1945)

- Принцип однородности памяти

- Команды и данные хранятся в одной памяти и внешне не различимы – Принстонская архитектура
- Гарвардская архитектура – отдельная память команд и отдельная память данных.

- Узкое место принстонской архитектуры – пропускная способность тракта «процессор-память»

- В последнее время благодаря кэш-памяти всё чаще начинает использоваться гарвардская архитектура.

Принципы фон-Неймана (1945)

- Принцип адресности

- Память – набор пронумерованных ячеек
- В каждый момент доступна любая ячейка
- Двоичные коды данных и команд хранятся в единицах информации – словах
- Доступ к данным по номерам ячеек – адресам

Computer of the day

Историческая ретроспектива: последние минуты лекции

Computers:
4000 BC
to 1940's

Ввод: уши, глаза

Вывод: рот, карандаш

Регистры, кэш, ОЗУ, АЛУ: мозг

Хранилище: бумага

Datapath: глаза-мозг-карандаш

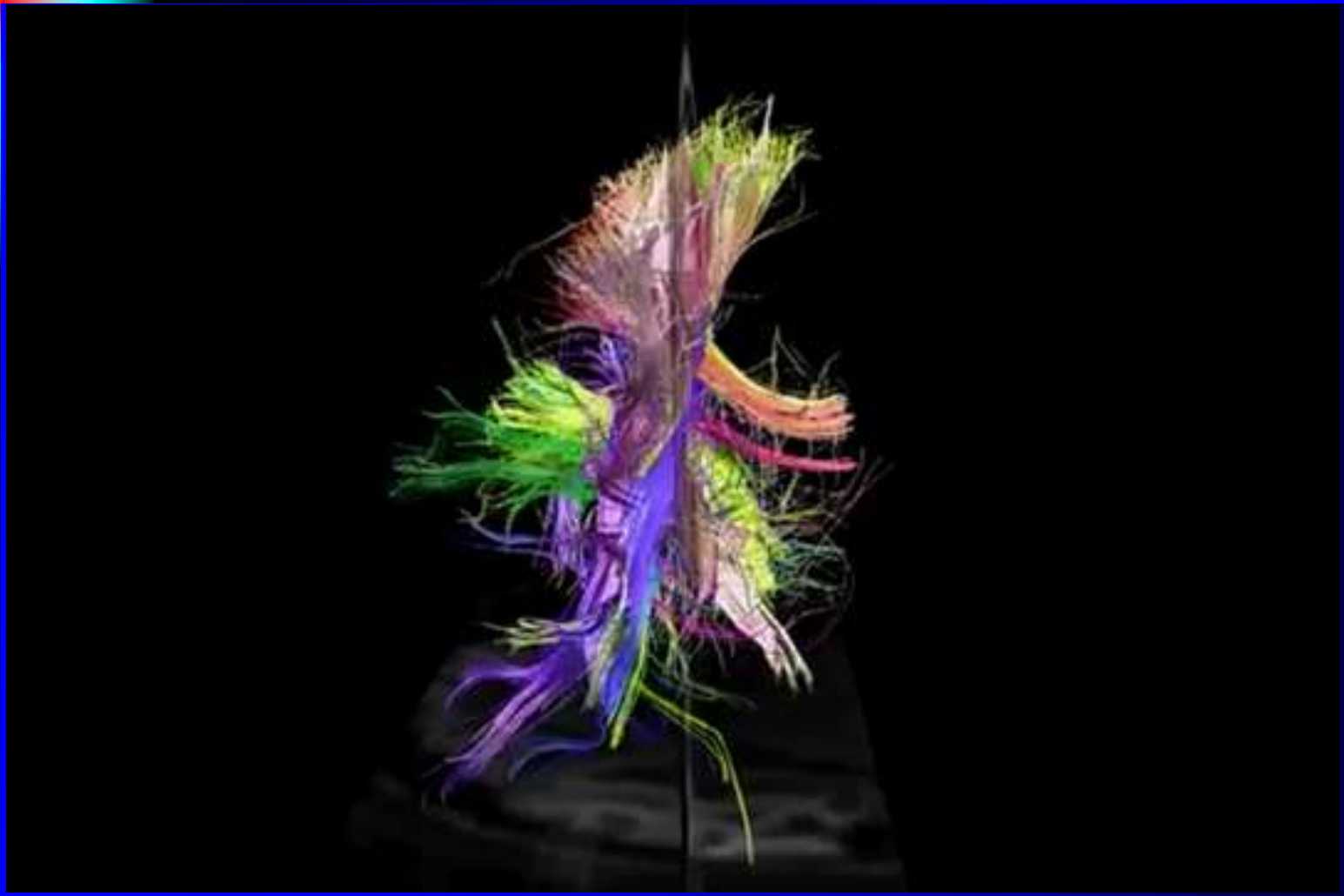
УУ: мозг

- Мозг человека имеет порядка 10 миллиардов нейронов.
- Каждый из них соединён с 100 000 соседями.
- Нейрон может "пульсировать" с частотой 1000 Гц.
  - Под «пульсацией» подразумевается решение о соединении с одним из 100 000 соседних нейронов (≈ транзисторов) за один «тактовый цикл» на соединение.

Структуры связей головного мозга человека

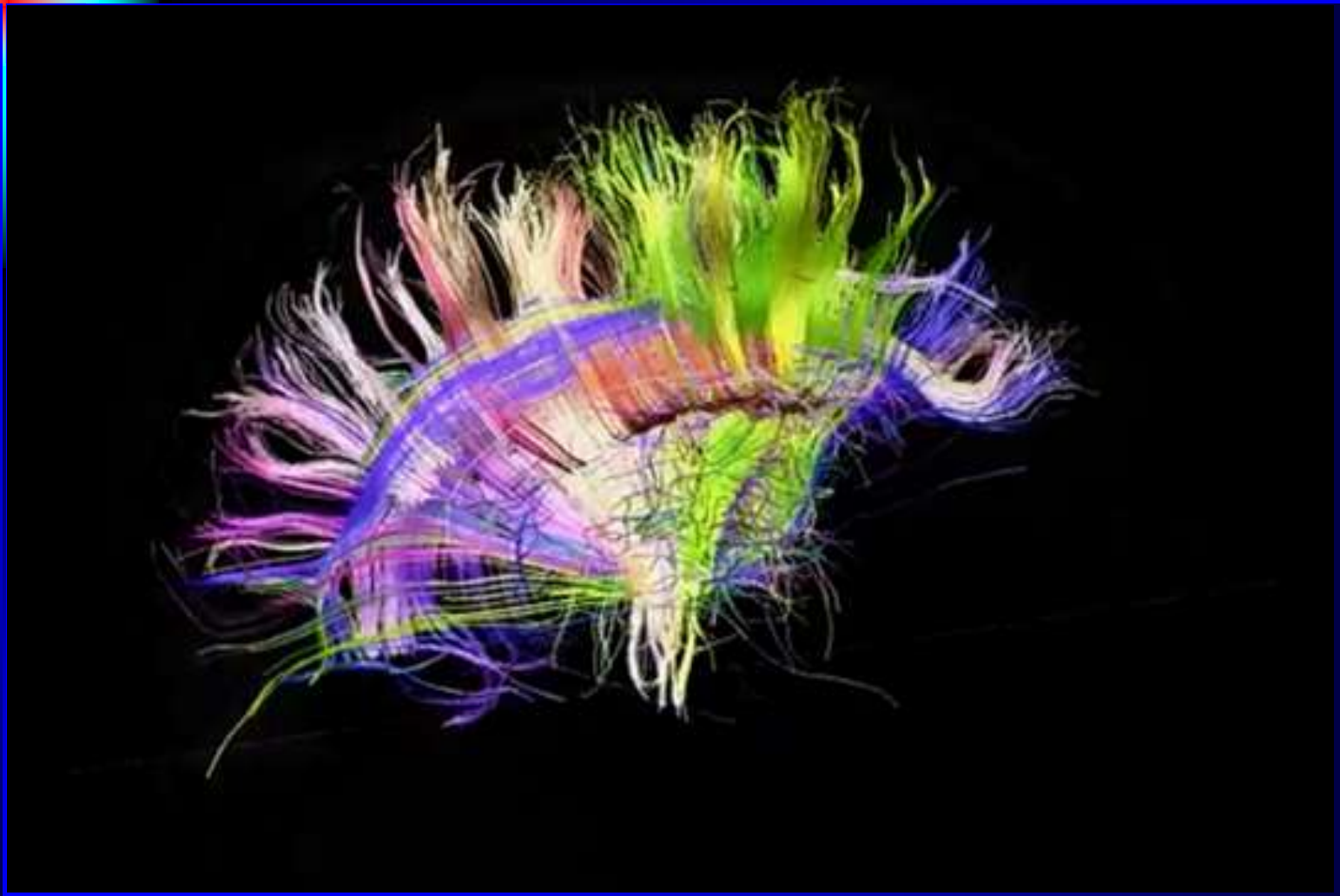
Вид со стороны левого полушария. (Van J. Wedeen, Douglas L. Rosene, Ruopeng Wang, Guangping Dai, Farzad Mortazavi, Patric Hagmann, Jon H. Kaas, Wen-Yih I. Tseng, 2012)

Структуры связей головного мозга человека



Вид сзади. (Van J. Wedeen, Douglas L. Rosene, Ruopeng Wang, Guangping Dai, Farzad Mortazavi, Patric Hagmann, Jon H. Kaas, Wen-Yih I. Tseng, 2012)

Структуры связей головного мозга человека



Вид справа. (Van J. Wedeen, Douglas L. Rosene, Ruopeng Wang, Guangping Dai, Farzad Mortazavi, Patric Hagmann, Jon H. Kaas, Wen-Yih I. Tseng, 2012)

Вопросы к лекции

1. Что такое «алгоритм»?
2. В чём состоит основная дилемма вычислительной техники?
3. В чём заключается смысл построения многоуровневых вычислительных машин?
4. Перечислите уровни абстракции современных ЭВМ?
5. В чём смысл микропрограммирования?
6. Какие факторы определяют развитие ЭВМ?
7. Пять основных блоков классической ЭВМ по фон Нейману.
8. Принципы организации вычислений по фон Нейману.

Литература

1. Танненбаум Э. Архитектура компьютера, 4-е изд., 2006 г. Стр. 18 – 40.
2. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем, Питер, 2007 г., Стр. 17-43.