Федеральное агентство по образованию
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Зав кафедрой АОИ, профессор
_____Ю.П. Ехлаков

“___”_____2010 г.

ПРАКТИКУМ

Методические указания по выполнению практических занятий по дисциплине: «Вычислительные системы и телекоммуникации» для студентов направления подготовки 080700 «Бизнес-информатика»

Проф. Н.В. Замятин

Томск 2010

Методические указания к практическим занятиям по дисциплине "Вычислительные системы и телекоммуникации" для студентов специальности 080700 “бизнес-информатика” дневной формы обучения.
Составил Замятин Н.В.-Томск, ТУСУР, 2010 г. – 51 с.

Кафедра автоматизации обработки информации

Содержание

1. Введение……..4
2. Операционный автомат…….....5
3. Управляющий автомат…….....12
4. Программирование ЭВМ в машинных кодах……..19
5.Исследование обстановки в сети. Функции ping и tracert ……..…..32
6. Исследование канала связи…….36
7. Изучение приложения FTP…….41
8. Список рекомендуемой литературы……..……..51

1. ВВЕДЕНИЕ

Методические указания к практическим занятиям по дисциплине " Вычислительные системы и телекоммуникации" содержат описания материала для выпонения практических работ.

Практические работы расположены последовательно, начиная синтеза аппаратных элементов ЭВМ, затем программное управление, далее сети ЭВМ и сетевые технологии.

Практические работы ориентированы на использованиеПЭВМ, совместимых с микропроцессорами семейства 8086.

Цель практимума

Целью практических занятий является закрепление практических навыков по синтезу операционных устройств, программированию ЭВМ, исследованию локальных сетей, знакомство с сетевыми технологиями.

Организация и проведение практических занятий

Каждый студент получает индивидуальное задание в соответствии с номером в журнале, выпоняет его и оформляет отчет по заданию на практическое занятие. Выполнение индивидуального задания предполагает предварительное изучение соответствующего раздела дисциплины и методических указаний к очередному занятию.

Для допуска к выполнению индивидуального задания студент должен ознакомиться с темами для проработки и предварительно написать текст программы, в соответствии со своим заданием.

Текст программы составляется на ассемблере по указанию преподавателя и с учетом глубины знаний студентами этого языка.

В течение выполнения индивидуального задания студент должен ответить на контрольные вопросы по предыдущему индивидуальному заданию. К практическим занятиям не допускаются студенты, не сдавшие более двух индивидуальных заданий.

Пропущенные индивидуальные задания выполняются в конце семестра.

2. Индивидуальное задание **№1. Синтез операционного автомата (4 часа)**

Понять, каким образом выполняются арифметическо - логические операции в микропроцессоре. Научиться синтезировать операционный автомат (ОА) для выполнения арифметических операций сложения и умножения.

Основные понятия

Любой универсальный преобразователь информации состоит из следующих операционных устройств (ОУ):

- арифметическо-логического устройства,
- памяти
- устройств ввода-вывода
- шин передачи сигналов.

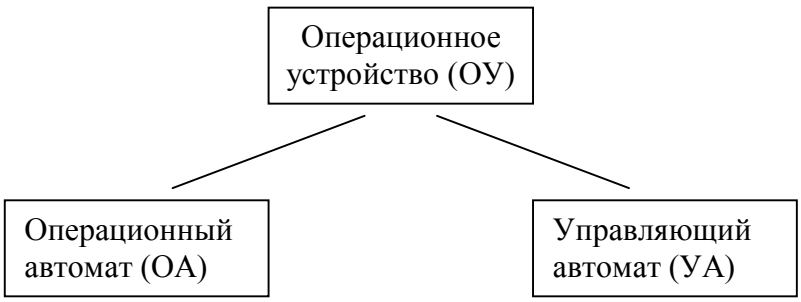
Согласно концепции академика Глушкова В.М. любое операционное устройство декомпозируется на две части - пассивную исполнительную (ОА) и активную управляющую (УА).

АЛУ часть узлов (сумматор и др. КС) являются исполнителями, а часть узлов (распределитель сигналов) являются управляющими. У этих узлов разные принципы построения, организации. Поскольку принципы организации разные, то и их проектирование тоже раздельное.

Операционное устройство представляет собой конечный автомат, состоящий из двух частей:

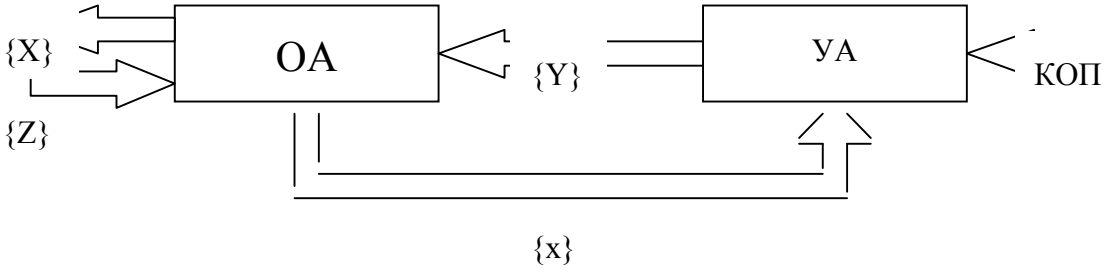
1.) операционный автомат;

2.) управляющий автомат;



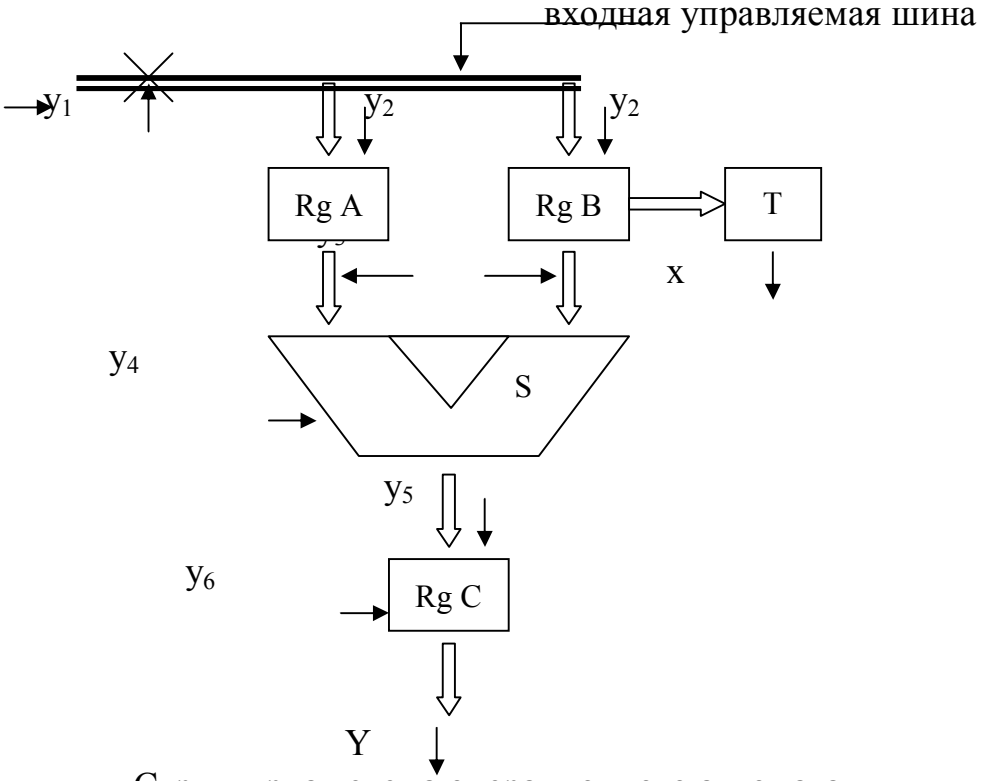
Операционный автомат (ОА) – предназначен для арифметических и логических операций под управлением управляющего автомата.

Управляющий автомат (УА) – предназначен для формирования управляющих воздействий.



КОП – код операции;
{X} – входные сигналы;
{Y} – управляющие команды;
{Z} – выходные сигналы;
{x} – множество осведомительных сигналов (количество, переполнение и т. п.);
Основа функционирования – принцип микропрограммного управления

1. Любая операция это сложное действие, состоящее из совокупности элементарных действий, называемых микрооперациями (МО). Выполнение каждой МО осуществляется специальной комбинационной схемой (КС) за один такт машинного времени.
2. Порядок выполнения МО определяется алгоритмом операции и зависит от значений логических условий x. ЛУ принимают значения истина или ложь в зависимости от значений операндов.
3. Алгоритм, представленный, записанный в терминах МО и ЛУ, называется микропрограммой (МП). МП задает порядок выполнения МО и проверки ЛУ во времени.
4. Совокупность микропрограмм $МП_1$, …, $МП_g$ задает функцию ОУ.



Структурная схема операционного автомата

Микрооперации y_1, y_2, y_3, y_4, y_5, y_6 представляющие одноразрядные сигналы, и поступающие на входы регистров и сумматоров, формирует УА
Обработка информации с помощью ОУ осуществляется путем выполнения операций из списка F в последовательности, которая задается алгоритмом (программой) решения задачи: ОА, выполняя программу, распределяет выполнение операций, предписанных командами программы, между различными ОУ - АЛУ, контроллерами ПУ.

Для синтеза ОА необходимо выполнить следующую последовательность действий:

1. Синтезировать RS –триггер,
2. Синтезировать параллельный регистр,
3. Синтезировать сумматор,
4. Собрать триггер, регистр и сумматор в среде Electronics Workbench, промоделировать их работу. Привести процедуры синтеза, схемы и эпюры в отчете,
5. Выполнить процедуру операции сложения двух трехразрядных чисел без знака,
6. Собрать операционный автомат в среде Electronics Workbench, используя модели регистров и сумматора, имеющиеся в библиотеке системы.
7. Промоделировать выполнение арифметических процедур на операционном автомате, путем подачи управляющих воздействий на соответствующие входы регистров и сумматора,
8. Подготовить отчет с описанием результатов выполненного задания и ответы на вопросы.

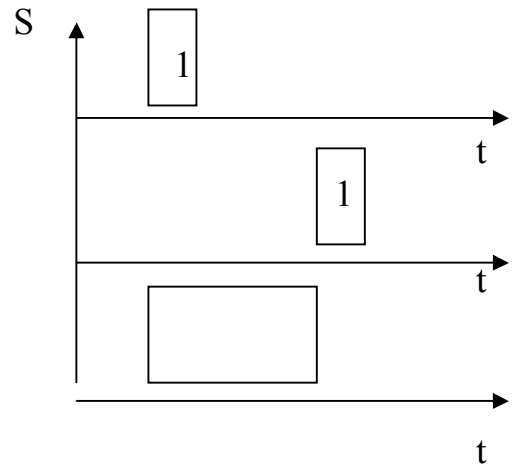
Пример синтеза RS-триггера.
Синтезировать логическую схему триггера.
последовательность действий:

- уяснить задачу
- составить таблицу истинности
- получить булеву функцию
- минимизировать булеву функцию
- взять заданный базис (и - не либо или - не)
- составить логическую схему триггера
- произвести проверку
- изобразить эпюры сигналов

асинхронный RS – триггер.

S – SET (установка);
R – RESET (сброс);
Q(t) – состояние триггера;
Таблица истинности.

R	S	Q(t)	Q(t+1)	
0	0	0	0	} т. к. на входе ничего нет.
0	0	1	1	
0	1	0	1	} установка единицы.
0	1	1	1	
1	0	0	0	} установка нуля.
1	0	1	0	
1	1	0	-	} не используются.
1	1	1	-	



Эпюры выходных сигналов

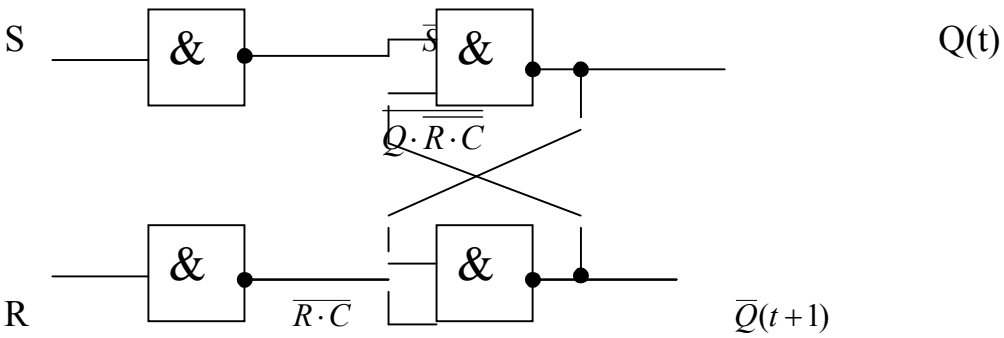
Методом карт Карно определяем булеву функцию:

S \ R				
	- (6)	- (7)	1 (3)	1 (2)
	0 (4)	0 (5)	0 (1)	1 (0)

Q

$$Q\,(t+1) = S + \overline{R} \cdot \overline{Q};$$

Выберем базис на элементах И – НЕ:

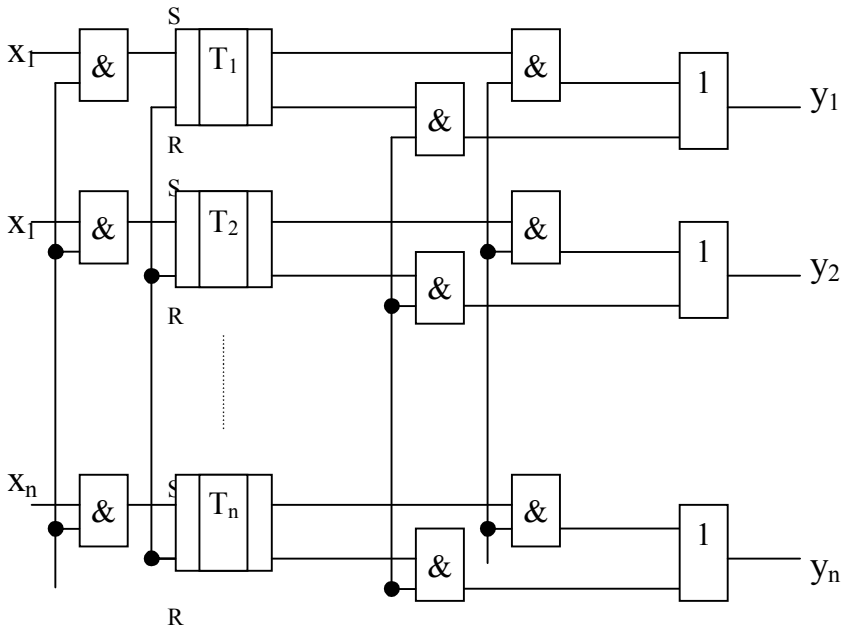


Проверка работы схемы:

C=0; S=0; R=0; Q(t)=0; Q(t+1)=0.

Синтезировать параллельный регистр, схема которого представлена на рисунке

Параллельный регистр



сброс считывание в прямом коде

запись считывание в обратном коде

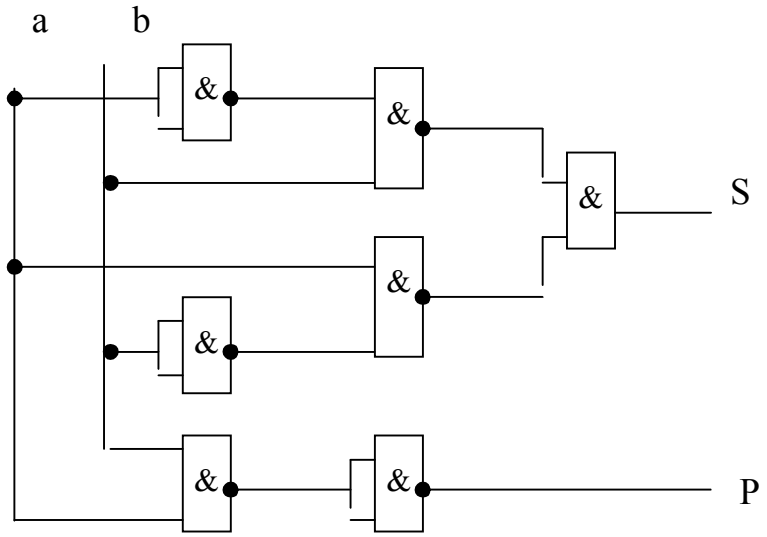
T_n – однотактный RS - триггер.

) Сумматор

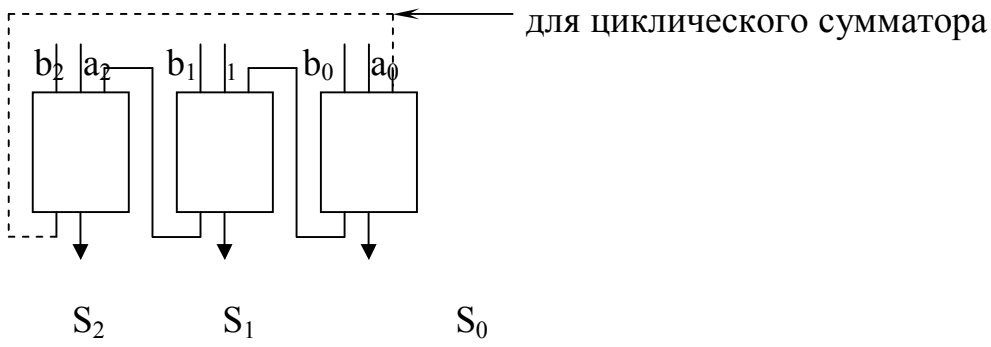
a	b	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{\overline{a \cdot b + a \cdot b}} = \overline{\overline{a \cdot b} \cdot \overline{a \cdot b}};$$
$$P = a \cdot b;$$

Одноразрядный сумматор



Трёхразрядный сумматор



Выполнение арифметических операций

Двоичная арифметика

Правила выполнения арифметических действий над двоичными числами определяются арифметическими действиями над одноразрядными двоичными числами.

	Сложение	Вычитание	Умножение
	$0+0=0$	$0-0=0$	$0*0=0$
	$0+1=1$	$1-0=0$	$1*0=0$
	$1+0=1$	$1-1=0$	$0*1=0$
	$1+1=10$	$10-1=1$	$1*1=1$
	↑↵		
	перенос в старший разряд		

Правила выполнения арифметических действий во всех позиционных системах счисления аналогичны.

Сложение

Как и в десятичной системе счисления, сложение двоичных чисел начинается с правых (младших) разрядов. Если результат сложения цифр МЗР обоих слагаемых не помещается в этом же разряде результата, то происходит перенос. Цифра, переносимая в соседний разряд слева, добавляется к его содержимому. Такая операция выполняется над всеми разрядами слагаемых от МЗР до СЗР.

Вычитание

Операция вычитания двоичных чисел аналогична операции в десятичной системе счисления. Операция вычитания начинается, как и сложение, с МЗР. Если содержимое разряда уменьшаемого меньше содержимого одноименного разряда вычитаемого, то происходит заем 1 из соседнего старшего разряда. Операция повторяется над всеми разрядами операндов от МЗР до СЗР.

Второй вариант операции вычитания - когда уменьшаемое меньше вычитаемого. В этом случае отрицательное число представляется в дополнительном коде.

Умножение

Операция перемножения двоичных многоразрядных чисел производится путем образования частичных произведений и последующего их суммирования. Частичные произведения формируются в результате умножения множимого на каждый разряд множителя, начиная с МЗР. Операция умножения состоит в формировании суммы частичных произведений, которые суммируются с соответствующими сдвигами друг относительно друга. Этот процесс суммирования можно начинать либо с младшего, либо со старшего частичного произведения. В ЭВМ формируют одно частичное произведение, к нему с соответствующим сдвигом прибавляют следующее и т.д. (т.е. не формируют все частичные произведения, а потом их складывают). В зависимости от того, с какого частичного произведения начинается суммирование (старшего или младшего), сдвиг текущей суммы осуществляется влево или вправо.

Содержание отчета:

1. Титульный лист
2. Цель работы
3. Ход работы
   - 3.1. RS-триггер
   - 3.2. Многоразрядный сумматор
   - 3.3. Регистр
   - 3.4. Описание арифметической операции, согласно задания,
   - 3.5. Схема операционного автомата.
4. Ответы на вопросы

Вопросы к индивидуальному заданию

1. Принцип академика Глушкова В.М.
2. Какова разрядность управляющих сигналов, поступающих на ОА
3. Основные функции ОА
4. Какие основные микрооперации реализуются в ОА
5. В чем заключается принцип микропрограммирования
6. Каким образом меняется структура ОА для выполнения различных арифметическо-логических операций.

3. Индивидуальное задание № 2 Синтез управляющего автомата (4 часа)

Понять, каким образом выполняются арифметическо - логические операции в микропроцессоре. Научиться синтезировать управляющий автомат (УА) для выполнения арифметических операций сложения и умножения.

Основные понятия

Типы УА:

1.) УА с жёсткой логикой;

2.) УА с программируемой логикой;

УА с жёсткой логикой вырабатывает управляющие воздействия, путем срабатывания триггеров, число которых определяется числом устойчивых состояний в граф-схеме алгоритма, и входными условиями.

В УА с программируемой логикой коды микроопераций выбирается из ячеек ПЗУ, исходя из входных условий,

Пример схемы алгоритма (ГСА) – представлен на рис. Здесь Сч - счетчик циклов.

Операция умножения разделяется на 7 МО, основные из которых сложение (реализуется за один такт комбинационным двоичным сумматором) и сдвиг (реализуется регистром сдвига). Порядок выполнения МО зависит от значений двух ЛУ (осведомительных сигналов): B(0), СЦ=0. ГСА умножения задает порядок выполнения МО и проверки ЛУ во времени. Например, в зависимости от B(0) в следующий момент времени, в следующем такте будет выполняться либо МО сложения C:=C+A (если B(0)=1), либо МО сдвига, если B(0)=0.

Обработка информации с помощью ОУ осуществляется путем выполнения операций из списка F в последовательности, которая задается алгоритмом (программой) решения задачи: ОА, выполняя программу, распределяет выполнение операций, предписанных командами программы, между различными ОУ - АЛУ, контроллерами ПУ.

Запуск (инициализация) операции $f_g \in F$ осуществляется путем подачи кода операции в ОУ из УУ. Реализация операции f_g осуществляется путем выполнения МО в порядке, заданном микропрограммой, хранимой внутри ОУ (т.е. без участия УУ). Работа (функционирование) ОУ во времени осуществляется тактами. Реализация МПg в общем случае занимает различное количество тактов n, т.е. время выполнения операции $t_g = nT$, где T-продолжительность такта.

Принцип микропрограммного управления является основой организации (построения) ОУ. Эти процедуры достаточно схожи с фон Неймановскими принципами функционирования. В основе управления лежит алгоритм. Только у Неймана он представляется в виде макропрограммы и поступает в процессор извне (из ОЗУ извлекается процессором). Здесь алгоритм в виде МП уже находится внутри ОУ. При выполнении программы ЦП генерирует определенную последовательность операций, реализуемых ОУ. При выполнении операции f_g ОУ генерирует последовательность МО, реализуемых комбинационными схемами КС.

Отличия между этими принципами: 1)операция - сложное действие, для реализации которого необходимо ОУ. МО - элементарное действие, для реализации которого достаточно КС. 2)Операция выполняется за n тактов: $t_{опер}=nT$. МО выполняется за один такт (алгоритм – в структуре КС). КС управляется данными на ее входах.

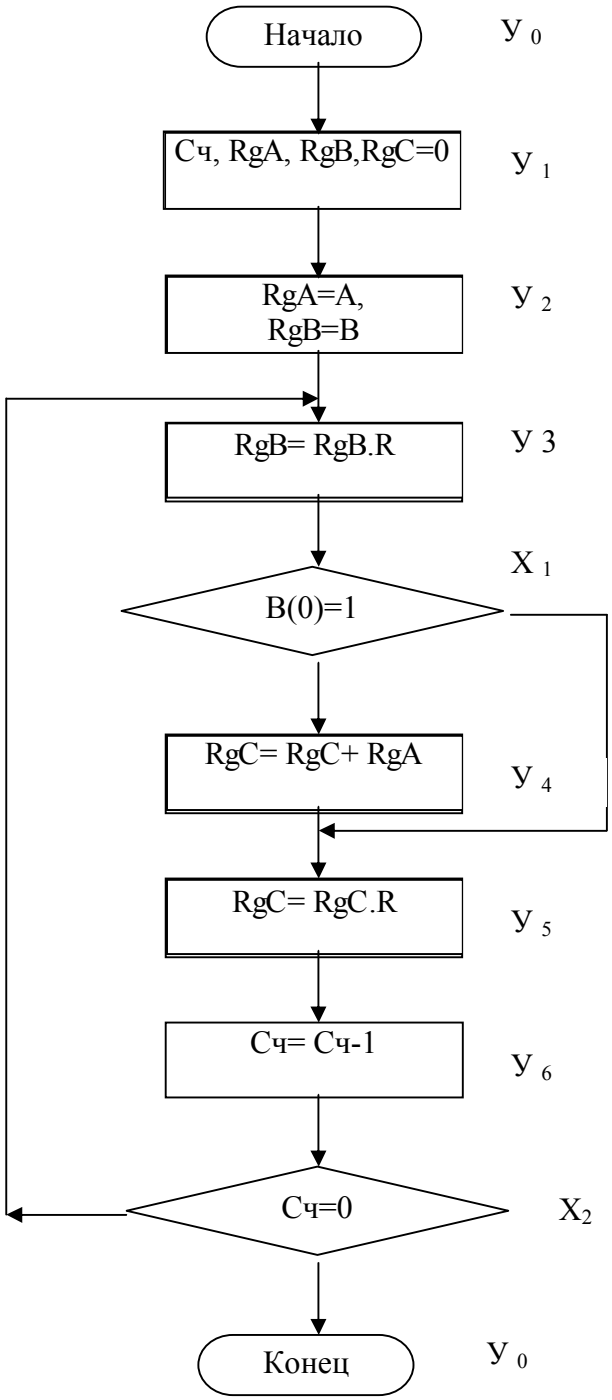


Рис. Граф-схема алгоритма

Пример синтеза регистра со сдвигом.
синтезировать логическую схему регистра;

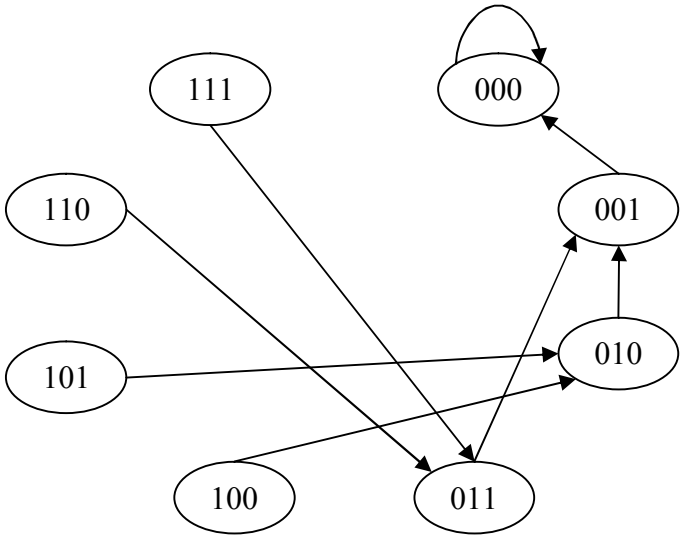
- уяснить задачу;
- составить граф переходов;
- составить таблицу истинности;
- получить булевы функции;
- минимизировать булевы функции;
- взять заданный базис в виде заданных триггеров;
- составить логическую схему;

- произвести проверку.

Наименование регистра	Вид сдвига
D-триггер	Сдвиг вправо

Регистр – логический узел с памятью, состоящий из группы триггеров. Число триггеров соответствует количеству разрядов в слове, которое обрабатывается или запоминается в регистре. Рассмотрим 3-рязрядный регистр со сдвигом вправо. Он состоит из трёх D-триггеров, соединенных таким образом, что каждый тактовый импульс вызывает перемещение содержимого триггера в следующий за ним триггер.

Составляем граф переходов.



Граф переходов

Составляем таблицу истинности.

	x	$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$	D_2	D_1	D_0
8	1	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0	0
10	1	0	1	0	0	0	1	0	0	1
11	1	0	1	1	0	0	1	0	0	1
12	1	1	0	0	0	1	0	0	1	0
13	1	1	0	1	0	1	0	0	1	0
14	1	1	1	0	0	1	1	0	1	1
15	1	1	1	1	0	1	1	0	1	1

По таблице истинности получим булевы функции

$$D_2 = 0$$
$$D_1 = XQ_2\overline{Q_1}\,\overline{Q_0} + XQ_2\overline{Q_1}Q_0 + XQ_2Q_1\overline{Q_0} + XQ_2Q_1Q_0$$
$$D_0 = X\overline{Q_2}Q_1\overline{Q_0} + X\overline{Q_2}Q_1Q_0 + XQ_2Q_1\overline{Q_0} + XQ_2Q_1Q_0$$

С помощью карт Карно минимизируем булевы функции

Q_2 \ x				
	1_{12}	1_{14}	0_{6}	0_{4}
	1_{13}	1_{15}	0_{7}	0_{5}
	0_{9}	0_{11}	0_{3}	0_{1}
	0_{8}	0_{10}	0_{2}	0_{0}

Q_0 / Q_1

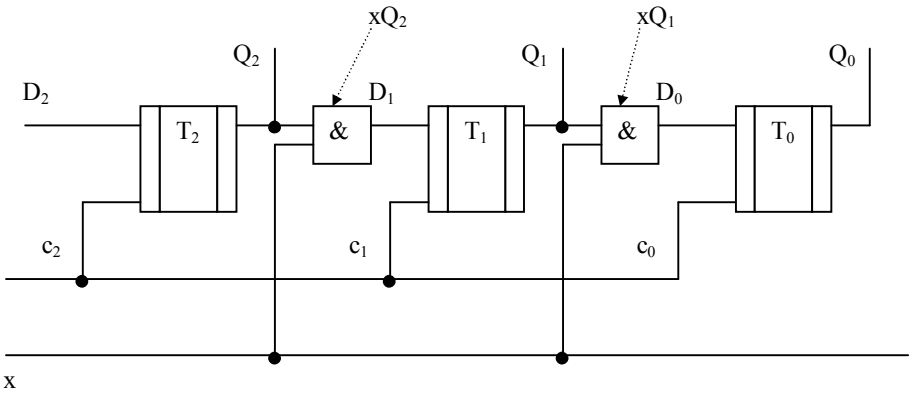
$$D_1 = XQ_2$$

Q_2 \ x				
	0_{12}	1_{14}	0_{6}	0_{4}
	0_{13}	1_{15}	0_{7}	0_{5}
	0_{9}	1_{11}	0_{3}	0_{1}
	0_{8}	1_{10}	0_{2}	0_{0}

Q_0 / Q_1

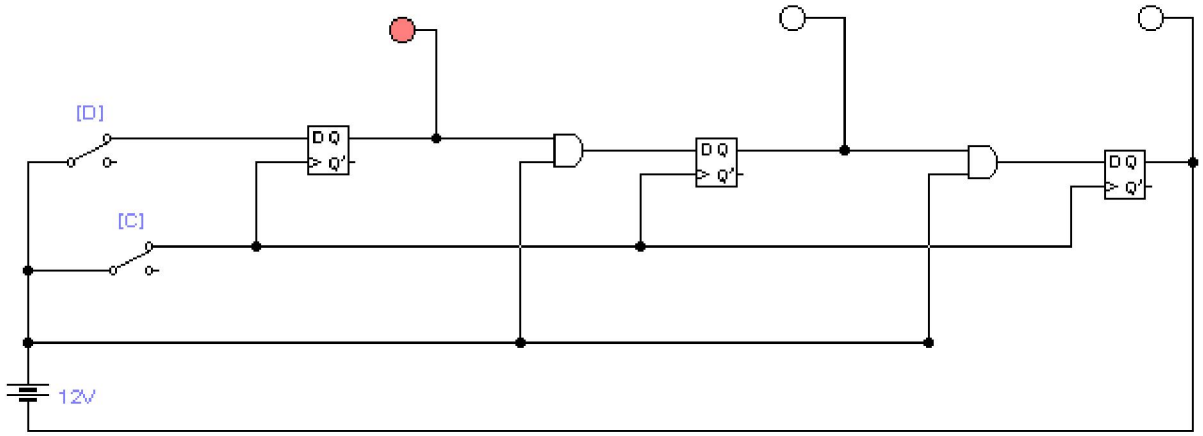
$$D_0 = XQ_1$$

Составим логическую схему



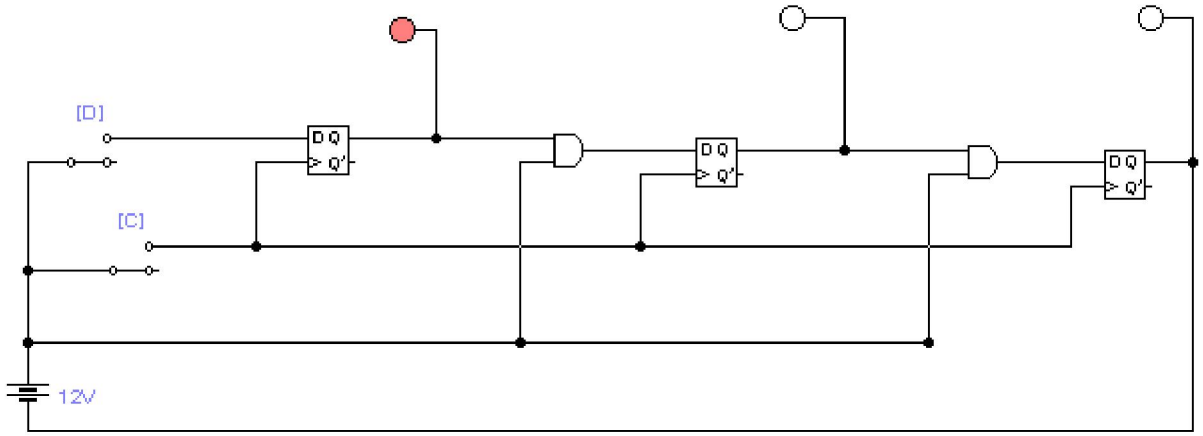
Реализуем логический узел в среде EWB и докажем его работоспособность записыванием и считыванием трехразрядных кодов.

Шаг 1. $Q_2=Q_1=Q_0$.Запишем в регистр трехразрядный код – 100, включив оба ключа D и C ($D_2=1$, $D_1=x\&Q_2=1\&0=0$, $D_0=x\&Q_1=1\&0=0$).



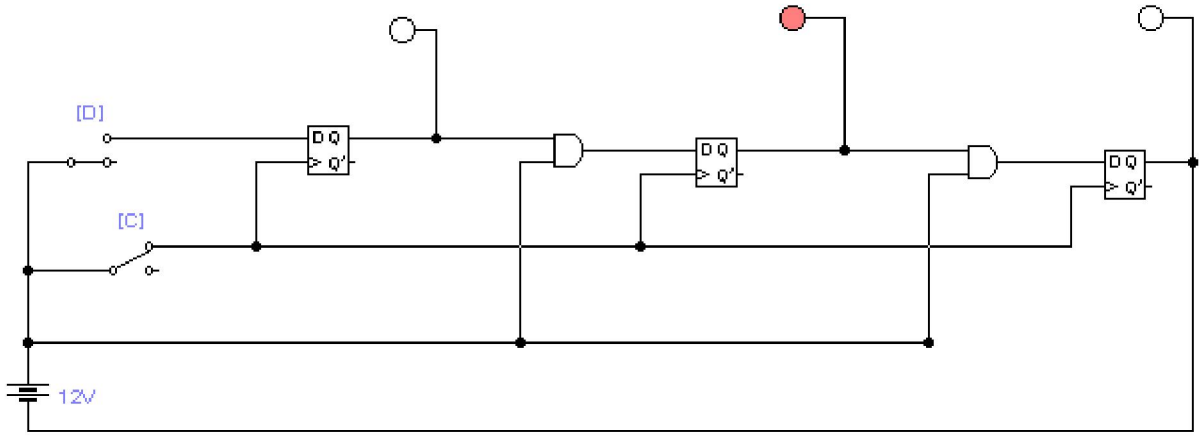
$Q_2(t+1)=1$, при $D_2=1$ и $C_2=1$; $Q_1(t+1)=0$, при $D_1=0$ и $C_1=1$; $Q_0(t+1)=0$, при $D_0=0$ и $C_0=1$.

Шаг 2. Выключим оба ключа (D и C).



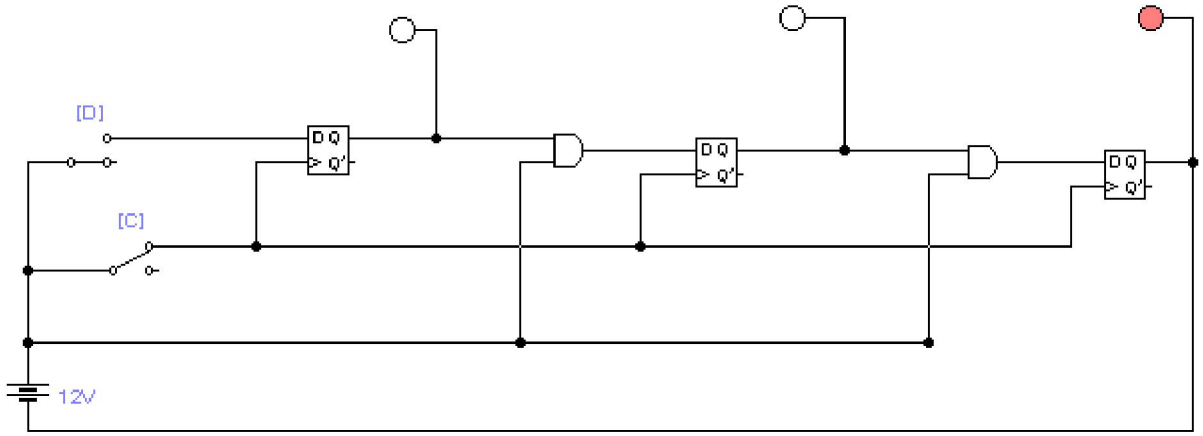
Произошла задержка сигнала, т.е. в регистре сохранился трехразрядный код – 100.

Шаг 3. Включим ключ C, что соответствует тактовому импульсу, вызывающему перемещение содержимого триггера в следующий за ним триггер, при выключенном ключе D ($D_2=0$, $D_1=x\&Q_2=1\&1=1$, $D_0=x\&Q_1=1\&0=0$).



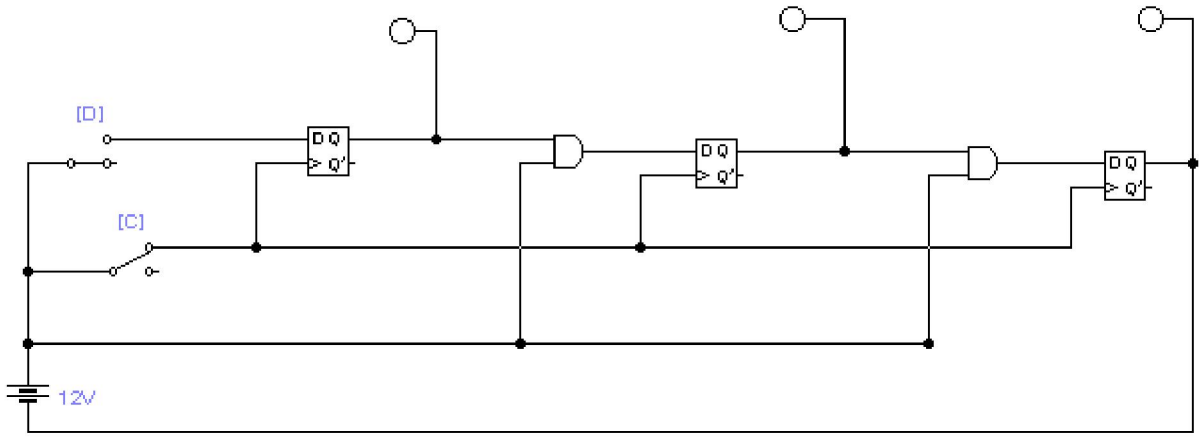
Получили трехразрядный код – 010. $Q_2(t+1)=0$, при $D_2=0$ и $C_2=1$; $Q_1(t+1)=1$, при $D_1=1$ и $C_1=1$; $Q_0(t+1)=0$, при $D_0=0$ и $C_0=1$.

Шаг 4. Выключим и снова включим ключ C, тем самым подав следующий тактовый импульс (D_2=0, D_1=x&Q_2=1&0=0, D_0=x&Q_1=1&1=1).



Получили трехразрядный код – 001. $Q_2(t+1)$=0, при D_2=0 и C_2=1; $Q_1(t+1)$=0, при D_1=0 и C_1=1; $Q_0(t+1)$=1, при D_0=1 и C_0=1.

Шаг 5. Повторим предыдущую операцию (D_2=0, D_1=x&Q_2=1&0=0, D_0=x&Q_1=1&0=0).



Получили трехразрядный код – 000. $Q_2(t+1)$=0, при D_2=0 и C_2=1; $Q_1(t+1)$=0, при D_1=0 и C_1=1; $Q_0(t+1)$=0, при D_0=0 и C_0=1.

В результате выполнения работы синтезирована логическая схема последовательного регистра со сдвигом вправо на D-триггерах. Функционирование синтезированной схемы подтверждает правило изменения трехразрядного кода занесенного в регистр, при подачи тактовых импульсов ($100 \to 010 \to 001 \to 000$), что соответствует построенному графу переходов.

Аналогичным образом необходимо синтезировать управляющий автомат.

3. Порядок выполнения индивидуального задания

Последовательность действий при синтезе УА

1. Уяснить задачу
2. Ознакомиться со способом выполнения арифметической операции
3. Составить ГСА
4. Разметить ГСА
5. Составить граф переходов

6. Выбрать триггер
7. Составить таблицу истинности
8. Получить булевы функции;
9. Минимизировать булевы функции;
10 Взять заданный базис в виде заданных триггеров;
11. Составить логическую схему;
12. Произвести проверку.

Содержание отчета:

4. Титульный лист
5. Цель работы
6. Ход работы
   - 6.1. Триггер со сдвигом
   - 6.2. Описание арифметической операции, согласно задания
   - 6.3. ГСА арифметической операции
   - 6.4. Граф переходов
   - 6.5. Схема управляющего автомата.
   - 6.6. Результаты проверки
4. Ответы на вопросы

Вопросы к индивидуальному заданию

1. В чем отличие УА с жесткой логикой от УА с программируемой логикой.
2. В какой форме представляются сигналы из УА для управления в ОА.
3. Какую роль играет разметка в ГСА
4. Почему D- триггер предпочтительнее применять в УА с жесткой логикой.
5. Чем конечный автомат Мили отличается от автомата Мура.
6. Чем определяется количество триггеров в УА с жесткой логикой.

4. Индивидуальное задание.**№ 3. Программирование ЭВМ в машинных кодах (2 часа)**

Программирование в машинных кодах. Системный отладчик и дизассемблер Debug.

Изучение программирования компьютера IBM в машинных кодах с использованием возможностей системного отладчика DEBUG. Для выполнения задания необходимо предварительно ознакомиться с описанием DEBUG.

Основные понятия

Необходимо использовать системную программу DEBUG, позволяющую вводить программы, осуществлять трассировку их выполнения, просматривать память. В этой работе изучается процесс ввода программы непосредственно в память в область сегмента кодов и объясняется каждый шаг ее выполнения. Изучаются команды отладчика DEBUG.

Для запуска этой программы наберите td.exe и нажмите Enter, в результате программа DEBUG должна загрузиться с диска в память. После окончания загрузки на экране появится главное окно, что свидетельствует о готовности программы DEBUG для приема команд.

1. Проверка размер доступной для работы памяти. В зависимости от модели компьютера это значение связано с установкой внутренних переключателей и может быть меньше, чем реально существует. Данное значение находится в ячейках памяти 413h и 414h и его можно просмотреть из DEBUG по адресу, состоящему из двух частей:
400 это адрес сегмента, который записывается как 40 (последний нуль подразумевается) и 13 это смещение от начала сегмента.

Таким образом, можно ввести следующий запрос:
D 40:13 (и нажать Enter).

Первые два байта, появившиеся в результате на экране, содержат объем памяти в килобайтах и в шестнадцатеричном представлении, причем байты располагаются в обратной последовательности.

Несколько следующих примеров показывают шестнадцатеричное обратное, шестнадцатеричное нормальное и десятичные представления.

Шестн. обратн.	Шестн. норм.	Десятичное
80 00	00 80	128
00 01	01 00	256
80 01	01 80	384
00 02	02 00	512
80 02	02 80	640

2. Серийный номер компьютера "зашит" в ROM по адресу FEOOOh. Для его выделения следует ввести:

D FEOO:0
(и нажать Enter)
В результате на экране появится семизначный номер компьютера и дата копирайт.
Дата ROM BIOS в формате мм/дд/гг (месяц,день,год) находится по адресу FFFF5h.

Введите

D FFFF:05 (и нажмите Enter)

Знание этой информации (даты) иногда бывает полезным для определения модели и возраста компьютера.
3. Установка адреса любой ячейки памяти для просмотра ее содержимого. Можно также пролистывать память, периодически нажимая клавишу D, DEBUG выведет на экран адреса, следующие за последней командой.
Для окончания работы и выхода из отладчика в DOS введите команду Q (Quit).
Для иллюстрации ввода программ в память, рассмотрим в качестве примера простую программу на машинном языке, ее представление в памяти и результаты ее выполнения. Программа показана в шестнадцатеричном формате:

Команда	Назначение
B82301	Переслать значение 0123h в AX
052500	Прибавить значение O025h к AX
8BD8	Переслать содержимое AX в BX
03D8	Прибавить содержимое AX к BX
88CB	Переслать содержимое BX в CX
2BC8	Вычесть содержимое AX из AX (очистка AX)
90	Нет операции
CB	Возврат в DOS.

Машинные команды имеют различную длину: один, два или три байта. Машинные команды находятся в памяти непосредственно друг за другом. Выполнение программы начинается с первой команды, и далее последовательно выполняются остальные.

Можно ввести эту программу непосредственно в память машины и выполнить ее покомандно. Одновременно на каждом шаге можно просматривать содержимое регистров после выполнения каждой команды.

Введем команду запуска отладчика DEBUG и нажмем клавишу Enter. После загрузки DEBUG на экране высвечивается приглашение к вводу команд в виде дефиса.

Для непосредственного ввода программы на машинном языке введите следующую команду, включая пробелы:

E CS:100 B8 23 01 05 25 00 (нажмите Enter)

Команда E обозначает Enter (ввод). Параметр CS:100 определяет адрес памяти, куда будут вводиться команды, шест. 100 (256) байт от начала сегмента кодов (обычный стартовый адрес для машинных кодов в отладчике DEBUG). Команда E записывает каждую пару шестнадцатеричных цифр в память в виде байта, начиная с адреса CS:100 до адреса CS:105.

Следующая команда Enter

E CS:106 8B D8 03 D8 8B CB (Enter)

вводит шесть байтов в ячейки, начиная с адреса CS:106 и далее в 107, 108, 109, 10A и 10B. Последняя команда Enter:

E CS:10C 2B C8 2B C0 90 CB (Enter)

вводит шесть байтов, начиная с CS:10C в 10D,10E, 10F, 110 и 111. Проверьте правильность ввода значений. Если имеются ошибки, то следует повторить команды, которые были введены неправильно. Для выполнения машинных команд используйте команду T. Введите команду R для просмотра содержимого регистров и флагов. В данный момент отладчик покажет содержимое регистров в шестнадцатеричном формате, например,

AX = 0000, BX = 0000, ...

Содержимое регистра IP (указатель команд) выводится в виде IP = 0100, показывая, что выполняемая команда находится на смещении 100h байт от начала сегмента кодов. (Вот почему использовалась команда E CS:100 для установки начала программы.) Регистр флагов показывает следующие значения флагов:

NV UP DI PL NZ NA PO NC.

Данные значения соответственно показывают: нет переполнения, правое направление, прерывания запрещены, знак плюс, не ноль, нет внешнего переноса, контроль на четность и нет переноса. В данный момент значение флагов несущественно. Команда R показывает также по смещению 0100h первую выполняемую машинную команду, которая выглядит следующим образом:

13C6:0100 B82301 MOV AX, 0123.

CS = 13C6 указывает, что начало сегмента кода находится по смещению 13C6 или, точнее, 13C60. Значение 13C6:01OO обозначает 100 (шестн.) байт от начального адреса 13C6 в регистре CS.

B8201 машинная команда, введенная по адресу CS:100. MOV AX,0123 ассемблерный мнемонический код соответствующий введенной машинной команде. Это есть результат операции дисассемблирования, которую обеспечивает отладчик для более простого понимания машинных команд. Рассматриваемая в данном случае команда обозначает пересылку непосредственного значения 0123 в регистр AX.

В данный момент команда MOV еще не выполнена. Для ее выполнения нажмите клавишу T (для трассировки) и клавишу Enter . В результате команда MOV будет выполнена и отладчик выдаст на экран содержимое регистров, флаги, а также следующую на очереди команду. Заметим, что регистр AX теперь содержит значение 0123. Машинная команда пересылки в регистр AX имеет код B8, за которым следуют непосредственные данные 2301. В ходе выполнения команда B8 пересылает значение 23 в младшую часть регистра AX, в AL, а значение 01 в старшую часть регистра AX, т. е. в регистр AH.

Содержимое регистра IP:0103 показывает адрес следующей выполняемой команды в сегменте кодов:

13C6:0103 052500 ADD AX,0025

Для выполнения данной команды снова введите T. Команда прибавит 25H к младшей (AL) части регистра AX и 00 к старшей (AH) части регистра AX, т.е прибавит 0025 к регистру AX. Теперь регистр AX содержит 0148, а регистр IP 0106 адрес следующей команды для выполнения. Введите снова команду T. Следующая машинная команда пересылает содержимое регистра AX в регистр BX, и после ее выполнения в регистре BX будет содержаться значение 0148. Регистр AX сохраняет прежнее значение 0148, поскольку команда MOV только копирует данные из одного места в другое. Теперь вводите команду T для пошагового выполнения каждой оставшейся в программе команды. Следующая команда прибавит содержимое регистра AX к содержимому регистра BX, в последнем получим 0290. Затем программа скопирует содержимое регистра BX в CX, вычтет AX из CX и вычтет AX из него самого. После этой последней команды флаг нуля изменит свое состояние с NZ (не нуль) на ZR (нуль), так как результатом этой команды является нуль (вычитание AX из самого себя обнуляет этот регистр).

Можно ввести T для выполнения последних команд NOP и RET, но это будет сделано позже. Для просмотра программы в машинных кодах в сегменте кодов введите D:

D CS:100

В результате отладчик выдаст на каждую строку экрана по 16 байт данных в шестнадцатиричном представлении (32 шестнадцатиричные цифры) и в символьном представлении в коде ASCII (один символ на каждую пару шестнадцатиричных цифр). Представление машинного кода в символах ASCII не имеет смысла и может быть игнорировано.

Первая строка дампа начинается с 00 и представляет содержимое ячеек от CS:100 до CS:10F. Вторая строка представляет содержимое ячеек от CS:110 до CS:11F. Несмотря на то, что ваша программа заканчивается по адресу CS:111, команда Dump автоматически выдаст на восьми строках экрана дамп с адреса CS:100 до адреса CS:170.
При необходимости повторить выполнение этих команд сбросьте содержимое регистра IP и повторите трассировку снова. Введите R IP, введите 100, а затем необходимое число команд T. После каждой команды нажимайте клавишу Enter.

Для завершения работы с программой DEBUG введите Q (Quit выход). В результате произойдет возврат в DOS.

4. В предыдущем примере использовались непосредственные данные, описанные непосредственно в первых двух командах (MOV и ADD). Рассмотрим аналогичный пример, в котором значения 0123 и 0025 определены в двух полях сегмента данных. Следующий пример показывает, как компьютер обеспечивает доступ к данным посредством регистра DS и адресного смещения.
В этом примере определены области данных, содержащие соответственно следующие значения:

Адрес в DS	Шестн. значения	Номера байтов
0000	2301	0 и 1
0002	2500	2 и 3
0004	0000	4 и 5
0006	2A2A2A	6, 7 и 8

Так как, шестнадцатеричный символ занимает половину байта, то 23 находится в байте 0 (в первом байте) сегмента данных, 01 в байте 1 (во втором байте). Ниже показаны команды машинного языка, которые обрабатывают эти данные:

Команда	Назначение
A10000	Переслать слово (два байта), начинающееся в DS по адресу 0000, в регистр AX.
03060200	Прибавить содержимое слова (двух байт), начинающегося в DS по адресу 0002, к регистру AX.
A30400	Переслать содержимое регистра AX в слово, начинающееся в DS по адресу 0004.
CB	Вернуться в DOS.

Обратите внимание, что здесь имеются две команды MOV с различными машинными кодами: A1 и A3. Фактически машинный код зависит от регистров, на которые имеется ссылка, количества байтов (байт или слово), направления передачи данных (из регистра или в регистр) и от ссылки на непосредственные данные или на память.

Воспользуемся опять отладчиком DEBUG для ввода данной программы и анализа ее выполнения. Когда отладчик выдал свое дефисное приглашение, он готов к приему команд. Сначала введите команды E (Enter) для сегмента данных:

E DS: 00 23 01 25 00 00 00 (Нажмите Enter)

E DS: 06 2A 2A 2A (Нажмите Enter)

Первая команда записывает три слова (шесть байтов) в начало сегмента данных DS:00. Заметьте, что каждое слово вводилось в обратной последовательности, так что 0123 есть 2301, а 0025 есть 2500. Когда команда MOV будет обращаться к этим словам, нормальная последовательность будет восстановлена и 2301 станет 0123, а 2500 0025.

Вторая команда записывает три звездочки (***) для того, чтобы их можно было увидеть впоследствии по команде D (Dump) другого назначения эти звездочки не имеют.

Введем теперь команды в сегмент кодов, опять начиная с адреса CS:100:

E CS:100 A1 00 00 03 06 02 00
E CS:107 A3 04 00 CB

Эти команды теперь находятся в ячейках памяти от CS:100 до CS:10A. Команды можно выполнить так же, как это делалось ранее.

Для просмотра информации в сегменте данных и в сегменте кодов введите команды D (Dump) соответственно

1. для сегмента данных: D DS:0000 (Enter)
2. для сегмента кодов: D CS:100 (Enter)

Теперь введите R для просмотра содержимого регистров и флагов и для отображения первой команды. Регистры содержат те же значения, что и в первом примере. Команда отобразится в виде 13C6:0100 A10000 MOV AX,[0000] Так как регистр CS содержит 13C6, то CS:100 содержит первую команду A10000. Отладчик интерпретирует эту команду как MOV и определяет ссылку к первому адресу [0000] в сегменте данных. Квадратные скобки необходимы для указания ссылки к адресу памяти, а не к непосредственным данным. Если бы квадратных скобок не было, то команда MOV AX,0000 обнулила бы регистр AX непосредственным значением 0000.

Теперь введем команду T. Команда MOV AX, [0000] перешлет содержимое слова, находящегося по нулевому смещению в сегменте данных, в регистр AX. Содержимое 2301 преобразуется командой в 0123 и помещается в регистр AX. Следующую команду ADD можно выполнить введением еще раз команды T. В результате содержимое слова в DS по смещению 002 прибавится в регистр AX. Теперь регистр AX будет содержать сумму 0123 и 0025, т.е. 0148.

Следующая команда MOV [0004],AX выполняется опять по вводу T. Эта команда пересылает содержимое регистра AX в слово по смещению 0004. Для просмотра изменений содержимого сегмента данных введите D DS:00. Первые девять байт будут следующими:

Значение в сегменте данных: 23 01 25 00 48 01 2A 2A 2A

Величина смещения: 00 01 02 03 04 05 06 07 08

Значение 0148, которое было занесено из регистра AX в сегмент данных по смещению

04 и 05, имеет обратное представление 4801. Заметьте, что эти шестнадцатиричные значения представлены в правой части экрана их символами в ASCII-коде. Например, 23h генерируется в символ #, а 25h в символ %. Три байта с значениями 2Ah - высвечиваются в виде трех звездочек (***). Левая часть дампа показывает действительные машинные коды, которые находятся в памяти. Правая часть дампа только помогает проще локализовать символьные (строчные) данные.

Для просмотра содержимого сегмента кодов введите D DS:100.

Для завершения работы с программой введите Q.

5. Для доступа к машинной команде процессор определяет ее адрес из содержимого регистра CS плюс смещение в регистре IP. Например, пусть регистр CS содержит 04AFh (действительный адрес 04AF0), а регистр IP содержит 023h: CS: 04AF0 IP: 0023 Адрес команды: 04B13 Если, например, по адресу 04B13 находится команда: A11200 MOV AX,[0012], то в памяти по адресу 04B13 содержится первый байт команды. Процессор получает доступ к этому байту и по коду команды (A1) определяет длину команды три байта.

Для доступа к данным по смещению [0012] процессор определяет адрес, исходя из содержимого регистра DS (как правило) плюс смещение в операнде команды. Если DS содержит 04B1h (реальный адрес 04B10), то результирующий адрес данных определяется следующим образом: DS: 04B10 Смещение: 0012 Адрес данных: 04B22. Предположим, что по адресам 04B22 и 04B23 содержатся следующие данные: Содержимое: 24 01 Адрес: 04B22 04B23 Процессор выбирает значение 24 из ячейки по адресу 04B22 и помещает его в регистр AL, а значение 01 по адресу 04B23 в регистр AH. Регистр AX будет содержать в результате 0124. Во время выборки каждого байта команды процессор увеличивает значение регистра IP на единицу, так что к началу выполнения следующей команды в нашем примере IP будет содержать смещение 0026. Таким образом, процессор теперь готов для выполнения следующей команды, которую он получает по адресу из регистра CS [04AF0] плюс текущее смещение в регистре IP (0026), т.е. 04B16.

Процессоры 8086, 80286 и 80386 действуют более эффективно, если в программе обеспечивается доступ к словам, расположенным по четным адресам. В предыдущем примере процессор мог сделать выборку слова по адресу 4B22 для загрузки его непосредственно в регистр за одно обращение к памяти. Но если слово начинается на нечетном адресе, процессор производит два обращения к памяти.

Пусть команда должна выполнить выборку слова, начинающегося по адресу 04B23, и загрузить его в регистр AX: Содержимое памяти: |xx| 24 | 01 | xx | Адрес: 04B23 04B24 04B25 Сначала процессор получает доступ к байтам по адресам 4B22 и 4B23 и пересылает байт из ячейки 4B23 в регистр AL. Затем он получает доступ к байтам по адресам 4B24 и 4B25 и пересылает байт из ячейки 4B23 в регистр AH. В результате регистр AX будет содержать 0124.

Команды обращения к памяти меняют слово при загрузке его в регистр так, что получается правильная последовательность байтов, и, если программа имеет частый

доступ к памяти, то для повышения эффективности можно определить данные так, чтобы они начинались по четным адресам.

Например, поскольку начало сегмента должно всегда находиться по четному адресу, первое поле данных начинается также по четному адресу, и пока следующие поля определены как слова, имеющие четную длину, они все будут начинаться на четных адресах.

Ассемблер имеет директиву EVEN, которая вызывает выравнивание данных и команд на четные адреса памяти.

3. Порядок выполнения индивидуального задания

В первом упражнении этого занятия проводилась проверка объема памяти RAM, которую имеет компьютер. Базовая система ввода-вывода (BIOS) имеет в ROM подпрограмму, которая определяет размер памяти. Можно обратиться в BIOS по команде INT, в данном случае по прерыванию 12h. В результате BIOS возвращает в регистр AX размер памяти в килобайтах. Загрузите в память отладчик DEBUG и введите для команд INT 12h и RET следующие машинные коды:

E CS: 100 CD 12 CB

Нажмите R (и Enter)

для отображения содержимого регистров и первой команды. Регистр IP содержит 0100, при этом высвечивается команда INT 12h. Теперь нажмите T (и Enter) несколько раз и просмотрите выполняемые команды BIOS (отладчик показывает мнемокоды, хотя в действительности выполняются машинные коды):
STI

PUSH DS

MOV AX,0040

MOV DS,AX

MOV AX,[0013]

POP DS

IRET

В этот момент регистр AX содержит размер памяти в шестнадцатеричном формате. Теперь введите еще раз команду T для выхода из BIOS и возврата в вашу программу. На экране появится команда RET для машинного кода CB, который был введен вами.

В операционной системе DOS версии 2.0 и старше можно использовать программу DEBUG для ввода команд ассемблера так же, как и команд машинного языка. На практике можно пользоваться обоими режимами.

Команда отладчика A (Assemble) переводит DEBUG в режим приема команд ассемблера и преобразования их в машинные коды. Установим начальный адрес следующим образом:

A 100 [Enter]

Отладчик выдаст значение адреса сегмента кодов и смещения в виде xxxx:0100. Теперь можно вводить каждую команду, завершая клавишей Enter. Когда вся программа будет введена, нажмите снова клавишу Enter для выхода из режима ассемблера. Введите следующую программу:

MOV AL,25 [Enter]

MOV BL,32 [Enter]

ADD AL,BL [Enter]

RET [Enter]

По завершении на экране дисплея будет следующая информация:

xxxx:0100 MOV AL,25

xxxx:0102 MOV BL,32

xxxx:0104 ADD AL,BL

xxxx:0106 RET

В этот момент отладчик готов к приему следующей команды. При нажатии Enter операция ввода будет прекращена.

Можно видеть, что отладчик определил стартовые адреса каждой команды. Прежде чем выполнить программу, проверим сгенерированные машинные коды.

Команда отладчика U (Unassemble) показывает машинные коды для команд ассемблера. Необходимо сообщить отладчику адреса первой и последней команд, которые необходимо просмотреть, в данном случае 100 и 106. Введите:

U 100,106 [Enter]

На экране дисплея появится

xxxx:0100 B025 MOV AL,25

xxxx:0102 B332 MOV BL,32

xxxx:0104 00D8 ADD AL,BL

xxxx:0106 C3 RET

Проведем трассировку выполнения программы, начиная с команды R, для вывода содержимого регистров и первой команды программы. С помощью команд T выполним последовательно все команды программы.

Ввод на языке ассемблера обычно используется, когда машинный код неизвестен, а ввод в машинном коде для изменения программы во время выполнения. Однако в действительности программа DEBUG предназначена для отладки программ, и в последующем, основное внимание будет уделено использованию языка ассемблера.

Можно использовать DEBUG для сохранения программ на диске в следующих случаях:

а) после загрузки программы в память машины и ее модификации необходимо сохранить измененный вариант. Для этого следует:
-загрузить программу по ее имени:
DEBUG имя файла [Enter],
просмотреть программу с помощью команды D и
ввести изменения по команде E,
записать измененную программу:
W [Enter];

б) необходимо с помощью DEBUG написать небольшую по объему программу и сохранить ее на диске. Для этого следует:
вызвать отладчик DEBUG,
с помощью команд A (assemble) и E (Enter) написать программу, присвоить программе имя:
N имя файла.COM [Enter].
Тип программы должен быть COM;
указать отладчику длину программы в байтах.
В последнем примере концом программы является команда
xxxx:0106 C3 RET
Эта команда однобайтовая, и поэтому размер программы бу
дет равен 107 (конец) минус 100 (начало), т.е. 7.

в) запросить регистр CX командой R CX [Enter], отладчик
выдаст на этот запрос CX:0000 (нулевое значение),
указать длину программы 7,
записать измененную программу: W [Enter].

В обоих случаях DEBUG выдает сообщение "Writing nn bytes". Если nn равно 0, то произошла ошибка при вводе длины программы, и необходимо повторить запись снова.

Отладчик DOS DEBUG это достаточное мощное средство, полезное для отладки ассемблерных программ. Однако следует быть осторожным при ее использовании, особенно для команды E (Enter).

Ввод данных в неправильные адреса памяти или ввод некорректных данных могут привести к непредсказуемым результатам. На экране в этом случае могут появиться "странные" символы, клавиатура заблокирована или даже DOS прервет DEBUG и перезагрузит себя с диска. Какие-либо серьезные повреждения вряд ли произойдут, но возможны некоторые неожиданности, а также потеря данных, которые вводились при работе с отладчиком.

Если данные, введенные в сегмент данных или сегмент кодов, оказались некорректными, следует, вновь используя команду E, исправить их. Однако, можно не заметить ошибки и начать трассировку программы. Но и здесь возможно еще использовать команду E для изменений. Если необходимо начать выполнение с первой команды, то следует установить в регистре командного указателя (IP) значение 0100. Введите команду R (Register) и требуемый регистр в следующем виде:

R IP [Enter]

Отладчик выдаст на экран содержимое регистра IP и перейдет в ожидание ввода. Здесь следует ввести значение 0100 и нажать для проверки результата команду R (без IP). Отладчик выдаст содержимое регистров, флагов и первую выполняемую команду. Теперь можно, используя команду T, вновь выполнить трассировку программы.

Если ваша программа выполняет какие-либо подсчеты, то, возможно, потребуется очистка некоторых областей памяти и регистров. Но убедитесь в сохранении содержимого регистров CS, DS, SP и SS, которые имеют специфическое назначение.

4. Требования к отчету

Отчет по индивидуальному заданию должен содержать:

а) титульный лист;
б) условие задания;
в) текст программ на языке Ассемблера;
г) ответы на контрольные вопросы.

5. Перечень вариантов индивидуального задания

1. Напишите машинные команды для:

а) пересылки значения 4629h в регистр AX;
б) сложения 036Ah с содержимым регистра AX.

2. Предположим, что была введена следующая E-команда:
E CS:100 B8 45 01 05 25 00
Вместо шестьнацатиричных . значения 45 предполагалось 54. Напишите команду E для корректировки только одного неправильно введенного байта, т.е. непосредственно замените 45 на 54.

3. Введена следующая команда:
E CS:100 B8 04 30 05 00 30 CB
Ответьте на вопросы

а) Что представляют собой эти команды?
б) После выполнения этой программы в регистре AX должно быть значение 0460, но в действительности оказалось 6004. В чем ошибка и как ее исправить?
в) После исправления команд необходимо снова выполнить программу с первой команды. Какие две команды отладчика потребуются ?

4. В машинных кодах имеется программа:
B0 25 D0 E0 B3 15 F6 E3 CB
Программа выполняет следующее:

- пересылает значение 25h в регистр AL;
- сдвигает содержимое регистра AL на один бит влево (в результате в AL будет 4A);
- пересылает значение 15h в регистр BL;
- умножает содержимое регистра AL на содержимое регистра BL.

Используйте отладчик для ввода (E) этой программы по адресу CS:100. Не забывайте, что все значения представлены в шестнадцатеричном виде. После ввода программы наберите D CS:100 для просмотра сегмента кода. Затем введите команду R и необходимое числа команд T для пошагового выполнения программы до команды RET. Какое значение будет в регистре AX в результате выполнения программы?

5. Используйте отладчик для ввода (E) следующей программы в машинных кодах
Данные: 25 15 00 00 код:

A0 00 00 D0 E0 F6 26 01 00 A3 02 00 CB
Программа выполняет следующее:

- пересылает содержимое одного байта по адресу DS:00 (25) в ре гистр AL;
- сдвигает содержимое регистра AL влево на один бит (получая в результате 4A);
- умножает AL на содержимое одного байта по адресу DS:01 (15);
- пересылает результат из AX в слово, начинающееся по адресу DS:02.

После ввода программы используйте команды D для просмотра сегмента данных и сегмента кода. Затем введите команду R и необходимое число команд T для достижения конца программы (RET). В этот момент регистр AX должен содержать результат 0612. Еще раз используйте команду D DS:00 и обратите внимание на то, что по адресу DS:02 значение записано как 1206.

6. Для предыдущего задания (5) постройте команды для записи программы на диск под именем TRIAL.COM.

7. Используя команду A отладчика, введите следующую программу:

```
MOV BX,25
ADD BX,30
SHL BX,01
SUB BX,22
NOP
RET
```

Сделайте ассемблирование и трассировку выполнения этой программы до команды NOP. Каков результат выполнения этой программы и где он находится? Объясните смысл каждой команды и напишите эту программу в машинных кодах. С помощью отладчика DEBUG выполните зти машинные коды с начального адреса CS:160 и сравните полученные результаты программы в машинных кодах. С помощью отладчика DEBUG выполните зти машинные коды с начального адреса CS:160 и сравните полученные результаты.

Контрольные вопросы

1. В чем назначение отладчика и отличие debugger от turbo debugger
2. В чем суть сегментации памяти.
3. Какие регистры являются регистрами общего назначения.
4. Что такое реальная адресация и виртуалmная память.
5. Какие модели распределения памяти существуют.
6. Какую роль выполняет транслятор.

5. Индивидуальное задание. №4. Исследование обстановки в сети. Функции ping и tracert (2 часа)

Исследование обстановки в сети и изучение функций ping и tracert. Установление закономерностей времени задержки от длины пакета и определение закона случай\ного процесса передачи трафика.

Основные понятия

Для выполнения данного задания необходимо иметь представление о понятиях, приведенных и определенных ниже.

Сообщение – логически законченное количество информации.

Пакет – это количество информации, которое целесообразно передавать по каналу связи. Другими словами, под пакетами подразумеваются более короткие последовательности двоичных символов 0 и 1, которые образуются путем разбиения сообщений, представленного длинной последовательностью таких символов.

Адрес – набор чисел, которые однозначно определяют какой-либо объект: рабочую станцию сети, местонахождение в памяти компьютера или пакет данных, проходящий через сеть.

Буфер – часть памяти, используемая для временного хранения поступающей информации

Сеть – объединение компьютеров посредством каналов связи.

Канал – среда, через которую соединяются узлы в сеть.

Узел – точка сети, в которой обслуживается пользователь или присоединен коммуникационный канал. Термин «узел» иногда используется вместо термина «рабочая станция».

Рабочая станция – персональный компьютер, присоединенный к сети.

Трафик – нагрузка в сети(количество информации, передаваемой в единицу времени).

Сеанс – логическое сетевое соединение между двумя рабочими станциями для обмена данными.

Соединение – линия взаимодействия объектов, расположенных на одном и том же уровне.

Уровень – слой вычислительной сети, выполняющий одну из ее основных задач.

Маршрутизатор – коммуникационный узел, выбирающий маршрут пакета в сети.

Кадр - транспортное средство, передающее пакет.

Использование команды ping для контроля передачи пакетов

Ping - удобная утилита для проверки качества связи с тем или иным узлом сети. Использование этой команды удобно, когда настроена IP-маршрутизация, но связь между машинами не устанавливается. Причина может быть в неправильной настройке как IP-маршрутизации, так и приложений.

В зависимости от операционной системы эта программа работает по разному - в Unix она выдает сообщение о приходе отклика по его приходу, а в MS-Windows ждет заданное время (по умолчанию - одна секунда), и если в течении этого времени ответ

не пришел, то сообщает "Request timed out." и посылает следующий, а запоздавший ответ игнорирует.

Передача пакетов происходит с помощью команды PING.

Использование: ping [-t] [-a] [-n число] [-l размер] [-f] [-i TTL] [-v TOS] [-r число] [-s число] [[-j список узлов] | [-k список узлов]] [-w интервал] список рассылки

Параметры:

-t Отправка пакетов на указанный узел до команды прерывания.

-a Определение адресов по именам узлов.

-n число Число отправляемых пакетов

-l размер Размер буфера отправки.

-f Установка флага, запрещающего фрагментацию пакета.

-i TTL Задание времени жизни пакета(Time to Life)

-v TOS Задание типа службы (Type of Service)

-r число Запись маршрута для указанного числа переходов

-s число Штамп времени для указанного числа переходов

-j список узлов Свободный выбор маршрута по списку узлов

-k список узлов Жесткий выбор маршрута по списку узлов

-w интервал Интервал ожидания каждого ответа в миллисекундах.

Опция time показывает за какое время в тысячных долях секунды 32 байта информации ушли с сервера, дошли до искомого сервера и вернулись обратно.

Если ping работает, то проблема в настройке приложений. Если сообщает "Destignation host unrecable", то ответивший хост не знает, куда передать запрос; =>, проблема в настройке IP-маршрутизации на нем. (Не все версии ping выдают такое сообщение.)

Для доступности машины IP-пакеты должны правильно проходить как туда, так и обратно; отсутствие отклика может говорить о неправильной маршрутизации как в ту, так и в другую сторону.

Использование команды tracert

Утилита - **Tracert** (trace route) - позволяет проследить путь, который проходит информация дойдя до определенного узла. Принцип ее работы в том, что сначала посылается IP-пакет с временем жизни "1", потом "2" и так далее; каждый роутер уменьшает это число на единицу, а по достижении нуля уничтожает пакет и сообщает отправителю об истечении времени жизни пакета - таким образом, мы получаем список роутеров на пути от нас до хоста назначения (у роутеров обычно по несколько IP-номеров; отвечает интерфейс, обращенный в сторону того, кому отвечают). Если же вместо ответа мы получаем звездочки - значит, очередной роутер не отвечает: недоступен из-за обрыва или сильной загрузки сети либо выключен или неправильно настроен.

При ошибке в настройке маршрутизации "туда" хост с ошибкой будет последним доступным, а при ошибке в маршрутизации "обратно" - первым недоступным в цепочке роутеров от нас до хоста назначения. Либо traceroute выявит хост, которого не должно быть в этой цепочке.

Если DNS-сервер недоступен, tracert будет тратить много времени, дожидаясь ответа от него, чтобы сообщеить доменное имя ответившего хоста; ключ "-d" отменяет обращение к DNS. Tracert в MS-Windows в любом случае будет определять доменное имя по IP-адресу.

Использование: tracert [-d] [-h макс число] [-j список узлов] [-w интервал] имя
Параметры:

-d Без определения адресов по именам узлов.
-h макс число Максимальное число переходов при поиске узла.
-j список узлов Свободный выбор маршрута по списку узлов
-w интервал Интервал ожидания каждого ответа в миллисекундах

Команда Tracert выводит листинг ситуации, как это показано в следующей таблице.

Трассировка маршрута к www.tomsk. Ru с максимальным числом переходов 30.

1.	<10мс	<10мс	<10мс	Aoi. Muma. Tuaur.ru
2.	<10мс	<10мс	<10мс	muma. Tasur.edu.ru
3.	<10мс	<10мс	<10мс	fet. Tusur.ru
4.	<10мс	<10мс	<10мс	rk-tsu-10M.rk.tusur.ru
5.	*	*	*	превышен интервал ожидания для запроса
6.	*	*	*	превышен интервал ожидания для запроса
7.	*	*	*	превышен интервал ожидания для запроса
8.	*	*	*	превышен интервал ожидания для запроса
9.	*	*	*	превышен интервал ожидания для запроса
10.	861мс	641мс	480мс	s-e.ocicat.tomsk.ru
11.	811мс	491мс	501мс	

3. **Порядок выполнения индивидуального задания**

Для выполнения задания используется команда PING, которая набирается в командной строке. PING имеет несколько параметров, из которых для достижения поставленной цели можно использовать лишь два: размер буфера, определяющий длину пакета(по умолчанию 32 байта) и число узлов. В результате выполнения команды для каждого заданного размера буфера (L) получится несколько значений времени задержки(Тз), затрачиваемого на передачу пакета. Данная зависимость может быть представлена в следующем виде на рис.

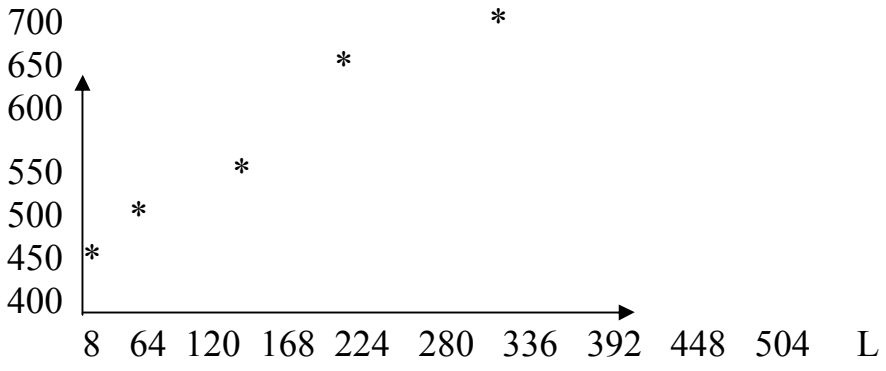


Рис. Зависимость времени задержки от длины пакета

Задав число запросов, и выполнив команду PING, также получится несколько чисел, характеризующих Тз. График возможной зависимости представлен на рис.

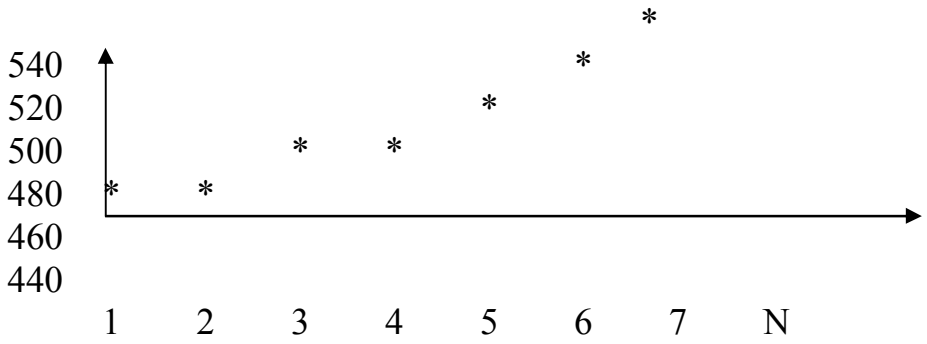


Рис. Зависимость времени задержки от числа пакетов

6. Варианты заданий

1 Получить зависимость времени задержки от длины передаваемого пакета.
2 Получить зависимость времени задержки от числа пакетов
3 Построить графики по данным пунктов 6.1 и 6.2
4 Получить листинг ситуации с помощью команды tracert
5 Снять трафик утром, вечером и днем. Используя статистический подход, определить закон распределения случайного процесса времени задержки от числа пакетов и длины пакетов
6 Сделать соответствующие выводы
7 Ответить на контрольные вопросы
8 Подготовить отчет

7. Контрольные вопросы

1 В чем разница понятий «сообщение» и «пакет» ?
2 Что такое маршрутные таблицы, привести пример маршрутной таблицы.
3 Почему не все пакеты доходят до адресата?
4 Объяснить соединение «точка в точку».
5 Объяснить понятие «пакетная коммутация».
6 В чем преимущество волоконно-оптических каналов связи?

6. Индивидуальное задание №5. Исследование канала связи (2 часа)

Изучение метода оценки качества канала связи на основе потерь энергии при передаче сигналов определенной формы по этому каналу.

Основные понятия

Использование информации потребителей связано с ее транспортировкой от источника. Транспортировка всегда осуществляется в пространстве и во времени. Основные требования к каналам связи заключаются в следующем:

1. Высокая эффективность передачи, обуславливаемая пропускной способностью канала.
2. Надежная передача данных путем использования помехоустойчивого кодирования.
3. Использование перспективных (в основном оптических) каналов связи.

Для повышения эффективности использования каналов связи широко применяется частотное разделение каналов. При этом для различных каналов отводятся непересекающиеся полосы частот Δf_1, Δf_2,…, Δf_n, на частотной шкале. Спектры сигналов соответствующих каналов должны укладываться в пределы Δf_1. Поэтому необходимо изучать спектры сигналов и оценивать какое количество энергии сигналов теряется при несоблюдении ширины спектра сигнала и канала связи.

Информационная модель канала связи состоит из источника информации, передающей оконечной аппаратуры для формирования сигнала с модуляцией, преобразователя кода, канала связи, приемника, декодера, приемной оконечной аппаратуры, потребителя информации.

Наиболее перспективно в качестве линии передачи использовать оптический кабель, по которому может передаваться сигнал. Состоит из сердечника, оболочки, и внешнего покрытия. Сердечник изготовляют из материала с наименьшими потерями, в оболочке допускаются большие потери.

Процент передачи излучения по кабелю осуществляется в результате отражения света от оболочки, имеющий больший коэффициент отражения, чем сердечник.

Рассмотренная выше структура канала является двухточечной (т.е. канала соединяет только два узла). Существует два больших класса двухточечных каналов: цифровые и аналоговые. При модульном подходе цифровой канал является просто битовым трактом с битовыми потоками на входе и выходе. Что касается аналогового сигнала, то на его вход поступает непрерывный сигнал, т.е. произвольная функция либо времени, либо пространства.

При изучении сетей передачи данных основное внимание уделяется аналоговым каналам, хотя и цифровые каналы важны для практики и более перспективны, но их лучше всего можно понять на основе изучения аналоговых каналов.

Для изучения каналов связи лучше всего применять спектральные методы, которые используют частотное представление функции в виде преобразования Фурье во временной или пространственной форме.

При рассмотрении спектров основных видов сигналов главное внимание уделяется определению их ширины, поскольку в основном этот фактор используется для согласования сигнала с аппаротурой обработки информации, причем для

исключения потерь информации ширина спектра сигнала не должна превышать полосы пропускания канала.

Для периодического сигнала $U_t(t)$ спектр определяется соотношением

$$A_K = \frac{2}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} U_t(t) * e^{-jk\omega_t t}\,dt ,$$

где к = ±1, ±2, ±3, …;

$\omega_t = \frac{2\pi}{T}$ – круговая частота.

Спектр периодического сигнала имеет дискретный характер.

Для непериодического сигнала, определяемого на бесконечном интервале времени, преобразование Фурье имеет вид:

$$W(\omega_t) = \int_{-\infty}^{\infty} U(t) e^{-j\omega t}\,dt ,$$

$$U(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} W(j\omega) e^{j\omega t}\,d\omega .$$

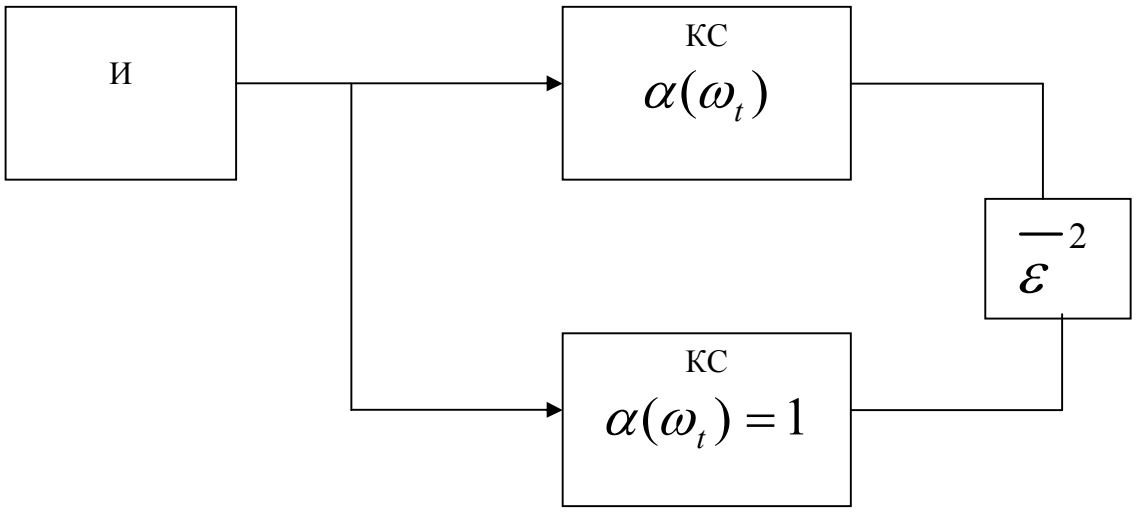


Рис. Схема формирования ошибки на выходе каналов связи,
где И - источник, формирующий сигнал U(t),
$\alpha(\omega_t)$ – передаточная характеристика канала,
ε^2 – потери энергии сигнала связи.

В случае непереодических сигналов рассматривается не спектр сигнала, а его производная функция по $\omega_t = W(j\omega)$, носящего название спектральной плотности. Поэтому спектр непериодической функции имеет непрерывный характер.

Для исследования качества передачи канала связи наиболее целесообразно использовать критерий в виде потерь энергии сигнала.

Для оценки потерь энергии сигнала необходимо найти разность энергий сигналов, прошедших реальный канал с передаточной характеристикой $\alpha(\omega_t)$ и идеальный канал с $\alpha(\omega_t) = 1$. Тогда энергия потерь ε^2 будет иметь вид:

$$\bar{\varepsilon}^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty}|W(\omega_t)|^2 * d\omega_t - \frac{1}{2\pi}\int_{-\infty}^{\infty}|W(\omega_t)|^2 * |\alpha(\omega_t)|^2\, d\omega_t =$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty}|W(\omega_t)|^2 * (1-|\alpha(\omega_t)|^2) d\omega_t,$$

где $W(\omega_t)$ – спектральная функция сигнала,

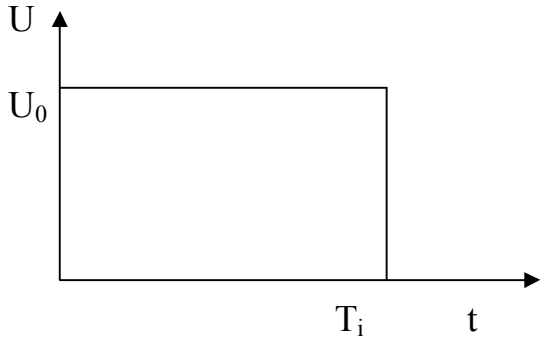
$\alpha(\omega_t)$ – передаточная характеристика канала,

$\bar{\varepsilon}^2$ - потери энергии, усредненные по всем частотам.

Сигналы и их спектральные функции

1. Прямоугольный сигнал

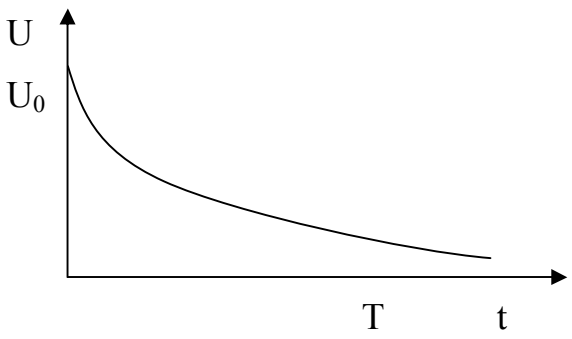
Вид сигнала



$$U(t) = \begin{cases} U_0, \text{ при } 0 < t < T_i \\ 0, \text{ для остальных } T_i \end{cases}$$

$$W(\omega_t) = U_0 T_i * \frac{\sin\frac{\omega_t T}{2}}{\frac{\omega_w T}{2}}$$

2. Экспоненциальный сигнал

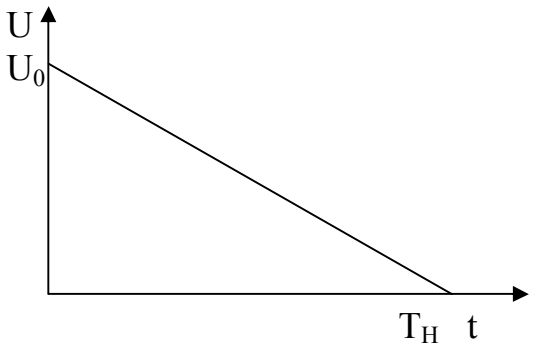


$$U(t) = U_0 e^{-pt},$$

где p - параметр, характеризующий вид экспоненты.

$$|W(\omega_t T)| = U_0 T \sqrt{\frac{(1-e^{-pt}\cos(\omega_t T))^2 + (e^{-pt}\sin(\omega_t T))^2}{(pT)^2 + (\omega_t T)^2}}$$

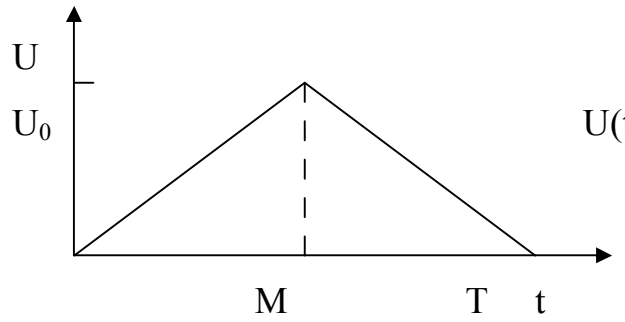
3. Линейный сигнал



$$U(t) = U_0(1-\frac{t}{T_H})$$

$$|W(\omega_t T)| = \frac{U_0}{\omega_t T}\sqrt{(\cos(\frac{\omega_t T}{2}) - \frac{\sin(\frac{\omega_t T}{2})}{\frac{\omega_t T}{2}})^2 + (\sin(\frac{\omega_t T}{2}))^2}$$

4. Треугольный сигнал

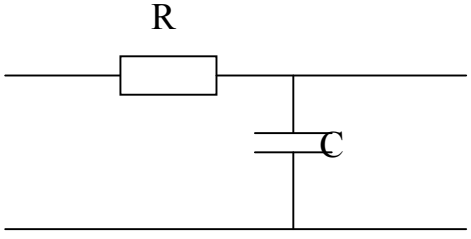


$$U(t) = \begin{cases} U_0 \frac{t}{T}, & \text{при } t < M \\ U_0(1-\frac{t-M}{T-M}), & t > M \end{cases}$$

для M =T/2

Передаточные характеристики канала

Функцию пропускания канала наиболее просто описывать интегральной RC–цепью.



В этом случае передаточная характеристика имеет вид

$$\alpha(\omega_t) = \frac{1}{1+\omega_t^2 T^2},$$

где T = RC.

Энергия сигнала на выходе канала

Энергия сигнала определяется соотношением

$$E^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} | W(\omega_t) |^2 \, d\omega_t .$$

1. Варианты индивидуальных заданий

1.1. Номер задания (вид сигнала) указан в описании.
1.2. Найти спектральную функцию для этого сигнала $W(\omega_t)$.
1.3. Выбрать передаточную функцию канала $\alpha(\omega_t)$.
1.4. Найти энергию сигнала на выходе реального и идеального каналов.
1.5. Найти количество энергии, которое теряется при несовпадении полосы частот сигнала и канала.
1.6. Построить зависимости $W(\omega_t)$, $\alpha(\omega_t)$, $\varepsilon^2(\omega_t)$.
1.7. Ответить на вопросы.
1.8. Подготовить отчет по всем пунктам задания.

2. Вопросы

2.1. Что такое спектральная функция сигнала?
2.2. Почему спектральный метод наиболее удобно использовать при анализе линейных систем?
2.3. Что означает передаточная функция канала связи?
2.4. Каким образом осуществляется частотное уплотнение каналов?
2.5. Каким образом осуществляется временное уплотнение каналов?
2.6. Чем отличается структура аналогового и цифрового каналов связи?
2.7. В чем достоинство критерия потерь количества информации от критерия отношения сигнал/шум?

7. Индивидульное задание №6. Изучение приложения FTP (3 часа)

изучение основных принципов организации протокола пересылки файлов (ftp), а также освоение основных команд ftp.

2.1. Основные принципы протокола FTP

FTP (File Transfer Protocol или "Протокол Передачи Файлов") - один из старейших протоколов в Internet и входит в его стандарты. Обмен данными в FTP проходит по TCP-каналу. Построен обмен по технологии "клиент-сервер". На рисунке 2.1 изображена модель протокола.

В FTP соединение инициируется интерпретатором протокола пользователя. Управление обменом осуществляется по каналу управления в стандарте протокола TELNET. Команды FTP генерируются интерпретатором протокола пользователя и передаются на сервер. Ответы сервера отправляются пользователю также по каналу управления.

Рис. 2.1 Диаграмма протокола FTP

Команды FTP определяют параметры канала передачи данных и самого процесса передачи. Они также определяют и характер работы с удаленной и локальной файловыми системами.

Сессия управления инициализирует канал передачи данных. При организации канала передачи данных последовательность действий другая, отличная от организации канала управления. В этом случае сервер инициирует обмен данными в соответствии с согласованными в сессии управления параметрами.

Канал данных устанавливается для того же host'a, что и канал управления, через который ведется настройка канала данных. Канал данных может быть использован как для приема, так и для передачи данных.

Рис. 2.2. Соединение с двумя разными серверами и передача данных между ними

Канал управления должен быть открыт при передаче данных между машинами. В случае его закрытия передача данных прекращается.

2.2. Режимы обмена данными

В протоколе большое внимание уделяется различным способам обмена данными между машинами различных архитектур. Действительно, чего только нет в Internet, от персоналок и Mac'ов до суперкомпьютеров. Все они имеют различную длину слова и многие различный порядок битов в слове. Кроме этого, различные файловые системы работают с разной организацией данных, которая выражается в понятии метода доступа.

2.3. Команды и процедуры

Сеанс работы с протоколом открывает команда FTP, после выполнения которой, появляется приглашение на ввод команд.

ftp>

Некоторые FTP команды могут отличаться в зависимости от типа компьютерной платформы, вы всегда можете проверить их список, набрав 'help' или '?'.

Ниже мы остановимся только на наиболее общих и полезных для практической работы, командах FTP:

ascii	Эта команда побуждает ftp преобразовывать файлы в ASCII код. (По умолчанию код всегда ASCII).
bell	Эта команда приводит к тому, что на вашем терминале появляется сигнал после завершения передачи каждого файла. Чтобы прекратить подачу сигнала, нужно снова набрать эту команду ftp.
binary	Эта команда побуждает ftp передавать файл в двоичном коде.
bye	По этой команде осуществляется выход из ftp. Эта команда закрывает все открытые

связи.

cd По этой команде имя директория на удаленной машине заменяется на новое. Вы можете записать новое имя, когда вызываете команду, как показано в примере:

```
ftp  > cd /usr/bin
```

В противном случае вы можете использовать только имя команды ftp, тогда машина запросит имя нового директория, например:

```
ftp > cd
(remote-directory) /usr/bin
```

close По этой команде закрывается текущая связь.

debug Эта команда включает и выключает многословный режим. Если режим включается, то об этом появляется сообщение на вашем дисплее, при выключении сообщений нет.

delete По этой команде удаляется файл в удаленной машине, к которой вы подключены в данный момент. Вы можете указать имя файла, который нужно удалить, при вызове ftp команды:

```
ftp > delete имя файла для удаления
```

dir Эта команда выдаст вам детальный список директория на удаленной машине, к которой вы подключены.(Сравните с опцией ls, данной ниже). Вы можете задать имя директория, который нужно распечатать, при вызове команды ftp. Например:

```
ftp > dir /usr/bin
```

Если вы не указали имя директория, то будет распечатан текущий директорий на удаленной машине. Вы можете также побудить ftp занести результаты выполнения команды в файл до того как он появится на экране. Делается это следующим образом:

```
ftp  > dir /usr/bin printfile
```

form Эта команда выводит на экран формат файла, который используется. Обычно поддерживается непечатный формат.

get Эта команда копирует файл из удаленной машины, к которой вы в данный момент подсоединены. Этот файл копируется в вашу машину. Например:

```
ftp > get имя файла удаленной машины имя файла вашей машины
```

Если вы просто укажете имя файла удаленной машины, который нужно скопировать, то файл на вашей машине будет иметь тоже самое имя. Пример:

```
ftp > get имя удаленной машины
```

help Эта команда выдает на экран информацию о работе ftp . Если после help задать имя команды, то появится информация об этой команде. Если набрать просто help, то появится информация обо всех командах ftp.

lcd Эта команда изменяет рабочий директорий, используемыйn ftp, на вашей машине. Вы можете задать имя директория, который вам нужен как рабочий, например:

```
ftp > lcd /ysr/deb
```

Если вы не задали имя директория, то будет использоваться ваш домашний директорий.

ls Эта команда распечатывает абревиатурный список содержания директория удаленной машины, с которой вы связаны. Вы связаны в данный момент. Вы можете задать имя директория, который вы хотите распечатать. Например:

```
ftp > ls /usr/bin
```

Если вы не задали имя, будет распечатан текущий директорий. Можно задать, чтобы результаты выполнения команды были помещены в файл до появления их на дисплее.

Это делается ftp, если указать имя файла на вашей машине куда следует поместить листинг директория, например:

ftp > ls /usr/bin printfile

mdelete Эта команда удаляет список файлов на удаленной машине, с которой вы связаны в данный момент. Например:

ftp > mdelete имя 1файла удаленной машины имя 2файла...

mdir По этой команде выдается список листинг директория удаленной машины и результат помещается в файл вашей машины. Вы можете задать список файлов удаленной машины и имя файла вашей машины, куда поместить результат при вызове команды. Например:

ftp > mdir имя 1файла удаленной машины... printfile

mget Эта команда копирования одного или более файлов с удаленной машины, к которой вы подключены в данный момент на вашу машину. Файлы после копирования будут иметь те же имена. Вы можете указать список файлов для копирования:

ftp > mget имя 1файла удаленной машины имя 2файла...

mkdir Эта команда создает директорий на удаленной машине, к которой вы в данный момент подключаетесь. например:

ftp > mkdir /u/mydir

mls Эта команда получает абревиатурный список группы файлов текущего директория на удаленной машине и помещает результат в файл на вашей машине. Вы можете задать список файлов удаленной машины и файл вашей машины, куда поместить результат при вызове команды, например:

ftp > mls имя 1файла удаленной машины ...printfile

mput Эта команда копирует один или более файлов с вашей машины в удаленную машину, с которой вы связаны в данный момент. На удаленной машине файлы будут иметь теже имена. Вы можете задать список фалов при вызове команды,например:

ftp > mput 1файл вашей машины 2файл вашей машины...

open Эта команда устанавливает связь с удаленной машиной, которая предполагается для передачи файлов. При вызове команды вы можете указать имя машины, например:

ftp > open admin

Если имя не указано, программа запросит его:

ftp > open

(to) имя машины

prompt Эта команда предотвращает ваш запрос к ftp о разрешении на переход между файлами в многофайловых командах, таких как mget. Эта команда подключается и отключается при повторном наборе.

put Эта команда перемещает файл из вашей машины в удаленную машину, к которой вы в данный момент подключены.

ftp > put имя вашего файла имя файла удаленной машины

или

ftp > put имя вашего файла

pwd Эта команда вынуждает ftp печатать имя текущего рабочего директория на удаленной машине, с которой вы связаны в данный момент.

quit Команда аналогична команде bye, о которой говорилось выше.

quote Команда заставляет ftp посылать параметры, которые вы вводите в машину, посылать к удаленной машине для выполнения. Параметры это ftp команды и другие параметры. Те команды, что ftp поддерживает, могут быть отображены на экране с помощью команды remotehelp. Вы можете ввести эту команду при вызове программы ftp , например:

ftp > quote NLST

recv Эта команда аналогична команде get, описанной выше.

remotehelp Эта команда запрашивает помощь ftp на удаленной машине, с которой вы связаны в данный момент. Эта информация сообщает о том какие команды поддерживает удаленная машина.

rename Эта команда переименовывает файл на удаленной машине, с которой вы связаны в текущий момент.

ftp > rename имя 1файла имя 2файла

mdir Эта команда удаляет директорий на удаленной машине, с которой вы связаны в данный момент. Вы можете задать имя директория, который следует удалить, при вызове команды, например:

ftp > rmdir /u/mydir

или вы можете не задавать имя при вызове команды, и машина запросит вас о нем:

ftp > rmdir

(directory-name) /u/mydir

Эта команда не всегда поддерживается.

send Эта команда аналогична команде put, описанной выше.

sendport Эта команда заставляет ftp запрещать возможность задания порта локальной машины для данных удаленной машины. Эта команда может подключаться и отключаться ее повторным набором. При вызове ftp по умолчанию задается определенный порт. Эту команду следует использовать по совету вашего системного администратора.

status Это команда заставляет ftp изображать свой текущий статус на вашем терминале. Статус включает режимы, которые выбраны командами bell,form,hash,glob,port,type.

type Эта команда устанавливает в каком виде передается файл. Допустимы коды ASCII и двоичный. Эта команда аналогична командам ascii и binary. Если вы не указали тип при вызове команды, то устанавливается ASCII.

trace Эта команда заставляет ftp разрешать пакетную трассировку. Эта команда включается и отключается ее повторным набором. Эту команду следует использовать только по совету вашего системного администратора.

user Эта команда позволяет вам идентифицировать самого себя на удаленной машине при установлении связи.

ftp > user mike cat myaccount

verbose Эта команда заставляет ftp запрещать многословный режим. Эта команда включается и выключается при повторном наборе. В многословном режиме ftp протокольные сообщения, посланные удаленной машиной появляются на вашем терминале. Кроме того,

в этом режиме отображается статистика после передачи каждого файла. Если этот режим запрещен, то данная информация не изображается.

? Другое название команды help.

2.4. FTP отклики

Отклики состоят из 3-циферных значений в формате ASCII, и необязательных сообщений, которые следуют за числами. Подобное представление откликов объясняется тем, что программному обеспечению необходимо посмотреть только цифровые значения, чтобы понять, что ответил процесс, а дополнительную строку может прочитать человек. Поэтому пользователю достаточно просто прочитать сообщение (причем нет необходимости запоминать все цифровые коды откликов).
Каждая из трех цифр в коде отклика имеет собственный смысл. На рисунке 2.4 показаны значения первых и вторых цифр в коде отклика.

Отклик	Описание
1yz	Положительный предварительный отклик. Действие началось, однако необходимо дождаться еще одного отклика перед отправкой следующей команды.
2yz	Положительный отклик о завершении. Может быть отправлена новая команда.
3yz	Положительный промежуточный отклик. Команда принята, однако необходимо отправить еще одну команду.
4yz	Временный отрицательный отклик о завершении. Требуемое действие не произошло, однако ошибка временная, поэтому команду необходимо повторить позже.
5yz	Постоянный отрицательный отклик о завершении. Команда не была воспринята и повторять ее не стоит.
x0z	Синтаксическая ошибка.
x1z	Информация.
x2z	Соединения. Отклики имеют отношение либо к управляющему, либо к соединению данных.
x3z	Аутентификация и бюджет. Отклик имеет отношение к логированию или командам, связанным с бюджетом.
x4z	Не определено.
x5z	Состояние файловой системы.

Рисунок 2.4. Значения первой и второй цифр в 3-циферном коде отклика.

Третья цифра дает дополнительное объяснение сообщению об ошибке. Ниже приведены некоторые типичные отклики с возможными объясняющими строками.

125 Соединение данных уже открыто; начало передачи.
200 Команда исполнена.
214 Сообщение о помощи (для пользователя).
331 Имя пользователя принято, требуется пароль.

425 Невозможно открыть соединение данных.
452 Ошибка записи файла.
500 Синтаксическая ошибка (неизвестная команда).
501 Синтаксическая ошибка (неверные аргументы).
502 Нереализованный тип MODE.

Обычно каждая FTP команда генерируют отклик в одну строку. Например, команда QUIT сгенерирует следующий отклик:

3. Задание

1. Изучить теоретический материал.
2. Получить вариант задания.
3. Выполнить задание.
4. Ответить на контрольные вопросы.

4. Пример выполнения индивидульного задания

Задание на практическое занятие

Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: put, get, delete.

Реализация

В качестве среды разработки воспользуемся средой Delphi 5, для выполнения команд ftp будем использовать функцию winexec. Сами команды будем помещать в текстовом файле.

5. Результаты

В результате проделанной работы должна быть написана программа, демонстрирующая предложенные функции ftp. Для примера приведен листинг основных процедур.

Соединение с ftp

```
procedure connectftp;
 var f:textfile;
     s:string;
begin
 create_file('lsfile');
 create_file('dirfile');
 AssignFile(f,lokpath+'start.ftp');
 Rewrite(f);
 writeln(f, 'open '+host);
 writeln(f, 'user '+user);
 writeln(f, pass);
 s:=copy(lokpath,1,length(lokpath)-1);
 if s[length(s)]=':' then  s:=s+'\';
 writeln(f, 'lcd '+s);
 writeln(f, 'ls -p  lsfile');
 writeln(f, 'dir -p dirfile ');
 writeln(f, 'quit');
```

```
 CloseFile(f);
 s:='ftp -n -s:'+lokpath+'start.ftp';
 winexec(Pchar(s),cons);
end;
```

Команда put

```
procedure importftp;
 var s:string;
     f:textfile;
begin
 create_file('lsfile');
 create_file('dirfile');
 AssignFile(f,lokpath+'start.ftp');
 Rewrite(f);
 writeln(f, 'open '+host);
 writeln(f, 'user '+user);
 writeln(f, pass);
 s:=directorylistbox1.GetItemPath(directorylistbox1.ItemIndex);
 writeln(f, 'lcd '+s);
 if length(path)>1 then writeln(f, 'cd'+copy(path,1,length(path)-1));
 writeln(f, 'put '
+Filelistbox1.items.strings[Filelistbox1.itemindex]);
 s:=copy(lokpath,1,length(lokpath)-1);
 if s[length(s)]=':' then s:=s+'\';
 writeln(f, 'lcd '+s);
 writeln(f, 'ls -p lsfile');
 writeln(f, 'dir -p dirfile ');
 writeln(f, 'quit');
 CloseFile(F);
 s:='ftp -n -s:'+lokpath+'start.ftp';
 winexec(Pchar(s),cons);
end;
```

Команда get

```
procedure TForm1.exportftp;
 var s:string;
     f:textfile;
begin
 AssignFile(f,lokpath+'start.ftp');
 Rewrite(f);
 writeln(f, 'open '+host);
 writeln(f, 'user '+user);
 writeln(f, pass);
 s:=directorylistbox1.GetItemPath(directorylistbox1.ItemIndex);
 writeln(f, 'lcd '+s);
 s:=copy(path,1,length(path)-1);
 if length(path)>1 then writeln(f, 'cd '+s);
 writeln(f, 'get '+listbox1.Items.Strings[listbox1.itemindex]);
 s:=copy(lokpath,1,length(lokpath)-1);
 if s[length(s)]=':' then s:=s+'\';
 writeln(f, 'lcd '+s);
 writeln(f, 'quit');
 CloseFile(F);
 s:='ftp -n -s:'+lokpath+'start.ftp';
 winexec(Pchar(s),cons);
end;
```

Команда delete

```
procedure TForm1.deleteftp;
 var s:string;
     f:textfile;
begin
 create_file('lsfile');
 create_file('dirfile');
 AssignFile(f,lokpath+'start.ftp');
 Rewrite(f);
 writeln(f, 'open '+host);
 writeln(f, 'user '+user);
 writeln(f, pass);
 s:=copy(lokpath,1,length(lokpath)-1);
 if s[length(s)]=':' then s:=s+'\';
 writeln(f, 'lcd '+s);
 s:=copy(path,1,length(path)-1);
 if length(path)>1 then writeln(f, 'cd '+s);
 writeln(f,'delete'+listbox1.Items.Strings[listbox1.itemindex]);
 writeln(f, 'ls -p lsfile');
 writeln(f, 'dir -p dirfile ');
 writeln(f, 'quit');
 CloseFile(f);
 s:='ftp -n -s:'+lokpath+'start.ftp';
 winexec(Pchar(s),cons);
end;
```

6. Содержание отчета

В результате выполнения индивидуального задания должен быть написан отчет, содержащий следующие пункты:

- Цель
- Полученные результаты
- Выводы
- Ответы на вопросы

7. Вопросы

1. Назовите виды режимов обмена данными ftp.
2. Назовите каналы, которые ftp использует в своей работе.
3. По какому каналу происходит обмен данными в FTP?
4. Что такое “отклики ftp”, для чего они предназначены?

8. Варианты индивидуальных заданий

1. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: put, get, delete.

2. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: rename, mls, mget.

3. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: put, bin, ascii.

4. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: prompt, mget, delete.

5. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: mkdir, mdir, send.

6. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: ls, lcd, cd.

7. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: pwd, rename, mdir.

8. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: status, bell, form.

9. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: ls, put, mdelete.

10. Продемонстрировать работу протокола ftp и знание основных команд ftp, путем написания программы на любом языке программирования, позволяющей открыть сеанс работы с ftp и выполнить следующие команды ftp: dir, debag, ascii.

8. Список рекомендуемой литературы

1. Замятин Н.В. Организация ЭВМ и систем. Томск. 2005 г. 198 с .
2. Замятин Н.В. Организация ЭВМ и систем. Учебно-методическое пособие, Томск. 2005 г. 200 с .

3. Фролов А.В., Фролов Г.В. аппаратное обеспечение IBM PC. Мифи-диалог, 1992 г.

4.Лю-ю жен, Гибсон. Микропроцессоры семейства 8086/88. М.Мир.1998 г.

5. Коган Б.М. Электронные вычислительные машины и системы. М. Энергоиздат. 1995 г.

2. Ровдо **А. А.** Микропроцессоры от 8086 до Pentium III Xeon и AMD-K6-3. - М.:, 2000. - 592 с.

3. Фролов А.В., Фролов Г.В. аппаратное обеспечение IBM PC. Мифи-диалог, 1992 г.