Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра ВС

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ

по дисциплине «Архитектура Вычислительных Систем»

Вариант 18

Выполнил: студент гр. ИП-712

Алексеев С.В.

Проверил: Ефимов А.В.

Новосибирск 2019 г.

Оглавление

[1. ЗАДАНИЕ 2](#_Toc25439342)

[2. АНАЛИЗ ВОЗМОЖНОСТЕЙ ПРОЦЕССОРОВ С МИКРОАРХИТЕКТУРОЙ ЭЛЬБРУС 2](#_Toc25439343)

[3. ПРИМЕР ФУНКЦИОНАЛЬНОЙ СТРУКТУРЫ СОВРЕМЕННОГО ПРОЦЕССОРА 6](#_Toc25439344)

[4. РАСЧЕТ ФУНКЦИЙ НАДЕЖНОСТИ И ГОТОВНОСТИ 10](#_Toc25439345)

[5. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 16](#_Toc25439346)

# 1. ЗАДАНИЕ

1. Произвести анализ возможностей процессоров с микроархитектурой Эльбрус. Привести пример функциональной структуры современного процессора.

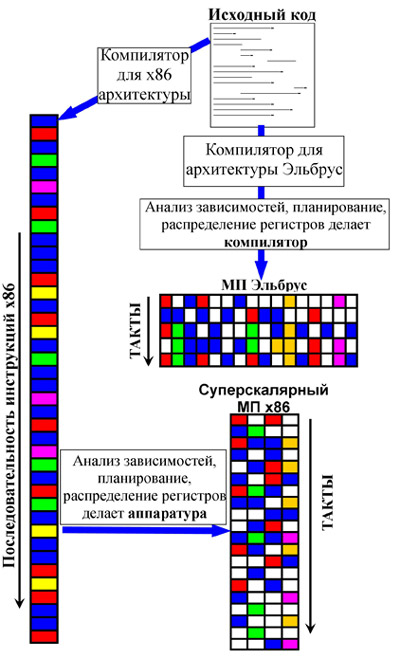
2. Выполнить численный расчет и построить графики для функции *r*(*t*) надежности и функции S(i, t) готовности ЭВМ для следующих количественных характеристик:

– интенсивности отказов 10-2 1/*ч* ,

– интенсивности восстановления 1 1/*ч* .

# 2. АНАЛИЗ ВОЗМОЖНОСТЕЙ ПРОЦЕССОРОВ С МИКРОАРХИТЕКТУРОЙ ЭЛЬБРУС

Процессоры данной архитектуры сравниваются по производительности с современными процессорами фирмы Интел в подсчете ГигаФлопс, при том, что имеют более скромные характеристики по частоте. Например при тактовой частоте всего в 1,3 ГГц Эльбрус 8С имеет производительность 250 гигафлопс (то есть 250 млрд операций в секунду) на операциях с одинарной точностью, что сравнимо с последними поколениями современного семейства процессоров Intel с архитектурой X86-64: 6-ядерный Intel Core i7-4930K (поколение Ivy Bridge) при тактовой частоте 3,7-4,2 ГГц имеет теоретический пик производительности в районе 177 гигафлопс.. Это достигается благодаря более продуманному распределению инструкций для процессора Эльбрус.



В традиционных архитектурах типа RISC или CISC (х86, PowerPC, SPARC, MIPS, ARM), на вход процессора поступает поток инструкций, которые рассчитаны на последовательное исполнение. Процессор может детектировать независимые операции и запускать их параллельно (суперскалярность) и даже менять их порядок (внеочередное исполнение). Однако динамический анализ зависимостей и поддержка внеочередного исполнения имеет свои ограничения: лучшие современные процессоры способны анализировать и запускать до 4-х команд за такт. Кроме того, соответствующие блоки внутри процессора потребляют заметное количество энергии.

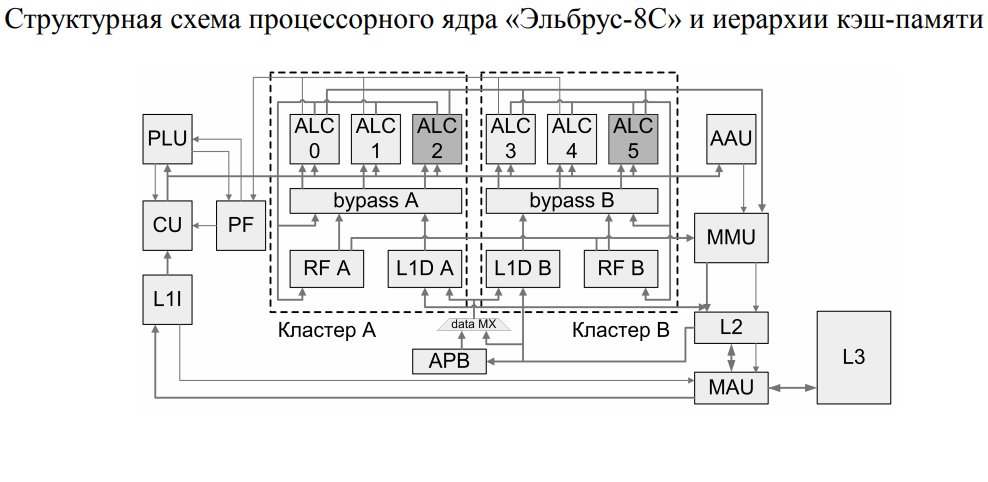
В архитектуре «Эльбрус» основную работу по анализу зависимостей и оптимизации порядка операций берет на себя компилятор. Процессору на вход поступают т.н. «широкие команды», в каждой из которых закодированы инструкции для всех исполнительных устройств процессора, которые должны быть запущены на данном такте. От процессора не требуется анализировать зависимости между операндами или переставлять операции между широкими командами: все это делает компилятор, исходя из анализа исходного кода и планирования ресурсов процессора. В результате аппаратура процессора может быть проще и экономичнее.

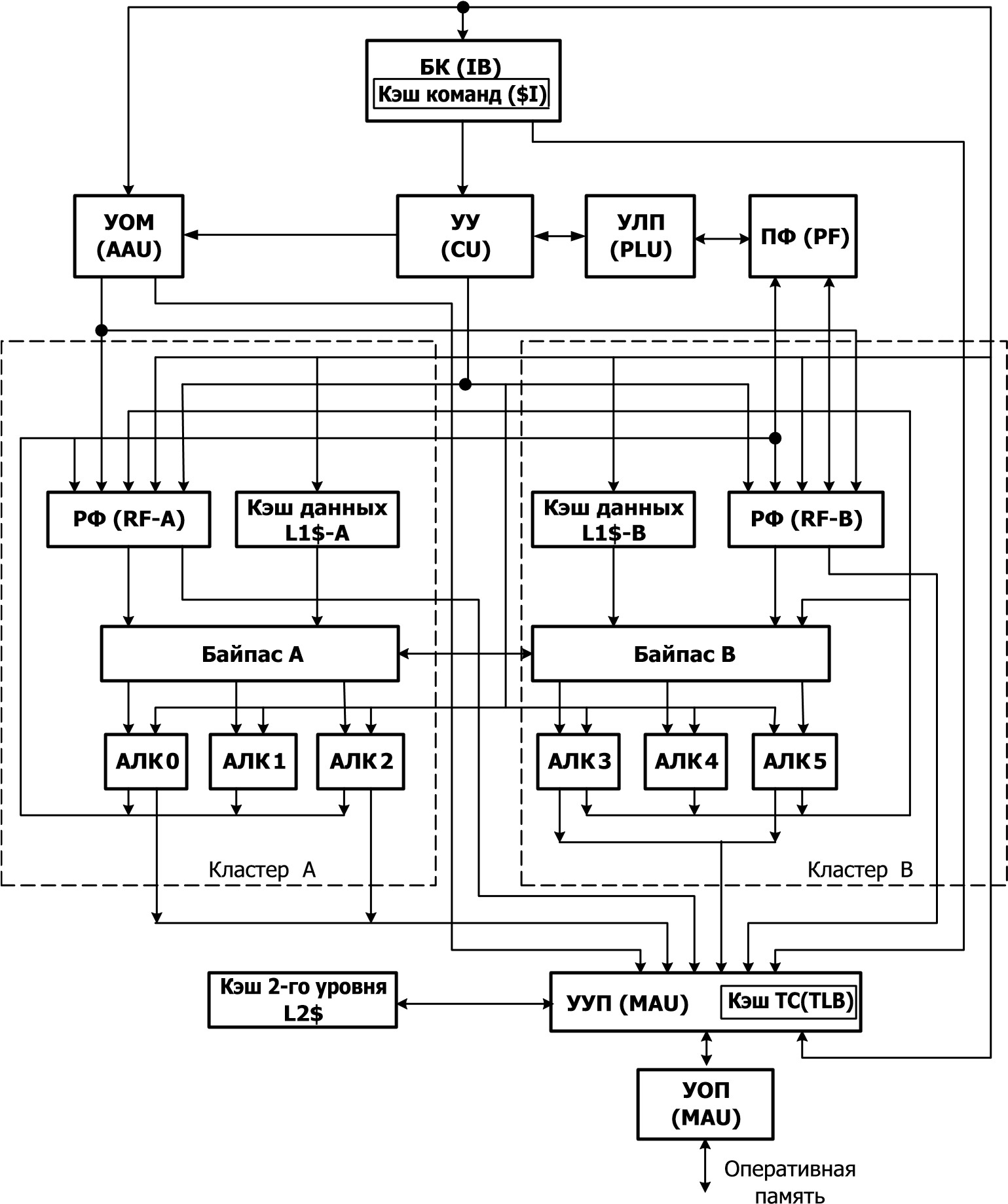
Компилятор способен анализировать исходный код гораздо тщательнее, чем аппаратура RISC/CISC процессора, и находить больше независимых операций. Поэтому в архитектуре Эльбрус больше параллельно работающих исполнительных устройств, чем в традиционных архитектурах, и на многих алгоритмах она демонстрирует непревзойденную архитектурную скорость.

**Возможности архитектуры Эльбрус:**

* 6 каналов арифметико-логических устройств (АЛУ), работающих параллельно.
* Регистровый файл из 256 84-разрядных регистров.
* Аппаратная поддержка циклов, в том числе с конвейеризацией. Повышает эффективность использования ресурсов процессора.
* Программируемое асинхронное устройство предварительной подкачки данных с отдельными каналами считывания. Позволяет скрыть задержки от доступа к памяти и полнее использовать АЛУ.
* Поддержка спекулятивных вычислений и однобитовых предикатов. Позволяет уменьшить число переходов и параллельно исполнять несколько ветвей программы.
* Широкая команда, способная при максимальном заполнении задать в одном такте до 23 операций (более 33 операций при упаковке операндов в векторные команды).

# 3. ПРИМЕР ФУНКЦИОНАЛЬНОЙ СТРУКТУРЫ СОВРЕМЕННОГО ПРОЦЕССОРА





**БК — буфер команд** (IB — Instruction Buffer) предназначен для вызова программного кода из памяти, буферизации на время, достаточное для обеспечения непрерывности дешифрации, и выдачи его в устройство управления для последующей обработки. Накопление программного (объектного) кода обеспечивает кэш команд I$ емкостью 64 Кбайт.

**УУ —** **устройство управления** (CU — Control Unit) выполняет считывание программного кода из буферной памяти БК, распаковку широких команд, их дешифрацию и переключение ветвей программы при выполнении команд переходов. С этой целью в структуре УУ предусмотрены обработка последовательности ШК основной ветви, предварительная подкачка ШК трех ветвей предполагаемого ветвления и распаковка первой из команд каждой ветви. Заметим, что подготовка перехода выполняется на фоне выполнения основной ветви и поэтому не приводит к замедлению вычислительного процесса. С выхода УУ в исполнительные устройства выдается распакованная ШК основного потока или одного из подготовленных потоков (при наличии в основном потоке операции передачи управления с реализовавшимся условием перехода).

**ПФ —** **предикатный файл** (PF — Predicate File) хранит первичные предикаты — битовые значения, выработанные операциями сравнения, и вторичные предикаты — результаты логических операций над первичными предикатами.

Предикатный файл — набор 32 двухразрядных регистров (по одному разряду для предиката и тега). С помощью предикатных значений может быть задан режим условного выполнения операции или условий для команд ветвлений.

**УЛП —** **устройство логических предикатов** (PLU — Predicate Logic Unit) предназначено для выполнения операций считывания предикатов из файла ПФ и логических операций формирования вторичных предикатов, которые также могут быть записаны в предикатный файл. Эти операции выполняются над малоформатными значениями, поэтому в целях разгрузки арифметико-логических каналов обработка предикатов возложена на УЛП. Ее результаты направляются в устройство управления для задания режима условного выполнения команд, отрабатываемых в арифметико-логических каналах, или выполнения команд передачи управления.

**Арифметико-логические каналы (АЛК)** предназначены для исполнения обычных арифметических и логических операций, операций обращения к памяти и обработки адресных данных (дескрипторов, указателей и др.). В состав микропроцессора входят шесть арифметико-логических каналов (АЛК0—АЛК5), разделенных на два кластера. Арифметико-логические каналы работают параллельно и исполняют в основном одинаковый набор операций. В качестве операндов служат данные из РгФ или результаты других исполнительных устройств. Не все каналы идентичны, поскольку не все операции выполняются одинаково часто. Например, довольно редкая операция деления реализована только в АЛК5. Арифметико-логические каналы имеют отдельные устройства для исполнения целочисленных и вещественных операций. Это не относится к целочисленным операциям умножения и деления — для них введены соответствующие блоки в вещественных исполнительных устройствах.

**Регистровый файл (РгФ)** предназначен для хранения локальных данных процедуры и результатов выполненных операций. Он представляет собой сверхоперативное запоминающее устройство с произвольным доступом, обращение которому осуществляется через порты. Многочисленность абонентов регистрового файла требует большого количества портов для обслуживания всех запросов одновременно. С целью сокращения их количества регистровый файл реализован в виде двух одинаковых блоков — РгФ-A и РгФ-B, по одному в каждом кластере. Причем, в отличие от используемых во многих микропроцессорах, блок является общим для целочисленных и вещественных устройств арифметики, что позволило повысить эффективность его использования. Блоки регистрового файла содержат одни и те же данные, поскольку результат любой операции записывается одновременно в оба блока. Этим обеспечивается когерентность данных — свойство, которое используется при работе с общими данными.

Блок РгФ содержит 256 регистров. При запуске процедуры ей выделяется определенный участок смежных регистров, называемый окном. Активная (выполняемая в данный момент) процедура может обращаться только к регистрам своего окна. Если все регистры окажутся занятыми, автоматически выполняется откачка. В результате откачки данные пересылаются в стек процедур, размещенный в оперативной памяти. Таким образом, регистровый файл служит в качестве аппаратной вершины стека процедуры. Все ранее работавшие, но не завершенные процедуры сохраняют свои окна в регистровом файле. Возврат процедур к своим окнам осуществляется по мере окончания работы запущенных ими процедур. Иными словами, реализуется дисциплина (LIFO) — активизируется та процедура, которая запустила завершившуюся в данный момент времени процедуру.

Чтобы результат, полученный некоторым исполнительным устройством, мог быть использован другим до записи в РгФ, исполнительные устройства связаны шинами байпаса. Благодаря этому в ряде случаев отпадает необходимость в считывании операндов из РгФ. Каждый кластер имеет отдельный блок байпаса, с помощью которого результаты передаются исполнительным устройствам не только своего, но и другого кластера. Кроме того, байпас позволяет использовать результаты считывания из кэш-памяти L1$ до их записи в регистровый файл.

**Кэш данных первого уровня L1$** выполнен в виде двух одинаковых блоков (L1$-A и L1$-B) емкостью 64 Кбайт, по одному в каждом кластере. Блоки L1$ хранят одинаковые данные, поскольку запись данных выполняется одновременно в оба блока. В блоке хранятся данные, которые используются в качестве операндов для исполнительных устройств АЛК. Но поскольку в общем случае операнды считываются из регистрового файла, они должны быть предварительно загружены в него из кэша L1$. Для этих целей предусмотрены операции загрузки, выполняемые исполнительными устройствами АЛК. В случае отсутствия требуемых данных в кэше L1$ операция загрузки продолжается поиском данных в кэш-памяти второго уровня L2$ с последующей записью их в оба блока L1$. Параллельная запись, так же как и в случае регистрового файла, вызвана необходимостью сохранения свойства когерентности данных обоих блоков L1$.

**Кэш второго уровня L2$** является общим для данных и программного кода, его объем составляет 256 Кбайт, степень ассоциативности — 4. Обращение к L2$ выполняется при отсутствии требуемых данных в L1$ или нужного программного кода в буферной памяти команд устройства БК. Если нужная информация отсутствует и в L2$, то формируется запрос к оперативной памяти. Считанная из ОП информация поступает потребителю и одновременно записывается в кэш-память вместо устаревших данных.

**УОМ — Устройство обращения к массивам** (AAU — Array Access Unit) предназначено для упреждающей подкачки элементов массива при выполнении векторных операций. Поскольку каждый элемент вектора обрабатывается одной и той же последовательностью операций, то обработка идет циклически. В микропроцессоре «Эльбрус» к началу очередного цикла нужный элемент вектора уже считан из памяти (кэша L2$ или ОП) и находится в буфере УОМ. Подкачка элементов массивов в буфер, использующий дисциплину очереди (FIFO), осуществляется на фоне выполнения основной (синхронной) программы параллельной ей (асинхронной) программой.

**УУП — устройство управления памятью** (MMU — Memory Management Unit) преобразует виртуальные адреса в физические. С целью ускорения этого процесса наиболее часто используемые строки таблицы страниц хранятся в кэше таблицы страниц (TLB) объемом 64 строки. Если нужная строка в нем отсутствует, выполняется аппаратный поиск в таблице страниц, хранящейся в памяти, и загрузка найденной строки в кэш вместо устаревшей. Кроме того, рассматриваемое устройство обеспечивает обращения в кэш L2$ и ОП.

**УОП — устройство обращения в память** (MAU — Memory Access Unit) предназначено для связи микропроцессора с ОП. Оно содержит буферы операций считывания и записи, позволяющих осуществить потоковое обслуживание заявок. Обмен с памятью осуществляется через 16-байтовый канал с раздельными шинами для передачи и приема данных. Обмен выполняется блоками по 32 или 64 байт.

# 4. РАСЧЕТ ФУНКЦИЙ НАДЕЖНОСТИ И ГОТОВНОСТИ

Функция (или вероятность безотказной работы) относится к основным показателям надежности ЭВМ. Характеризует производительность ЭВМ на промежутке времени, то есть эта функция обеспечивает потенциально возможную производительность. Функцией надежности ЭВМ называется



где запись  означает вероятность того, что для всякого , принадлежащего промежутку времени  производительность  ЭВМ равна единице, т.е. равна потенциально возможной.

Функция  обладает следующими свойствами:

1.  Т.е. машина в момент начала функционирования находится в работоспособном состоянии.
2.  Событие, заключающееся в том, что ЭВМ работоспособна на конечном промежутке времени, является достоверным.
3.  для ;

Функцией ненадежности (или вероятностью отказа) ЭВМ называется



Функция  позволяет определить среднее время безотказной работы (средняя наработка до отказа). По определению, среднее время  безотказной работы ЭВМ и оценка  соответственно равны:



где – время безотказной работы -й машины, 

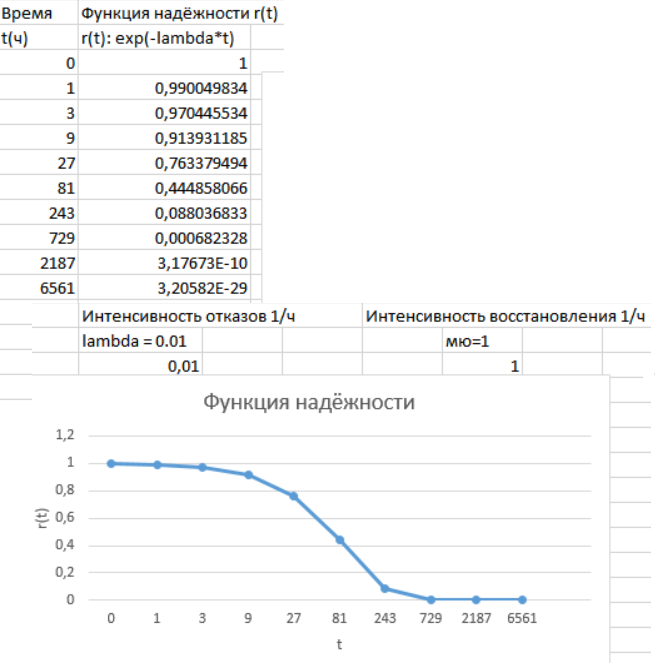
Интенсивностью отказов (лямбда-характеристикой) ЭВМ называется функция



Практически установлено, что зависимость интенсивности отказов от времени имеет место на периоде приработки ЭВМ. После приработки ЭВМ интенсивность отказов остается постоянной (до вхождения в предельное состояние или, по крайней мере, в течение промежутка времени, перекрывающего время морального старения). Следовательно, в нормальных условиях эксплуатации ЭВМ  а функция надежности и математическое ожидание времени *безотказной работы* соответственно равны:



рассчитаем значения функции и построим график:



Теперь рассчитаем значения функции готовности. Функция готовности ЭВМ

есть вероятность того, что (в условиях потока отказов и восстановлений) машина будет иметь в момент времени  производительность, равную единице, т.е. равную потенциально возможной.

Функция готовности ЭВМ обладает следующими свойствами:

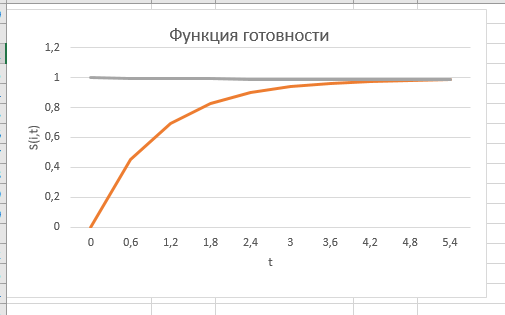
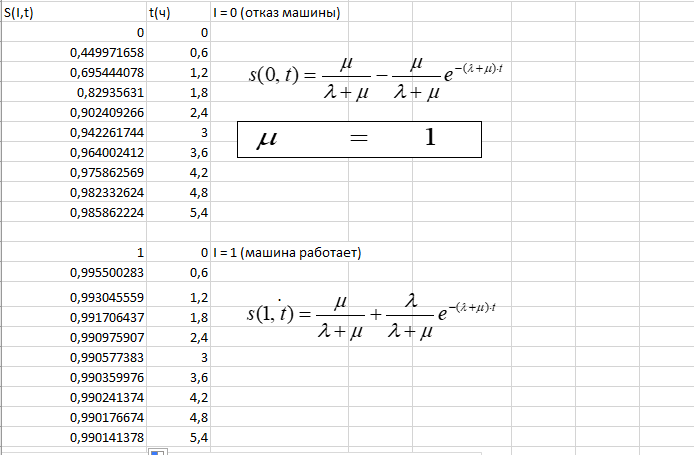
1. 
2. 
3.   для 

Расчет будем производить по следующим формулам:

;

.

для начальных состояний ЭВМ , причем  соответствует состоянию отказа, а  – работоспособному состоянию машины.

Рассчитаем значения функции и построим график: 

+

# 5. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. Конспект лекций по курсу “Архитектура вычислительных систем”
3. Ким А. К., Перекатов В. И., Ермаков С. Г. Микропроцессоры и вычислительные комплексы семейства «Эльбрус». — СПб.: Питер, 2013. — 272 с