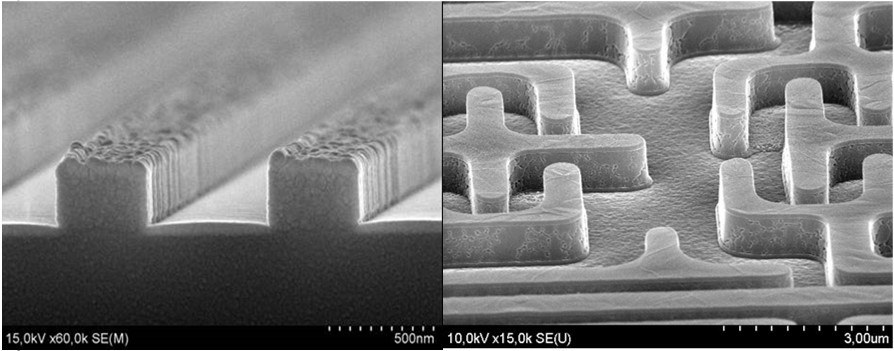
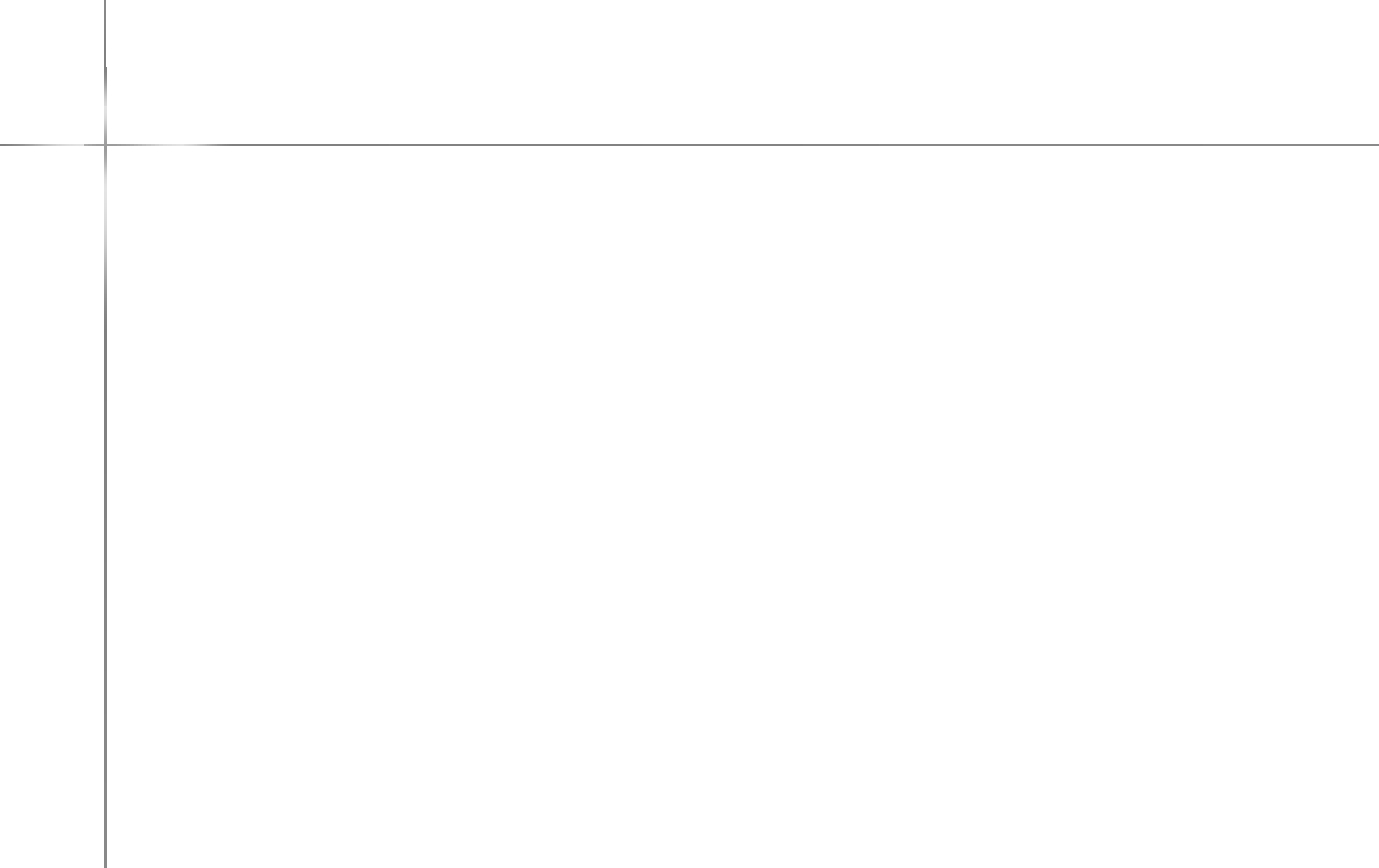


|  |  |
| --- | --- |
|  |  |
| Алгоритмические и логические основы цифровой  вычислительной техники  / Компьютерные системы и сети  БГТУ  кафедра ПИ | |
|  | доцент Самаль Дмитрий Иванович  [dmitry\_samal@mail.ru,](mailto:dmitry_samal@mail.ru)  a.408-1  Лекция 2  «Многоуровневые ЭВМ. Концепция фон Неймана» 2020 |

|  |  |
| --- | --- |
|  |  |
| План лекции   1. Уровни абстракции ЭВМ 2. Концепция фон Неймана 3. Принципы организации ЭВМ по фон Нейману | |
|  | 4. Вопросы к лекции  !**!** - обязательный вопрос к зачёту  ***Слайд*** 2 |

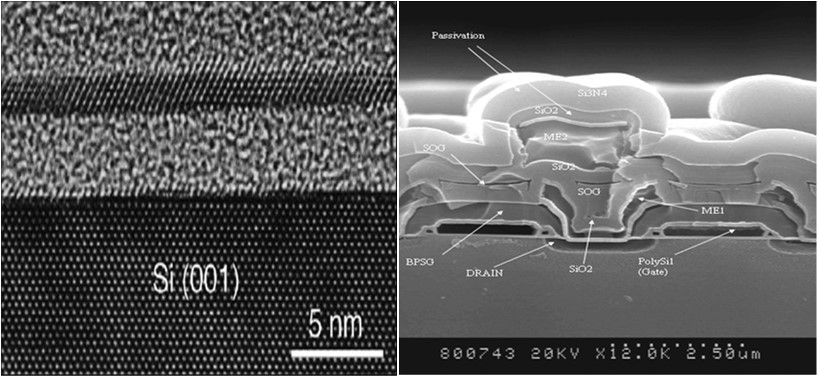
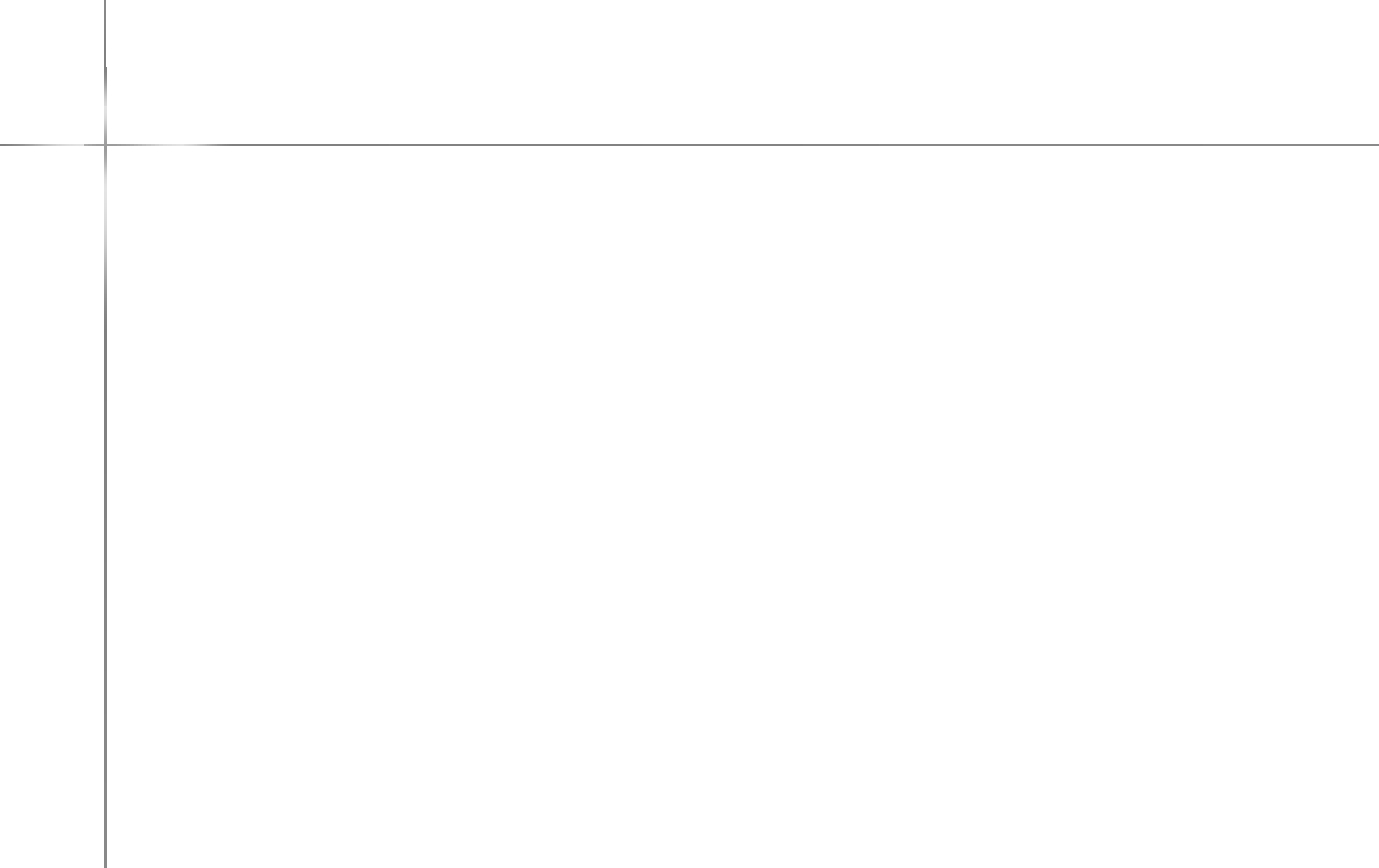
# 1 2

3 4



Транзисторный уровень

***3***



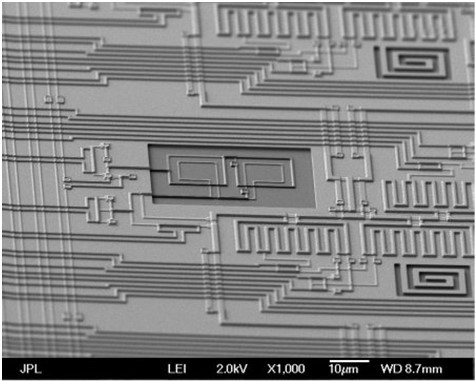
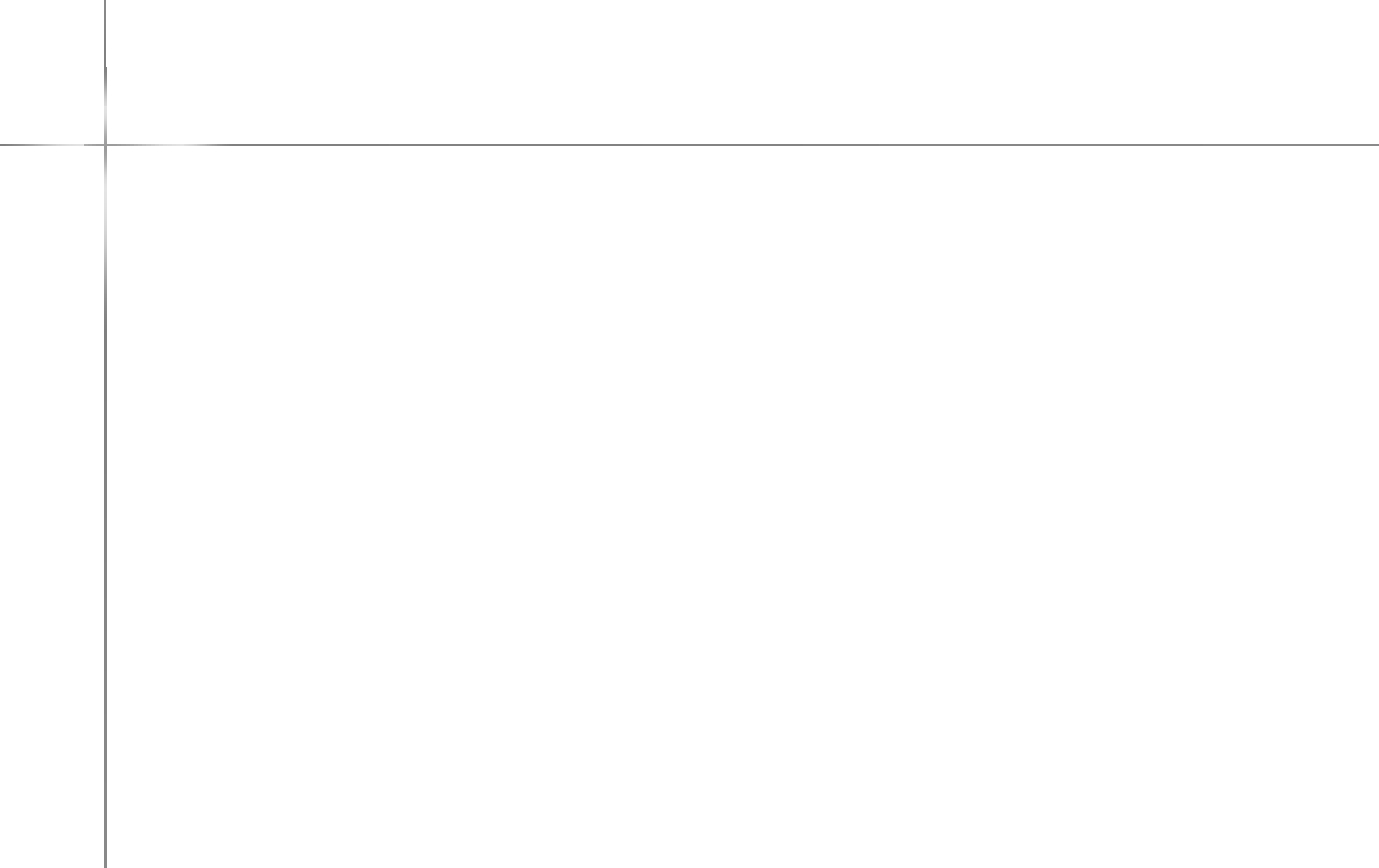
Транзисторный уровень

Топологический рисунок МДП- транзистора

***4***

А.С. Строгова, БГУИР, 2016

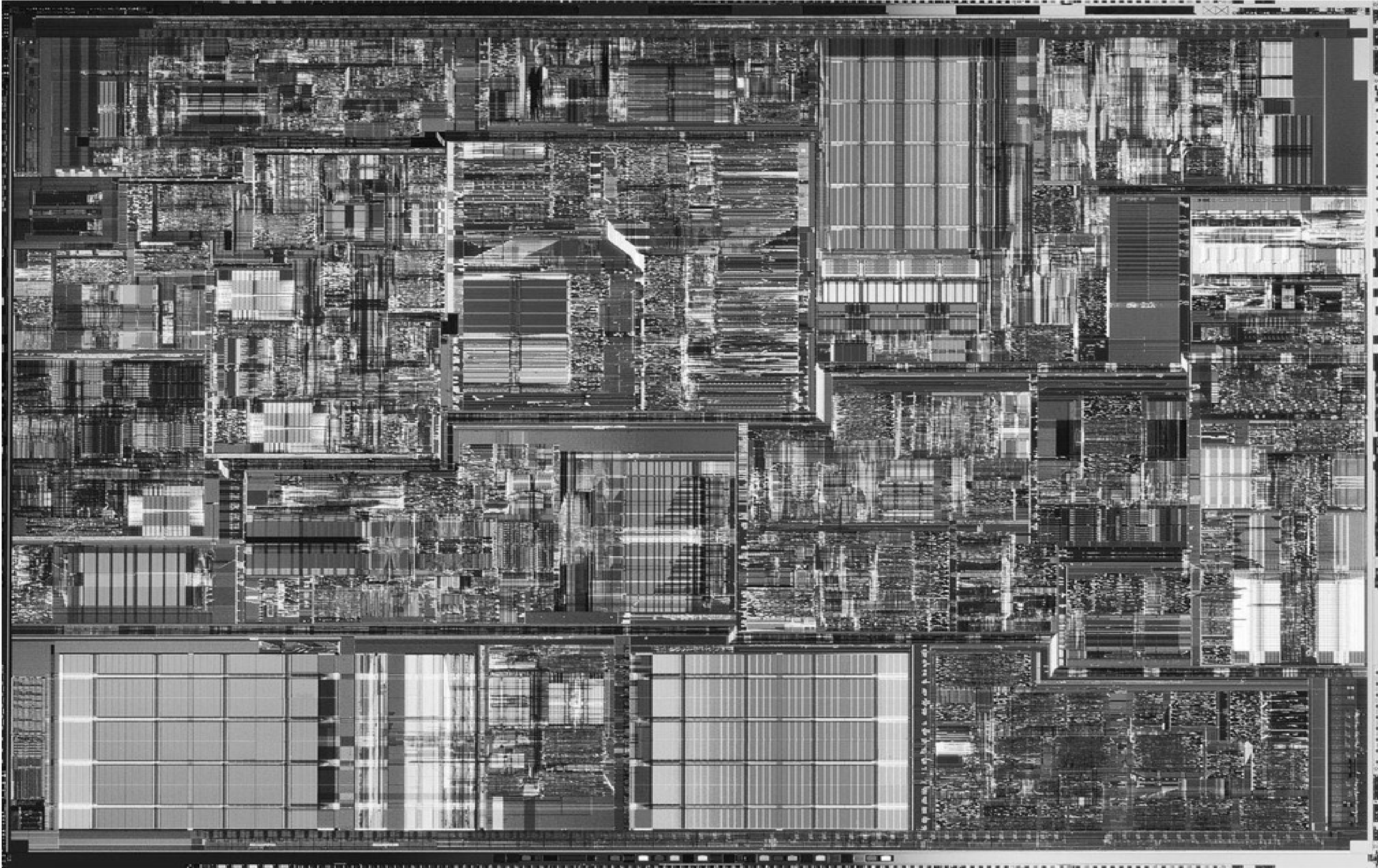
А.С. Строгова, БГУИР, 2016



Микроархитектурный уровень

Топологический части микроконтроллера

***5***



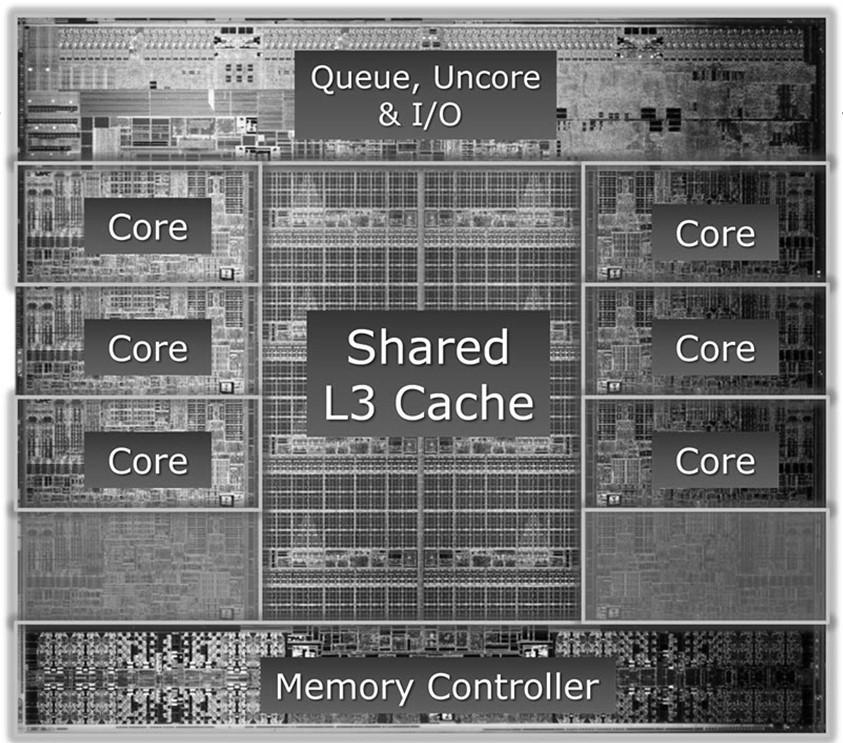
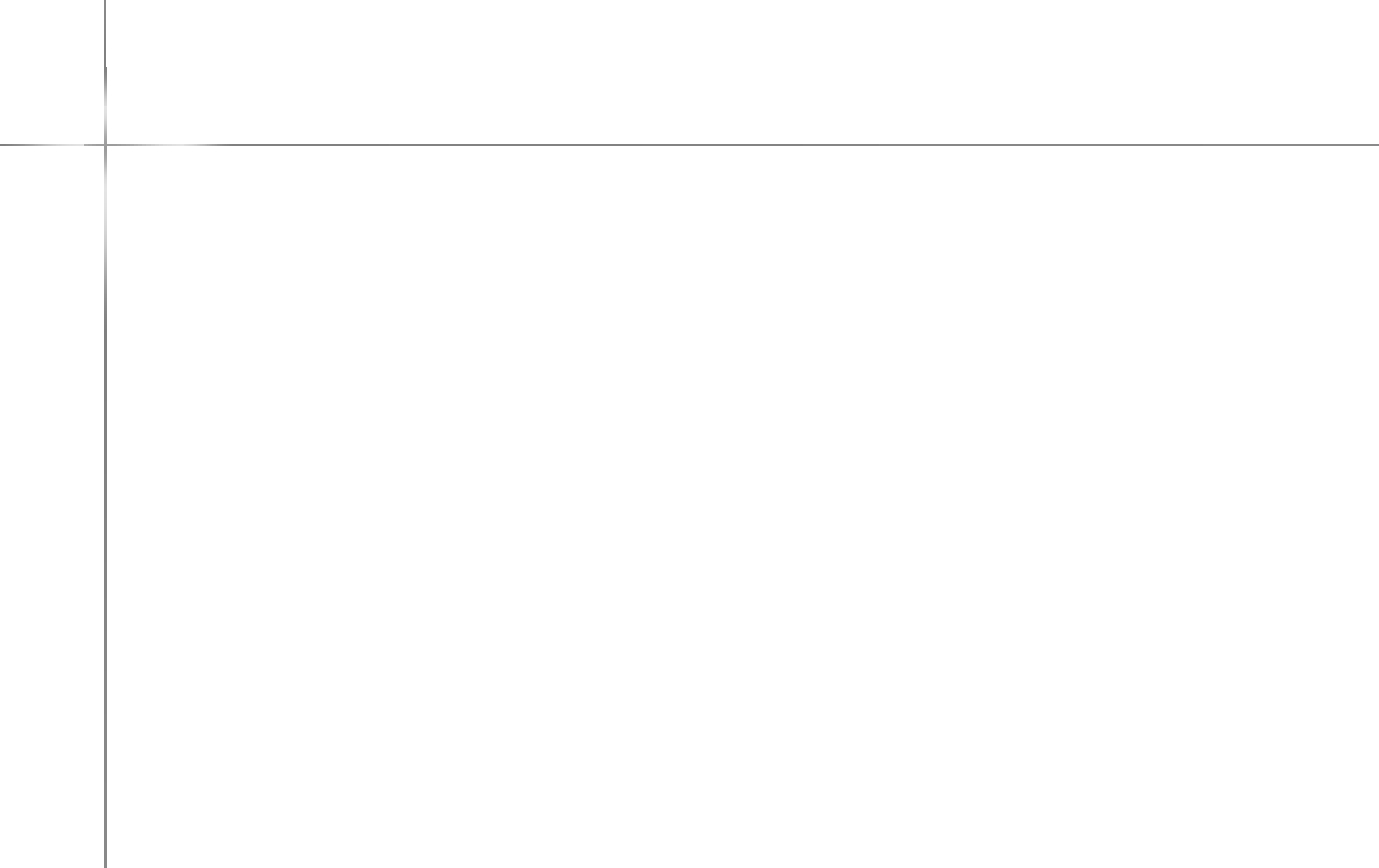
# 5 6

Микроархитектурный уровень

Топологический части микроконтроллера

***6***

Модель транзистора



***7***

|  |  |
| --- | --- |
|  |  |
| Алгоритмы. Программы. Команды. !**!**  «Алгоритм – конечный набор предписаний, определяющий решение задачи посредством конечного количества операций» (ISO 2382/1-84)  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   ***Слайд*** 8 |

# 7 8

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019



|  |  |
| --- | --- |
|  |  |
| Алгоритмы. Программы. Команды. !**!**  Свойства алгоритмов позволяют осуществлять их выполнение на вычислительной машине (ВМ). Процесс, порождаемый алгоритмом – | |
|  | *«вычислительныйпроцесс»*.  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)  ***Слайд*** 9 |

|  |  |
| --- | --- |
|  |  |
| 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.»  ***Слайд*** 10 |

# 9 10

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019



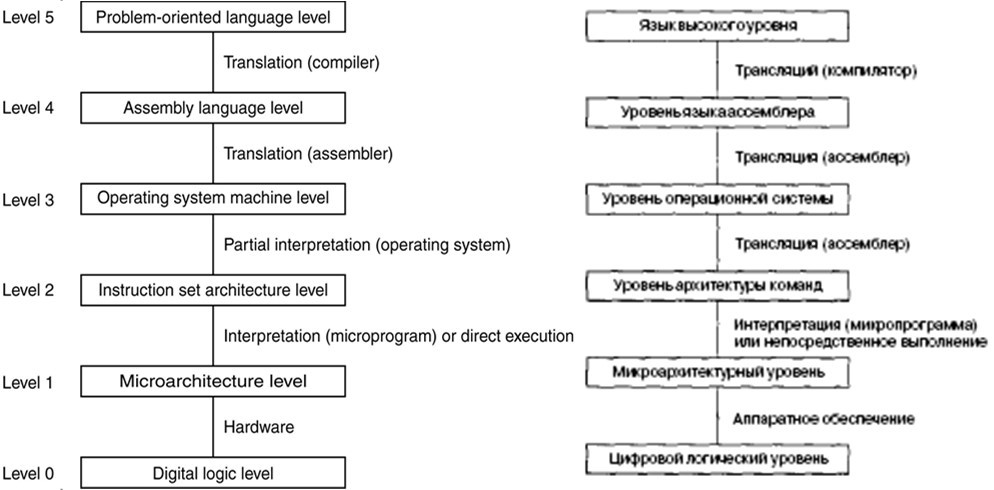
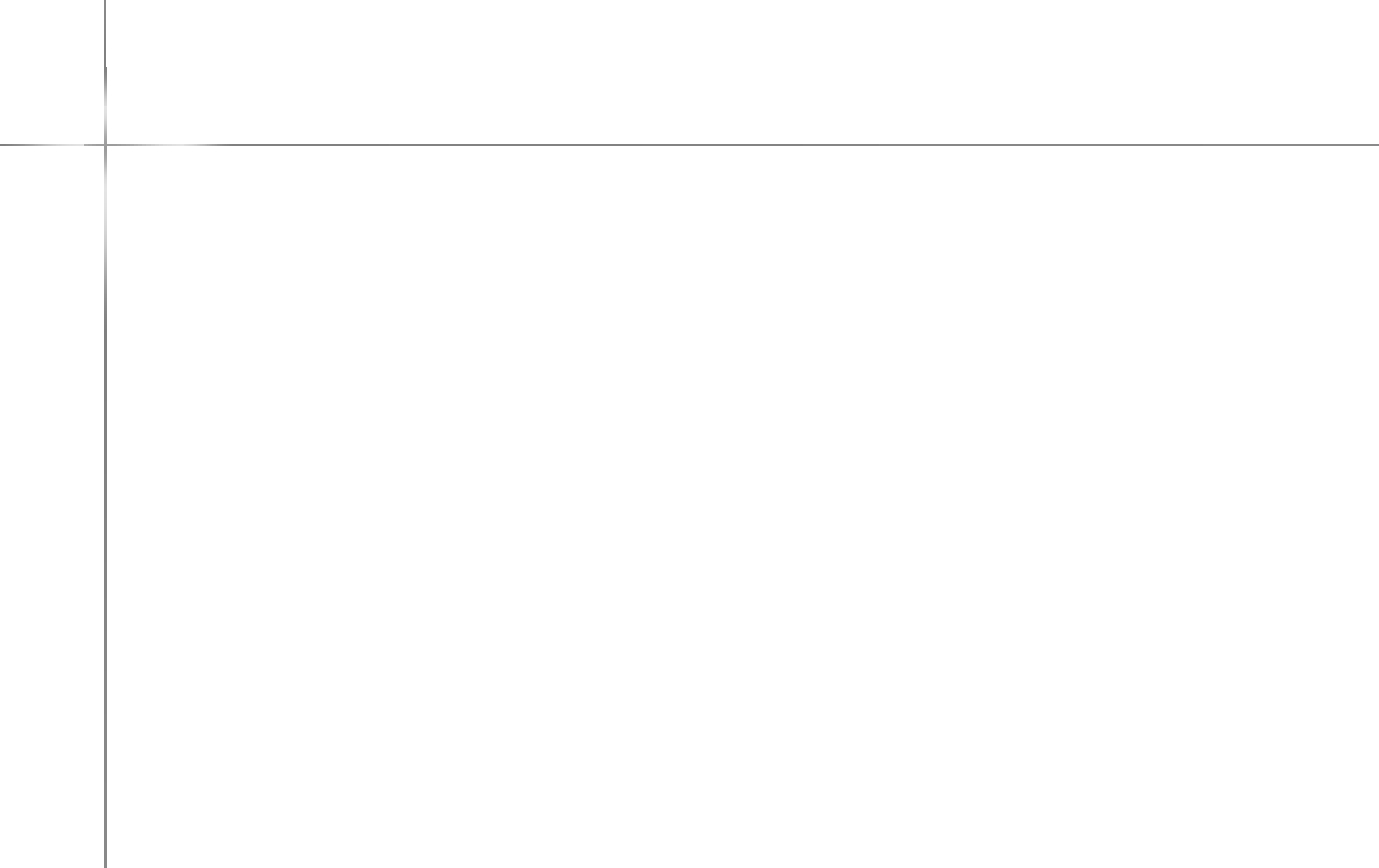
|  |  |
| --- | --- |
|  |  |
| Дилемма **!**!  Вычислительная машина намного проще человека  (на данный момент).  Алгоритм решения задачи для человека  алгоритму | |
|  | решения задачи для ВМ (по уровню детализации).  Электронные схемы могут распознавать и выполнять ограниченный набор простых команд. 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   ***Слайд*** 11 |

|  |  |
| --- | --- |
|  |  |
| Дилемма  Набор примитивных команд в совокупности составляют  *машинныйязык*.  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. | |
|  | ***Слайд*** 12 |

# 11 12

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019



Современные многоуровневые машины Contemporary Multilevel Machines

!**!**

A six-level computer.

The support method for each level is indicated below it . ***Слайд*** 13

|  |  |
| --- | --- |
|  |  |
| 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.   ***Слайд*** 14 |

# 13 14

Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем, Питер, 2007 г.



|  |  |
| --- | --- |
|  |  |
| 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.   ***Слайд*** 15 |

|  |  |
| --- | --- |
|  |  |
| 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. | |
|  | ***Слайд*** 16 |

# 15 16

Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем, Питер, 2007 г.

Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем, Питер, 2007 г.



|  |  |
| --- | --- |
|  |  |
| 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. ***Слайд*** 17 |

|  |  |
| --- | --- |
|  |  |
| 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.   ***Слайд*** 18 |

# 17 18

Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем, Питер, 2007 г.



|  |  |
| --- | --- |
| **High Level Language Program** | |
|  | **Compiler** |
| **Assembly Language Program** | |

|  |  |
| --- | --- |
|  |  |
| How to Speak Computer  **High Level Language** temp = v[k];  **Program** v[k] = v[k+1];  v[k+1] = temp; | |
|  | ***Слайд*** 19 |

|  |  |
| --- | --- |
|  |  |
| 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)**  ***Слайд*** 20 |

# 19 20



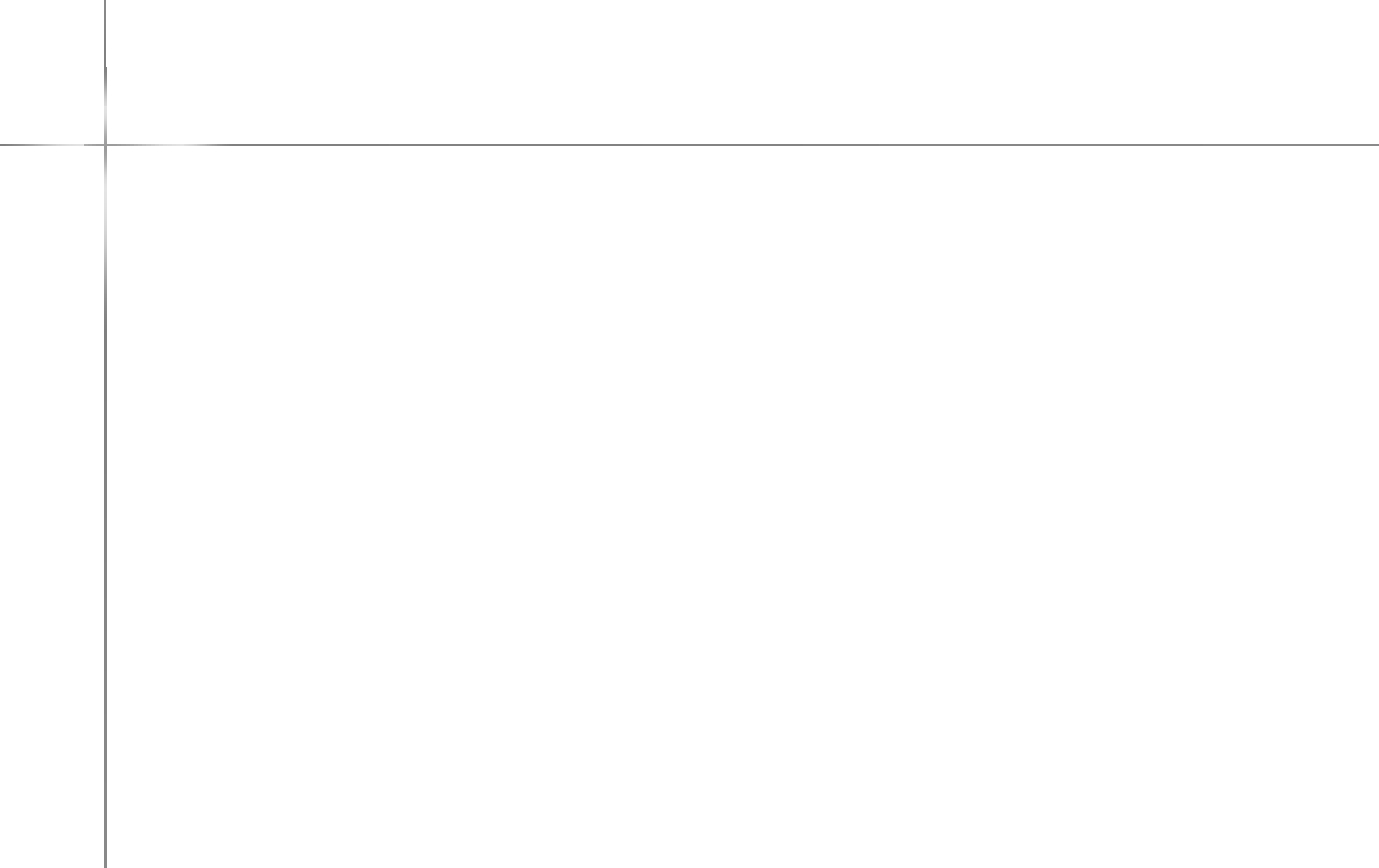
|  |  |
| --- | --- |
|  |  |
| How to Speak Computer  temp = v[k]; v[k] = v[k+1]; v[k+1] = temp;  **lw $15, 0($2)** | |
|  | **lw $16, 4($2)**  **sw $16, 0($2) sw $15, 4($2)**  1000110001100010000000000000000  1000110011110010000000000000100  1010110011110010000000000000000  1010110001100010000000000000100  ***Слайд*** 21 |

|  |  |
| --- | --- |
|  |  |
| 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)**  1000110001100010000000000000000  1000110011110010000000000000100  1010110011110010000000000000000  1010110001100010000000000000100  **Machine Interpretation**  Control Signal Spec ALUOP[0:3] <= InstReg[9:11] & MASK  ***Слайд*** 22 |

# 21 22

|  |  |  |
| --- | --- | --- |
| **High Level Language Program** | |  |
|  | **Compiler** | |
| **Assembly Language Program** | |  |
|  | **Assembler** | |
| **Machine Language Program** | |  |
|  | |  |

|  |  |  |
| --- | --- | --- |
| **High Level Language Program** | |  |
|  | **Compiler** | |
| **Assembly Language Program** | |  |
|  | **Assembler** | |
| **Machine Language Program** | |  |
|  | |  |
|  |  | |



The Instruction Set Architecture

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

**Application**

**Compiler**

**Operating System**

**ЦП Память**

**I/O system**

**Архитектура системы команд (ISA)**

**Цифровые лог. схемы Проектирование схем**

**Транзисторы**

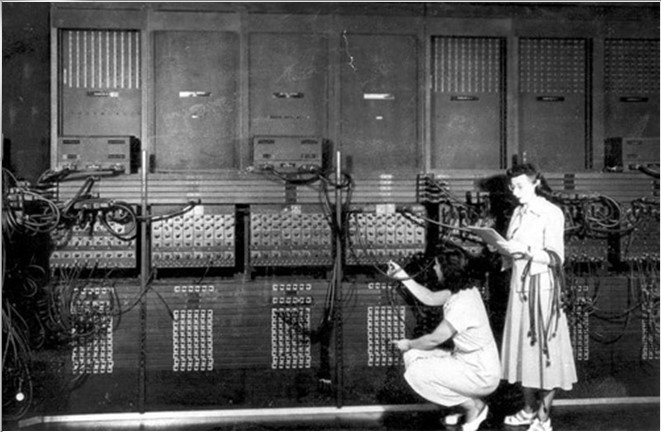
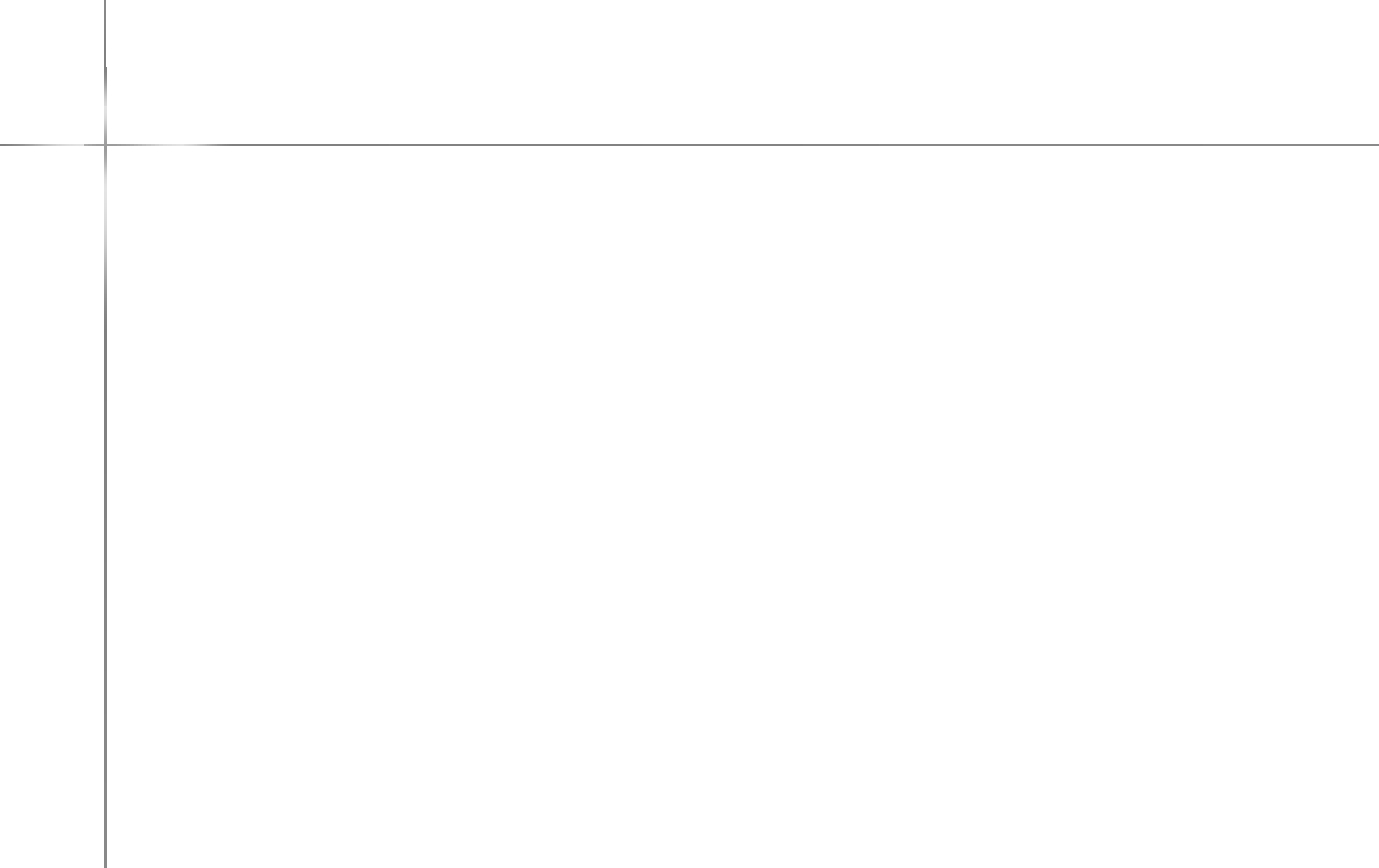
**Объект интереса дисциплины АЛОЦВМ**

***Слайд*** 23

|  |  |
| --- | --- |
|  |  |
| Развитие многоуровневых машин  Evolution of Multilevel Machines   * Изобретение микропрограммирования | |
|  | * Изобретение ОС * Перемещение функциональности системы на уровень микрокода * Устранение микропрограммирования   ***Слайд*** 24 |

# 23 24

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019



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. ***Слайд*** 25

|  |  |
| --- | --- |
|  |  |
| 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.  ***Слайд*** 26 |

# 25 26



|  |  |
| --- | --- |
|  |  |
| 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.   ***Слайд*** 27 |

|  |  |
| --- | --- |
|  |  |
| 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 ». *Слайд*** 28 |

# 27 28



|  |  |
| --- | --- |
|  |  |
| 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.  ***Слайд*** 29 |

|  |  |
| --- | --- |
|  |  |
| 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.   1. Команды для вызова и прекращения действия процедур. 2. Команды для ускорения циклов. 5.Команды для работы со строкой символов.   ***Слайд*** 30 |

# 29 30



|  |  |
| --- | --- |
|  |  |
| 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.   ***Слайд*** 31 |

|  |  |
| --- | --- |
|  |  |
| 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. ***Слайд*** 32 |

# 31 32



|  |  |
| --- | --- |
|  |  |
| Классификация архитектур системы команд **!**! (по составу и сложности команд)  Сложные операторы ЯВУ сущ. отличаются от простых машинных операций – семантический разрыв.  Архитектура c полным набором команд – CISC (Complex Instruction Set | |
|  | Computer)  (IBM 360, VAX, Intel 432, 1977 - 80)  Архитектура c сокр. набором команд – RISC (Reduced Instruction Set Computer)  (MIPS, SPARC, IBM RS600, 1987)  Архитектура c ком. словом сверхбольшой длины – VLIW (Very Long Instruction Word)  (Itanium, конец 1990-х)  ***Слайд*** 33 |

|  |  |
| --- | --- |
|  |  |
| Классификация архитектур системы команд **!**! (по составу и сложности команд)  Для CISC архитектуры характерно:   * небольшое число РОН * большое кол-во маш. команд, некоторые из них | |
|  | аппаратно реализуют операторы ЯВУ   * разнообразие способов адресации операндов * множество форматов команд различной разрядности * наличие команд, где обработка совмещается с обращением к памяти.   Недостатки CISC архитектуры:   * *высокаястоимостьаппарат.части(особенноУУ)* * *сложностисраспараллеливаниемвычислений*   ***Слайд*** 34 |

# 33 34



|  |  |
| --- | --- |
|  |  |
| Классификация архитектур системы команд !**!** (по составу и сложности команд)  Принципы RISC архитектуры (1980, Patterson & Ditzel): 1. Все команды непосредственно выполняются  аппаратным обеспечением. | |
|  | *Онинеинтерпретируютсямикрокомандами.*  В компьютерах типа CISC более сложные команды могут разбиваться на несколько частей, которые затем выполняются как последовательность микрокоманд. Эта дополнительная операция снижает скорость работы машины, но она полезна для редко встречающихся команд.  ***Слайд*** 35 |

|  |  |
| --- | --- |
|  |  |
| Принципы RISC архитектуры !**!** 2. Компьютер должен начинать выполнение большого  числа команд.  Один из главных способов увеличения производительности возможность приступать к как | |
|  | можно большему количеству команд в секунду.  Процессор 500-MIPS способен приступать к выполнению 500 млн. команд в секунду, и при этом не имеет значения, сколько времени занимает само выполнение этих команд  Этот принцип предполагает, *чтопараллелизмможет игратьглавнуюрольвулучшении производительности,посколькуприступатьк большомуколичествукомандзакороткий промежутоквремениможнотольковтомслучае, еслиодновременноможетвыполнятьсянесколько команд.*  ***Слайд*** 36 |

# 35 36



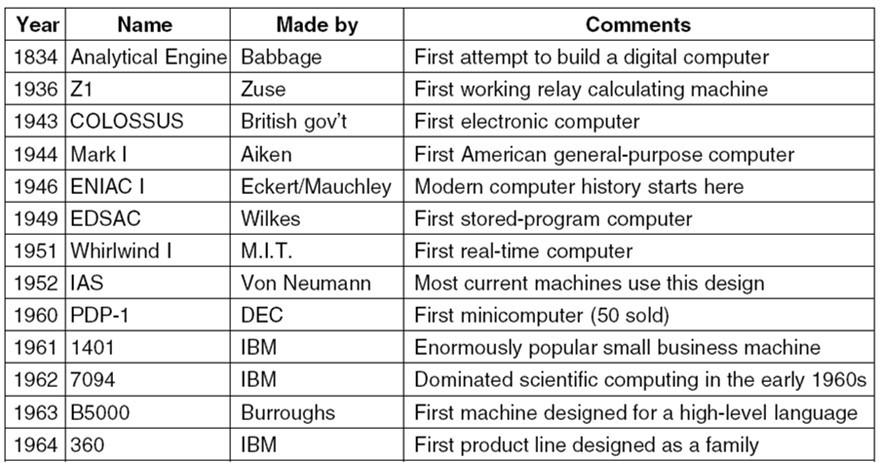
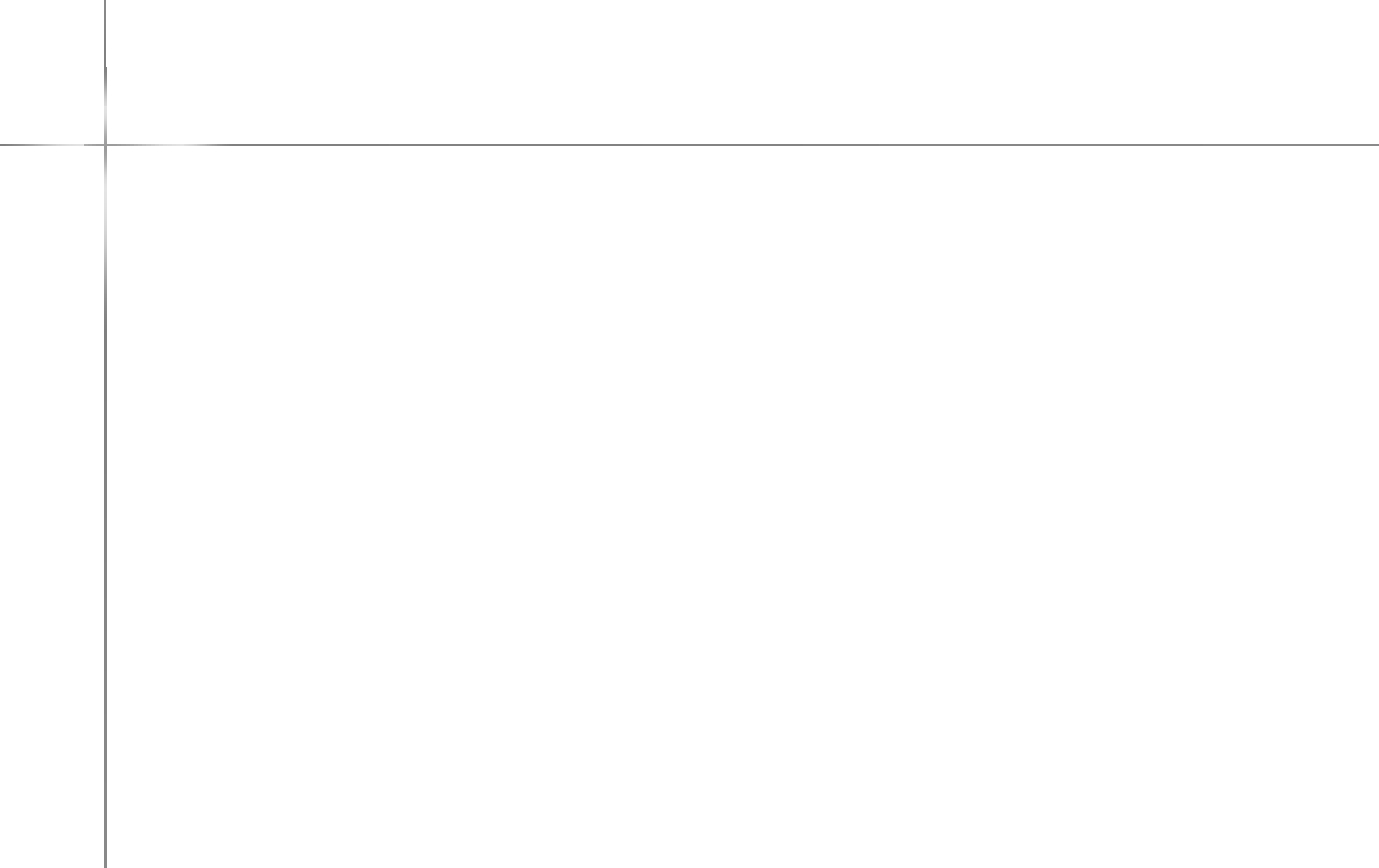
|  |  |
| --- | --- |
|  |  |
| Принципы RISC архитектуры !**!**  3. Команды должны легко декодироваться.  Предел количества вызываемых команд в секунду зависит от процесса декодирования отдельных | |
|  | команд.  Декодирование команд осуществляется для того, чтобы определить, какие ресурсы им необходимы и какие действия нужно выполнить.  Полезны любые средства, которые способствуют упрощению этого процесса.  *Чемменьшеразныхформатовкоманд,темлучше.*  ***Слайд*** 37 |

|  |  |
| --- | --- |
|  |  |
| Принципы RISC архитектуры **!**!  4. К памяти должны обращаться только команды загрузки и сохранения.  Поскольку доступ к памяти занимает много времени, а | |
|  | подобная задержка нежелательна, то эту могут выполнять другие команды, причём параллельно и заблаговременно.  Соответственно они не должны больше делать ничего, кроме передвижения операндов между регистрами и памятью. Из этого наблюдения следует,  *чтокпамятидолжныобращатьсятолькокоманды загрузкиисохранения(LOADиSTORE).*  ***Слайд*** 38 |

# 37 38



|  |  |
| --- | --- |
|  |  |
| Принципы RISC архитектуры **!**!  5. Должно быть большое количество регистров  Поскольку доступ к памяти происходит довольно медленно, в компьютере должно быть много | |
|  | регистров (по крайней мере 32). Если слово однажды вызвано из памяти, то при наличии большого числа регистров оно может содержаться в регистре до тех пор, пока будет не нужно.  Возвращение слова из регистра в память и новая загрузка этого же слова в регистр крайне нежелательны. Лучший способ избежать излишних перемещений —  *наличиедостаточногоколичестварегистров.*  ***Слайд*** 39 |

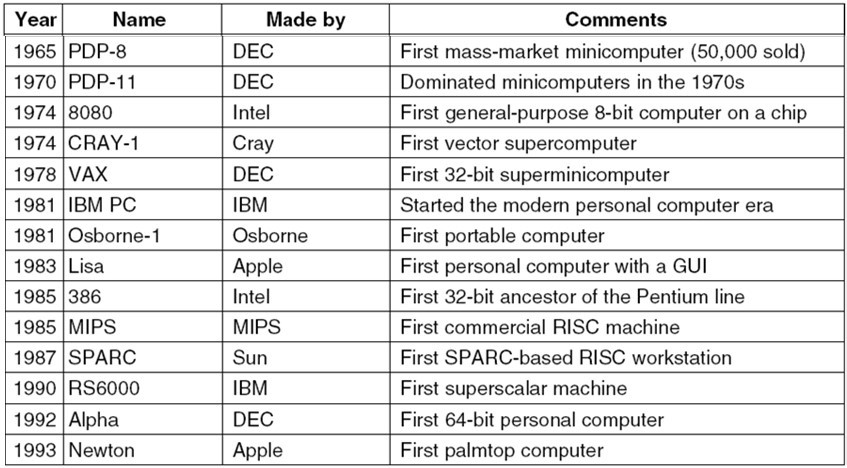
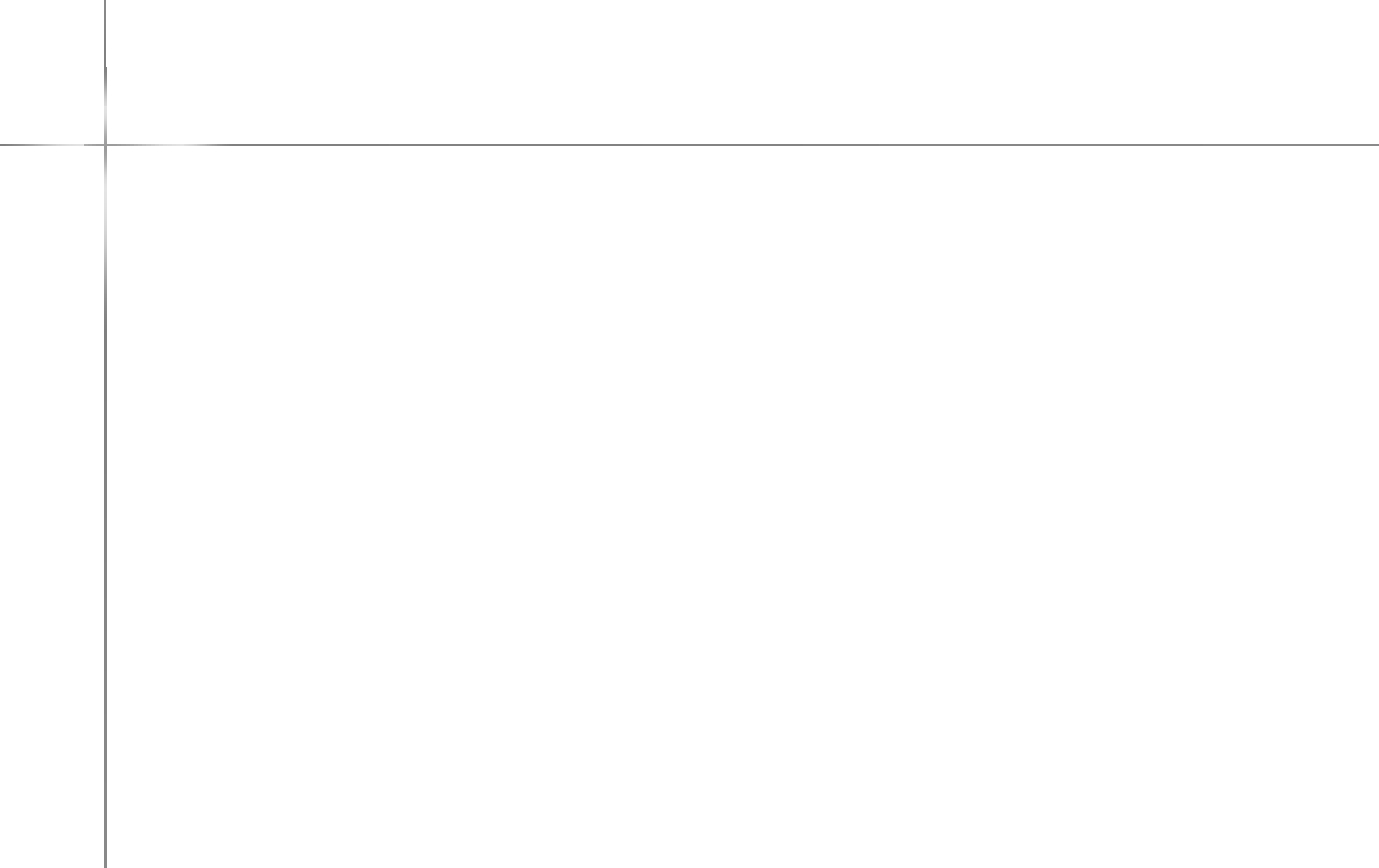


Вехи в развитии ЭВМ Milestones in Computer Architecture

***Слайд*** 40

# 39 40

Some milestones in the development of the modern digital computer.



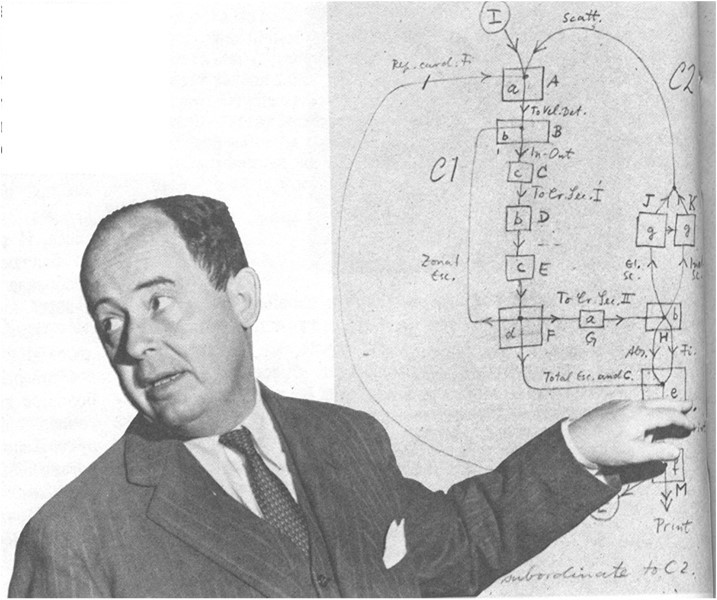
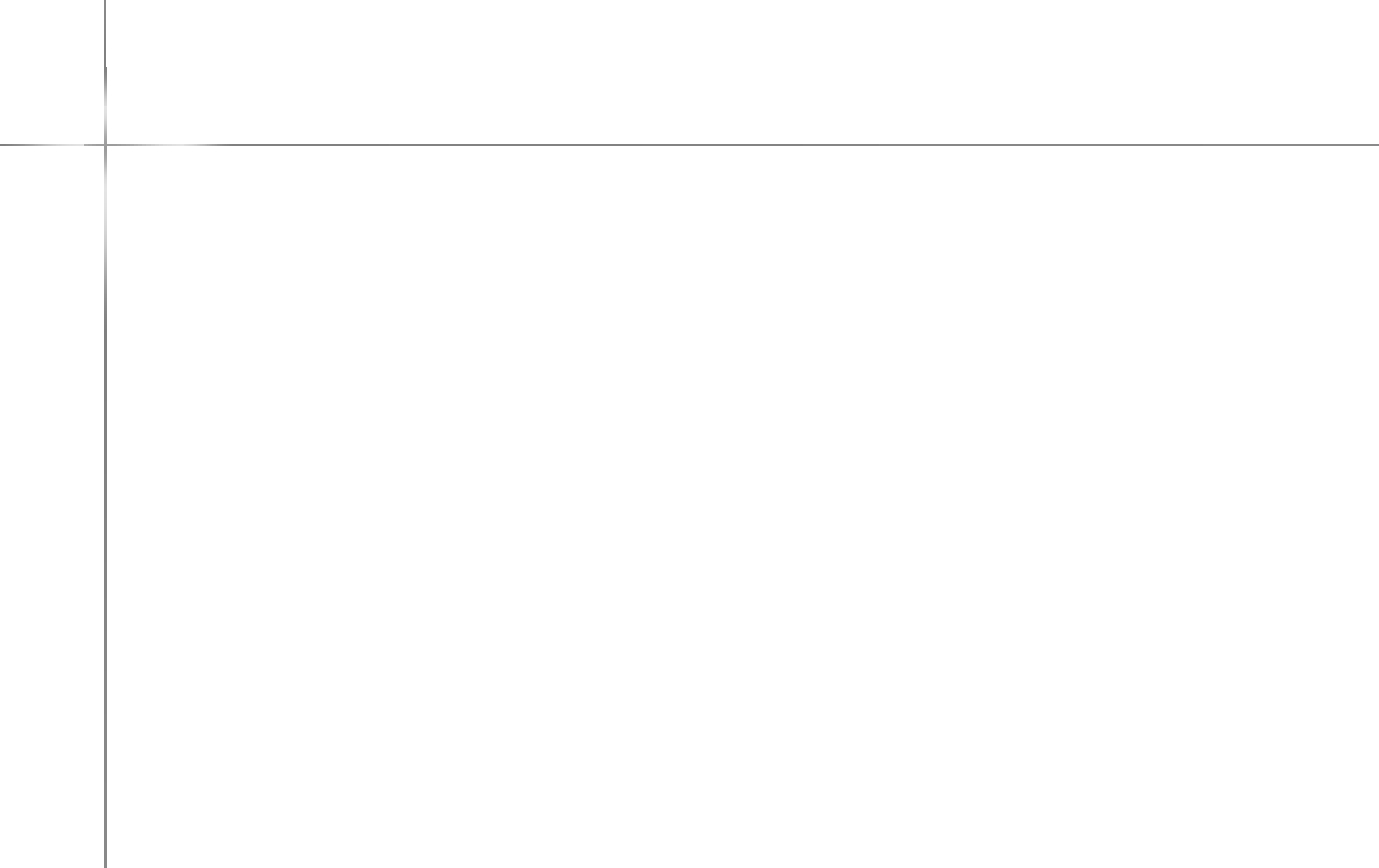
Вехи в развитии ЭВМ Milestones in Computer Architecture

***Слайд*** 41

|  |  |
| --- | --- |
|  |  |
| 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 – ?)  ***Слайд*** 42 |

# 41 42

|  |  |
| --- | --- |
|  |  |
| Факторы, влияющие на СиФО ЭВМ  Forces on Computer Architecture  **Technology Programming Languages** | |
|  | ***Applications***  **Computer Architecture**  **Operating Systems**  ***History***  ***Слайд*** 43 |



John von Neumann

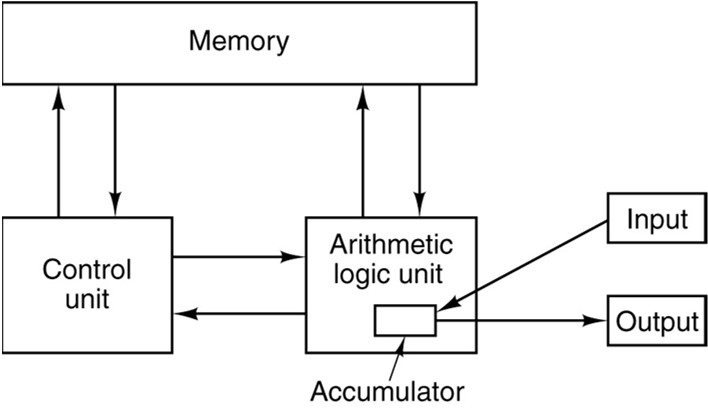
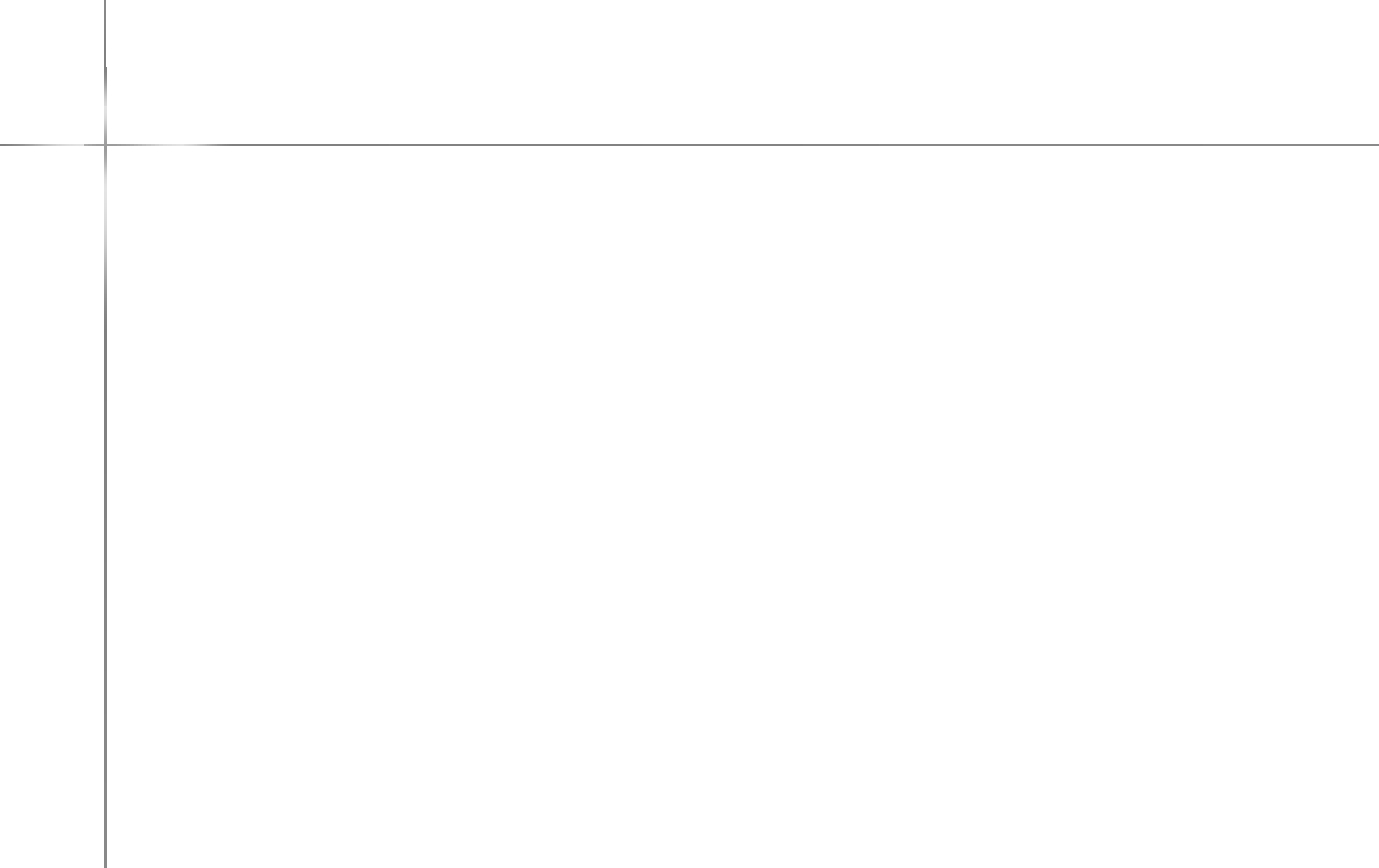
**!**!

***Слайд*** 44

# 43 44



|  |
| --- |
| 1001010010110000 |
| 0010100101010001 |
| 1111011101100110 |
| 1001010010110000 |
| 1001010010110000 |
| 1001010010110000 |



Von Neumann Machine

The original Von Neumann machine.

***Слайд*** 46

# 45 46



Пять классических компонентов любой ЭВМ

Input (mouse, keyboard, …)

* Output (display, printer, …)
* Memory
  + main (DRAM), cache (SRAM)
  + secondary (disk, CD, DVD, …)
* Datapath Processor

!**!**

**Input**

**Output**

* Control

(CPU)

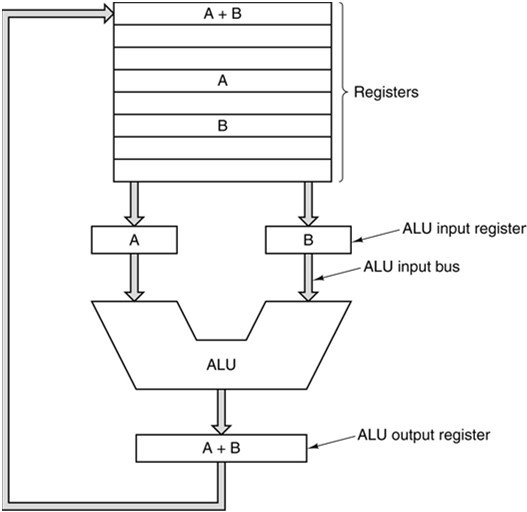
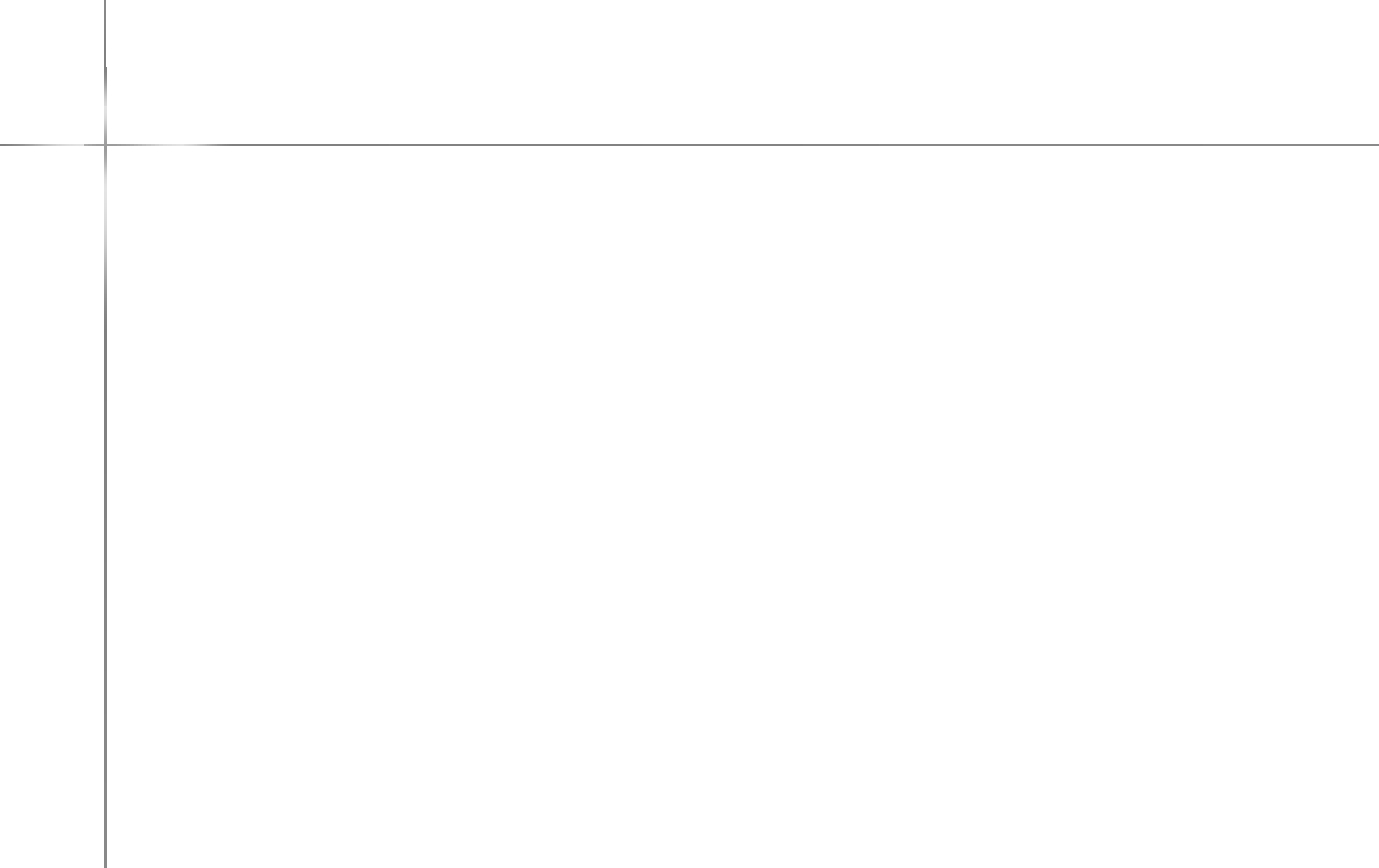
**Memory**

***Слайд*** 45

**Processor**

**Datapath**

**Control**



CPU Organization

!**!**

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

***Слайд*** 47



Цикл выполнения команды

!**!**

**Получение команды из «хранилища» программ**

**Определение задаваемого действия и фактического размера команды**

**Вычисление операндов**

**адресов**

**и получение**

**Вычисление результирующего значения либо состояния**

**Сохранение результатов операции в «хранилище» для последующего использования**

**Вычисление адреса следующей команды**

***Слайд*** 48

***Переход***

***к след. к-де***

*Сохранение*

*результата*

*Выполнение*

***Выборка операндов***

***Декод-ние команды***

***Выборка команды***

# 47 48



|  |  |
| --- | --- |
|  |  |
| 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)   ***Слайд*** 49 |

|  |  |
| --- | --- |
|  |  |
| 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   ***Слайд*** 50 |

# 49 50

Основы компьютерной техники, Д.И. Самаль, БГУИР, 2019



|  |  |
| --- | --- |
|  |  |
| 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   ***Слайд*** 51 |

|  |  |
| --- | --- |
|  |  |
| 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   ***Слайд*** 52 |

# 51 52



|  |  |
| --- | --- |
|  |  |
| 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.  ***Слайд*** 53 |

|  |  |
| --- | --- |
|  |  |
| 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.   ***Слайд*** 54 |

# 53 54



|  |  |
| --- | --- |
|  |  |
| 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   ***Слайд*** 55 |



Классификация архитектур системы команд

Архитектура c выделенным доступом к памяти Load/Store (CDC 6600, Cray I, 1963-76)

Архитектура c сокр. набором команд - RISC

(MIPS, SPARC, IBM RS600, 1987)

Архитектура c ком. словом сверхбольшой длины - VLIW (Itanium, конец 1990-х)

***Слайд*** 56

Архитектура c без- операндным набором команд

– ROSC (IGNITE, ИТФ

«Технофорт», 2001)

тектура c полным набором нд - CISC

, Intel 432, 1977 - 80)

Архи кома (VAX

Регистровая Архитектура (IBM 360, 1964)

Стековая Архитектура (B5500, B6500, 1963-66)

Аккумуляторная Архитектура (EDSAC, 1950)

Архитектура системы команд

# 55 56



|  |  |
| --- | --- |
|  |  |
| ИТОГО: !**!**  Stack machine:  “Push” загружает содержимое памяти в 1ый регистр (“вершину стека”), передвигая все остальные регистры вниз  “Pop” действует в обратном порядке | |
|  | “Add” комбинирует содержимое первых двух регистров, передвигая остальные вверх.  Accumulator machine:  Только один регистр (называемый “аккумулятором”)  Команды включают в себя “store” и саму операцию “acc  acc + mem”  Register-Memory machine :  Арифметические команды могут использовать данные как из регистров так и/или из памяти  Load-Store Machine (aka Register-Register Machine):  Арифметические команды могут оперировать только с ***Слайд*** 57  содержимым регистров. |

|  |  |
| --- | --- |
|  |  |
| Сравнение классов АСК  Последовательности кода для  **C = A + B**  Stack Accumulator Register-Memory Load-Store | |
| **P P A P** | **ush A Load A Add C, A, B Load R1,A ush B Add B Load R2,B**  **dd Store C Add R3,R1,R2**  **op C Store C,R3**  ***Слайд*** 58 |

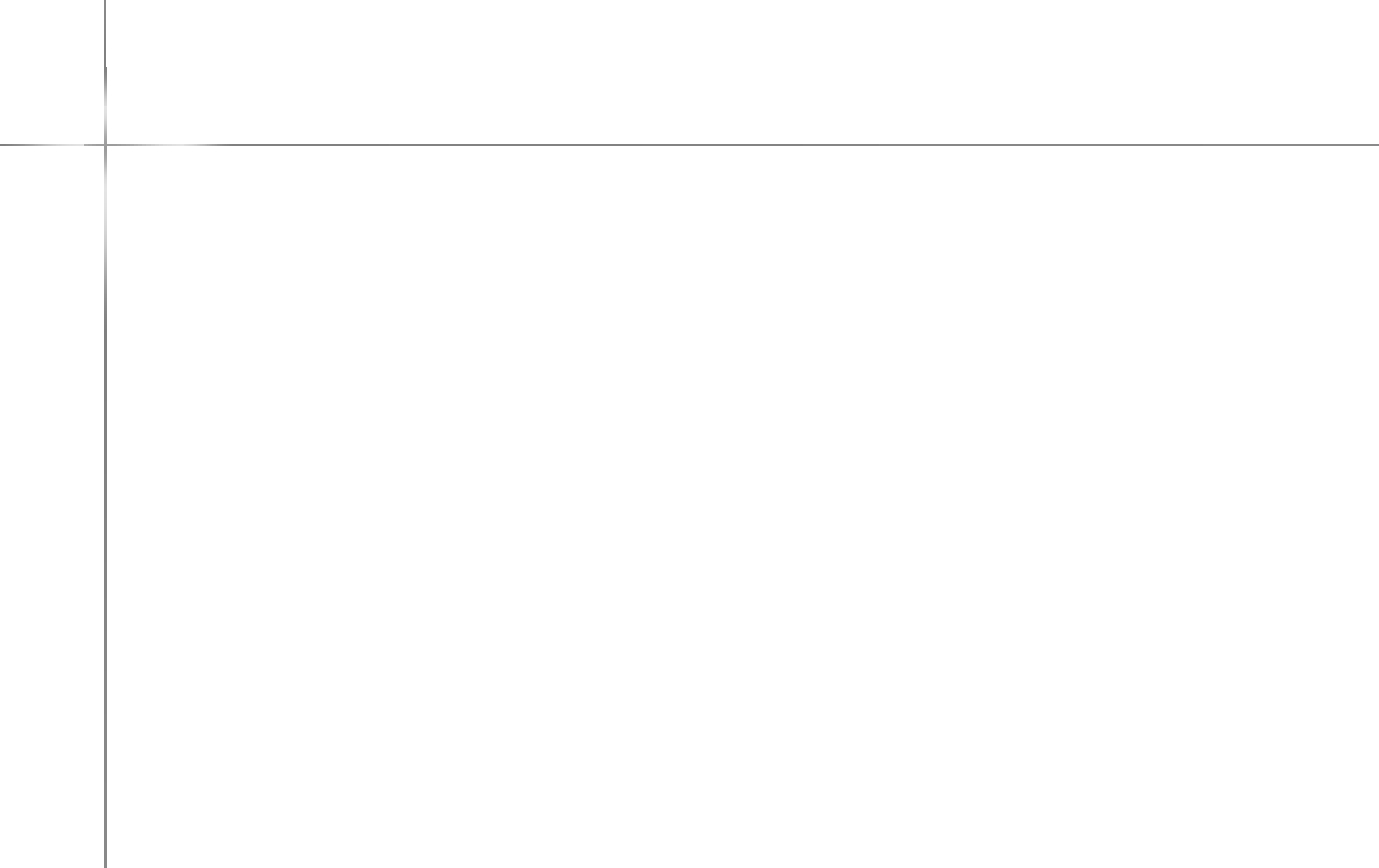
# 57 58



|  |  |
| --- | --- |
| Computer of the day | Историческая ретроспектива: последние минуты лекции |
| Computers: **Ввод: уши, глаза Вывод: рот, карандаш**  4000 BC **Регистры, кэш,**  to 1940’s **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.  ***Слайд*** 59 | |

|  |  |
| --- | --- |
|  |  |
| Вопросы к лекции   1. Что такое «алгоритм»? 2. В чём состоит основная дилемма вычислительной техники? | |
|  | 1. В чём заключается смысл построения   многоуровневых вычислительных машин?   1. Перечислите уровни абстракции современных ЭВМ? 2. В чём смысл микропрограммирования? 3. Какие факторы определяют развитие ЭВМ? 4. Пять основных блоков классической ЭВМ по фон Нейману. 5. Принципы организации вычислений по фон Нейману.   ***Слайд*** 60 |

# 59 60



|  |  |
| --- | --- |
|  |  |
| 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.  ***Слайд*** 61 |

61