МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра телекоммуникационных систем

УТВЕРЖДАЮ

Проректор по учебной работе

_____ Н.Г. Зарипов

«_21_» ____09_____ 2015 г.

РАБОЧАЯ ПРОГРАММА

УЧЕБНОЙ ДИСЦИПЛИНЫ

«*МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА В ИНФОКОММУНИКАЦИОННЫХ СИСТЕМАХ*»

Уровень подготовки: высшее образование – подготовка кадров высшей квалификации

Направление подготовки научно-педагогических кадров высшей квалификации (аспирантура)
11.06.01 Электроника, радиотехника и системы связи
(код и наименование направления подготовки)

Направленность подготовки
Системы, сети и устройства телекоммуникаций
(наименование программы подготовки)

Квалификация (степень) выпускника
Исследователь. Преподаватель-исследователь

Форма обучения
очная

Уфа 2015

Содержание

1. Место дисциплины в структуре образовательной программы

Дисциплина *микропроцессорные устройства в инфокоммуникационных системах* является дисциплиной *вариативной* части.
Рабочая программа составлена в соответствии с требованиями Федерального государственного образовательного стандарта высшего образования по направлению подготовки научно-педагогических кадров высшей квалификации (аспирантура) 11.06.01 Электроника, радитехника и системы связи, утвержденного приказом Министерства образования и науки Российской Федерации от "30" июля 2014 г. № 876 и приказа Министерства образования и науки Российской Федерации от 30.04.2015 N 464 "О внесении изменений в федеральные государственные образовательные стандарты высшего образования (уровень подготовки кадров высшей квалификации)". Является неотъемлемой частью основной образовательной профессиональной программы (ОПОП).

Целью освоения дисциплины является формирование профессиональной компетенции для понимания основ и принципов построения современной микропроцессорной техники и готовности к использованию процессоров в инфокоммуникационных системах, включая знания, умения и навыки, обеспечивающие успешность профессиональной деятельности.

В **задачи** дисциплины «Микропроцессорные устройства в инфокоммуникационных системах» для аспирантов входит

1) освоение аспирантами знаний:

- основ архитектуры микропроцессоров;
- основных алгоритмы цифровой обработки сигналов, применяемых в инфокоммуникационных системах, и методов их реализации;

2) формирование у аспирантов умений и навыков:

- разработки программного обеспечения микропроцессоров;
- использования программных пакетов для поддержки процесса разработки программного обеспечения для микропроцессорных устройств;
- реализации алгоритмов цифровой обработки сигналов с использованием микропроцессоров.

Входные компетенции не предусмотрены.

.

Исходящие компетенции

Компетенция	Код	Уровень освоения, определяемый этапом формирования компетенции	Название дисциплины (модуля), практики, научных исследований, сформировавших данную компетенцию
Способность к пониманию основ и принципов построения современной микропроцессорной техники и готовность к использованию процессоров в инфокоммуникационных системах	ПК-4	повышенный уровень освоения компетенции в рамках научных исследований	Научные исследования
	ПК-4	повышенный уровень освоения компетенции в рамках ГИА	ГИА

2. Перечень результатов обучения

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций.

Планируемые результаты обучения по дисциплине

№	Формируемые компетенции	Код	Знать	Уметь	Владеть
1	Способность к пониманию основ и принципов построения современной микропроцессорной техники и готовность к использованию процессоров в инфокоммуникационных системах	ПК-4	Основы архитектуры микропроцессоров; основные алгоритмы цифровой обработки сигналов, применяемых в инфокоммуникационных системах, и методы их реализации	Разрабатывать и отлаживать программное обеспечение для микропроцессоров; пользоваться программными пакетами для поддержки процесса разработки программного обеспечения для микропроцессорных устройств; реализовывать алгоритмы цифровой обработки сигналов с использованием микропроцессоров	Навыками разработки и отладки программного обеспечения для микропроцессоров; навыками использования программных пакетов для поддержки процесса разработки программного обеспечения для микропроцессорных устройств; навыками реализации алгоритмов цифровой обработки сигналов с использованием микропроцессоров

3. Содержание и структура дисциплины (модуля)

Общая трудоемкость дисциплины составляет 7 зачетных единиц (252 часа).

Трудоемкость дисциплины по видам работ

Вид работы	Трудоемкость, час.	
	3 семестр (3 ЗЕ/108)	4 семестр (4 ЗЕ/144)
Лекции (Л)	6 ч.	4 ч.
Практические занятия (ПЗ)	8 ч.	6 ч.
Лабораторные работы (ЛР)	-	-
КСР	-	-
Курсовая проект работа (КР)	-	-
Расчетно - графическая работа (РГР)	-	-
Самостоятельная работа (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к лабораторным и практическим занятиям, коллоквиумам, рубежному контролю и т.д.)	85 ч.	98 ч.
Подготовка и сдача экзамена	-	36 ч.
Подготовка и сдача зачета	9 ч.	-
Вид итогового контроля (зачет, экзамен)	зачет с оценкой	экзамен

Содержание разделов и формы текущего контроля

№	Наименование и содержание раздела	Количество часов				Литература, рекомендуемая студентам*	Виды интерактивных образовательных технологий**
		Аудиторная работа		СРС	Всего		
		Л	ПЗ				
1.	*Архитектура микропроцессоров* Архитектура микропроцессоров, особенности архитектуры цифровых сигнальных процессоров (ЦСП). Конвейерное выполнение команд. Классификация ЦСП. ЦСП ведущих производителей.	2	-	55	57	Р 6.1 №1	Лекция-визуализация – передача информации посредством схем, таблиц, рисунков
2.	*Цифровая обработка сигналов* Структура системы цифровой обработки сигналов. Z-преобразование. Разностные уравнения и передаточные функции для описания цифровых фильтров. Эффекты квантования в цифровых фильтрах. Свертка дискретных сигналов. Дискретное преобразование Фурье, быстрое преобразование Фурье.	2	2	60	64	Р 6.1 №2	Традиционное занятие с системным изложением материала и проверкой знаний на семинаре
3.	*Программирование на языке ассемблера* Регистры. Команды пересылки данных, режимы адресации. Арифметические команды. Логические команды. Команды передачи управления. Система команд сопроцессора. Технология MMX. Реализация вычисления свертки дискретных сигналов. Реализация вычисления быстрого преобразования Фурье.	6	12	68	86	Р 6.1 №3, 1	Традиционное занятие с системным изложением материала и проверкой знаний на семинаре

Практические занятия (семинары)

№ занятия	№ раздела	Тема	Кол-во часов
1, 2	3	Команды пересылки данных, режимы адресации	4
3	3	Арифметические команды	2
4	3	Логические команды и команды сдвига	2
5	3	Команды передачи управления	2
6	2, 3	Свертка дискретных сигналов, реализация вычисления свертки на ассемблере	2
7	2, 3	Быстрое преобразование Фурье (БПФ), реализация вычисления БПФ на ассемблере	2

4. Учебно-методическое обеспечение самостоятельной работы студентов

Вопросы для самостоятельного изучения (для подготовки к обсуждению на семинаре).

Тема 1. «Архитектура микропроцессоров, особенности архитектуры цифровых сигнальных процессоров».

1. Семейство сигнальных процессоров ADSP Blackfin фирмы Analog Devices.
2. Процессоры ADSP SHARC фирмы Analog Devices.
3. Процессор DSP 56300 фирмы Freescale. Общие сведение, структурная схема, характеристики.
4. Процессор DSP 56300 фирмы Freescale. Ядро процессора.
5. Процессоры семейства C2000 фирмы Texas Instruments.
6. Процессоры семейства C5000 фирмы Texas Instruments.
7. Процессоры семейства C6000 фирмы Texas Instruments.

Тема 2. «Цифровая обработка сигналов»**.**

1. Примеры вычисления z-преобразования.
2. Связь z-преобразования с преобразованиями Лапласа и Фурье.
3. Свойства z-преобразования.
4. Обратное z-преобразование.
5. Эффекты квантования в цифровых фильтрах. Шум квантования.
6. Эффекты квантования в цифровых фильтрах. Квантование коэффициентов цифровых фильтров.
7. Эффекты квантования в цифровых фильтрах. Масштабирование коэффициентов цифроых фильтров.
8. Эффекты квантования в цифровых фильтрах. Предельные циклы.

Тема 3. «Программирование на языке ассемблера»**.**

1. Система команд сопроцессора. Команды передачи данных, команды загрузки констант.
2. Система команд сопроцессора. Команды сравнения данных.
3. Система команд сопроцессора. Арифметические команды.
4. Система команд сопроцессора. Команды трансцендентных функций.
5. Система команд сопроцессора. Команды управления сопроцессором.
6. Технология MMX. Команды сравнения, арифметические команды.
7. Технология MMX. Команды упаковки, команды сдвигов, логические команды.

8. Технология MMX. Логические команды.
9. Технология MMX. Команды пересылки данных.

В ходе самостоятельной работы аспиранты:

- выполняют задания по подготовке к практическим занятиям;
- осуществляют поиск информации в библиотечно-информационной системе вуза, сети Интернет).

При выполнении самостоятельной работы по внеаудиторному чтению аспиранты пользуются литературой, рекомендуемой их научными руководителями.

Формы контроля самостоятельной работы:

- проверка письменных заданий на практических занятиях;
- индивидуальные консультации с преподавателем.

5. Фонд оценочных средств

Оценка уровня освоения дисциплины осуществляется в виде текущего и промежуточного контроля успеваемости аспирантов университета на основе критериев оценки уровня освоения дисциплины.

Контроль представляет собой набор заданий и проводится в форме контрольных мероприятий по оцениванию фактических результатов обучения студентов и осуществляется ведущим преподавателем.

Объектами оценивания выступают:

– учебная дисциплина (активность на занятиях, своевременность выполнения различных видов заданий, посещаемость всех видов занятий по аттестуемой дисциплине и пр.);

– степень усвоения теоретических знаний;

– уровень овладения практическими умениями и навыками по всем видам учебной работы;

– результаты самостоятельной работы.

Активность обучающегося на занятиях оценивается на основе выполненных работ и заданий, предусмотренных ФОС дисциплины.

Оценивание проводится преподавателем независимо от наличия или отсутствия обучающегося (по уважительной или неуважительной причине) на занятии. Оценка носит комплексный характер и учитывает достижения обучающегося по основным компонентам учебного процесса за текущий период.

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Уровень освоения, определяемый этапом формирования компетенции	Наименование оценочного средства*
1	Архитектура цифровых сигнальных процессоров	ПК-4	базовый	Ответы на вопросы (Т)
2	Цифровая обработка сигналов	ПК-4	базовый	Т
3	Программирование на языке ассемблера	ПК-4	базовый	Т

Типовые оценочные материалы

1. Комплект заданий для практических работ 1, 2

1. Приведите последовательность команд, пересылающих содержимое нулевого элемента массива (с 8-битными элементами) во второй элемент массива с использованием косвенной базовой адресации.
2. Приведите последовательность команд, пересылающих содержимое первого элемента массива (с 16-битными элементами) в нулевой элемент массива с использованием косвенной индексной адресации со смещением.
3. Приведите последовательность команд, пересылающих содержимое первого элемента массива (с 16-битными элементами) в нулевой элемент массива с использованием косвенной базовой индексной адресации.
4. Приведите последовательность команд mov, осуществляющих ту же операцию, что и команда xchg ax, bx. В качестве промежуточного можно использовать любой регистр общего назначения.
5. Приведите последовательность команд, загружающих 0 в сегментный регистр eds.
6. Приведите последовательность команд работы со стеком, копирующих содержимое регистра eflags в регистр eax.

2. Комплект заданий для практической работы 3

1. Приведите последовательность команд, вычисляющую в регистре ax результат сложения двух 8-битных чисел, записанных в ячейках памяти. Используйте команды сложения с 8-битными операндами.
2. Приведите последовательность команд, вычисляющую в регистре ax результат сложения трех 8-битных чисел, записанных в ячейках памяти. Используйте команды сложения с 8-битными операндами.
3. Приведите последовательность команд, вычисляющую в регистре ax результат вычитания ax=bx-cx. Используйте команды вычитания с 8-битными операндами.
4. Приведите два варианта последовательностей команд, прибавляющих к 32-битному содержимому ячейки памяти единицу. Один вариант с командой add, второй – с командой inc.
5. Приведите два варианта последовательностей команд, вычитающих из 16-битного содержимому ячейки памяти единицу. Один вариант с командой sub, второй – с командой dec.
6. Приведите последовательность команд, вычисляющую в регистре eax результат знакового умножения eax=ebx*-3.
7. Приведите последовательность команд, вычисляющую результат беззнакового деления ax/bl.
8. Приведите последовательность команд, вычисляющую результат знакового деления 32-битного числа на 16-битное.

3. Комплект заданий для практической работы 4

1. Приведите последовательность команд для переключения в единичное состояние битов 1, 2, 15 ячейки памяти (16-разрядной)?
2. Какой командой можно проверить состояние бита 14 регистра bx?
3. Какой командой сдвига можно поделить знаковое число в регистре ax на 2?
4. Приведите последовательность команд для умножения знакового числа в регистре al на 8 с размещением результата в регистре ax?
5. Приведите последовательность команд для сохранения содержимого 16 старших разрядов регистра ecx в ячейке памяти. Результирующее значение ecx после завершения работы должно совпадать с исходным.
6. Приведите последовательность команд для деления на четыре 96-битного знакового числа содержащегося в трех регистрах ecx:ebx:eax.
7. Приведите последовательность команд для умножения на шестнадцать 96-битного беззнакового числа содержащегося в трех регистрах ecx:ebx:eax.

4. Комплект заданий для практической работы 5

1. Приведите последовательность команд, сравнивающую два беззнаковых числа в регистре ax и ячейке памяти (ax и содержимое ячейки не должны измениться). Если первое число меньше второго, то перейти на ветку с инкрементом регистра cx, иначе декрементировать cx.

2. Приведите последовательность команд, вычитающую 10 из содержимого регистра ax до тех пор, пока ax не станет отрицательным.

3. Приведите последовательность команд, вычисляющую сумму eax=eax+edi и при наличии переноса инкрементирующую регистр edx.

4. Приведите последовательность команд с использованием команды организации цикла, выполняющую 200 раз вычитание числа 13 из содержимого регистра ax. Предусмотреть досрочное завершение цикла, если результат вычитания будет равен нулю.

5. Приведите последовательность команд, вычисляющих сумму четырех 8-битных ячеек памяти.

5. Комплект заданий для практической работы 6

1. Составьте программу вычисляющую линейную свертку последовательностей $x_1(n)=\{1,2,3\}$; $x_2(n)=\{4,5,6\}$ (результат свертки рассчитан в примере 2 и равен $y(n)=\{4,13,28,27,18\}$). Исходные данные и результат считать 8-разрядными. Программа должна работать правильно без дополнения исходных последовательностей нулями.

2. Составьте программу, вычисляющую круговую свертку последовательностей $x_1(n)=\{1,2,3\}$; $x_2(n)=\{4,5,6\}$ (результат свертки рассчитан в примере 1 и равен $y(n)=\{31; 31; 28\}$). Исходные данные и результат считать 8-разрядными. Для упрощения программы дополните исходную последовательность x_2 еще одним периодом.

3. Измените программу из п.5 так, чтобы она работала правильно без каких-либо модификаций или дополнений исходных последовательностей.

4. Приведите последовательность команд, загружающую 8 байт из ячейки памяти в mm0, переставляющую 16-битные слова mm0 в обратном порядке и сохраняющую mm0 в исходной ячейке.

5. Вычислите круговую свертку сигналов $x_1=\{2; -3; 1; -4\}$ и $x_2=\{3; -4; 2; -8\}$.

6. Составьте программу, вычисляющую круговую свертку последовательностей длиной по 4 элемента с использованием команд MMX. Исходные данные и результат считать 16-разрядными.

В качестве исходных данных можно использовать последовательности из задачи 5.

6. Комплект заданий для практической работы 7

1. Вычислить БПФ последовательности $x(n)=\{0; 1; 2; 3\}$ с прореживанием по частоте с основанием 2.

2. Вычислить БПФ последовательности $x(n)=\{0; 1; 2; 3\}$ с прореживанием по времени с основанием 2.

3. Вычислить обратное БПФ результата задачи 2.

4. Составьте процедуру для вычитания комплексных чисел. Исходные данные в регистрах eax и ebx, результат в регистре eax (младшие 16 разрядов – вещественная часть, старшие 16 разрядов – комплексная часть).

5. Напишите программу реализующую вычисление БПФ последовательности длиной 4 с прореживанием по времени с основанием 2. Разрядность исходных данных – 16 бит. Можно использовать процедуры для выполнения арифметических операций с комплексными числами. Можно использовать процедуры для выполнения арифметических операций с комплексными числами из задач 1-3 и процедуры swapfft из задачи 4.

В качестве исходных данных можно использовать данные: $x(n)=\{0; 1; 2; 3\}$, $X(k)=\{6; -2+j2; -2; -2-j2\}$.

6. Напишите программу реализующую вычисление ОДПФ с использованием БПФ последовательности длиной 4. Разрядность исходных данных – 16 бит. Можно использовать

процедуру для вычисления БПФ из задачи 4 или 5. Допустимо считать, что комплексная часть всех элементов результата ОДПФ равна нулю.

В качестве исходных данных можно использовать данные из примера 5: $X(k)=\{6; -2+j2; -2; -2-j2\}$, $x(n)=\{0; 1; 2; 3\}$.

Вопросы к зачету с оценкой

1. Архитектура микропроцессоров, особенности архитектуры цифровых сигнальных процессоров (ЦСП).
2. Конвейерное выполнение команд.
3. Классификация ЦСП.
4. Семейство сигнальных процессоров ADSP Blackfin фирмы Analog Devices.
5. Процессоры ADSP SHARC фирмы Analog Devices.
6. Процессор DSP 56300 фирмы Freescale. Общие сведение, структурная схема, характеристики.
7. Процессор DSP 56300 фирмы Freescale. Ядро процессора.
8. Процессоры семейства C2000 фирмы Texas Instruments.
9. Процессоры семейства C5000 фирмы Texas Instruments.
10. Процессоры семейства C6000 фирмы Texas Instruments.
11. Структура системы цифровой обработки сигналов.
12. Разностные уравнения и передаточные функции для описания цифровых фильтров.
13. Примеры вычисления z-преобразования.
14. Связь z-преобразования с преобразованиями Лапласа и Фурье.
15. Свойства z-преобразования.
16. Обратное z-преобразование.
17. Регистры процессора.
18. Команды пересылки данных, режимы адресации.
19. Арифметические команды.
20. Логические команды.

Критерии оценки:

- оценка «отлично» выставляется обучающемуся, если два вопроса имеют полные ответы[1] и решено более 75% задач на практических занятиях. Содержание ответов свидетельствуют об уверенных знаниях обучающегося и о его умении решать профессиональные задачи, соответствующие его будущей квалификации.
- оценка «хорошо» выставляется обучающемуся, если минимум один вопрос имеет полный ответ и решено от 50 до 75% задач на практических занятиях. Содержание ответов свидетельствует о достаточных знаниях обучающегося и о его умении решать профессиональные задачи, соответствующие его будущей квалификации.
- оценка «удовлетворительно» выставляется обучающемуся, если два вопроса имеют неполные ответы и решено от 25 до 50% задач на практических занятиях. Содержание ответов свидетельствует о недостаточных знаниях обучающегося и о его ограниченном умении решать профессиональные задачи.
- оценка «неудовлетворительно» выставляется обучающемуся, если два вопроса не имеют ответа и решено менее 25% задач на практических занятиях. Содержание ответов свидетельствует о слабых знаниях обучающегося и о его неумении решать профессиональные задачи.

Вопросы к экзамену

1. Свертка дискретных сигналов.
2. Дискретное преобразование Фурье, быстрое преобразование Фурье.
3. Эффекты квантования в цифровых фильтрах. Шум квантования.

[1] Полный ответ – развёрнутый ответ, охватывающий все технические аспекты вопроса; неполный ответ – ход рассуждения правильный, но конечный результат (вывод) неверный; нет ответа – отсутствует ответ или ход рассуждения выбран неправильно, наличие грубых ошибок.

4. Эффекты квантования в цифровых фильтрах. Квантование коэффициентов цифровых фильтров.

5. Эффекты квантования в цифровых фильтрах. Масштабирование коэффициентов цифроых фильтров.

6. Эффекты квантования в цифровых фильтрах. Предельные циклы.

7. Команды передачи управления.

8. Реализация вычисления свертки дискретных сигналов на ассемблере.

9. Реализация вычисления быстрого преобразования Фурье на ассемблере.

10. Система команд сопроцессора. Команды передачи данных, команды загрузки констант.

11. Система команд сопроцессора. Команды сравнения данных.

12. Система команд сопроцессора. Арифметические команды.

13. Система команд сопроцессора. Команды трансцендентных функций.

14. Система команд сопроцессора. Команды управления сопроцессором.

15. Технология MMX. Команды сравнения, арифметические команды.

16. Технология MMX. Команды упаковки, команды сдвигов.

17. Технология MMX. Логические команды.

18. Технология MMX. Команды пересылки данных.

Критерии оценки:

- оценка «отлично» выставляется обучающемуся, если два вопроса имеют полные ответы[2] и решено более 75% задач на практических занятиях. Содержание ответов свидетельствуют об уверенных знаниях обучающегося и о его умении решать профессиональные задачи, соответствующие его будущей квалификации.
- оценка «хорошо» выставляется обучающемуся, если минимум один вопрос имеет полный ответ и решено от 50 до 75% задач на практических занятиях. Содержание ответов свидетельствует о достаточных знаниях обучающегося и о его умении решать профессиональные задачи, соответствующие его будущей квалификации.
- оценка «удовлетворительно» выставляется обучающемуся, если два вопроса имеют неполные ответы и решено от 25 до 50% задач на практических занятиях. Содержание ответов свидетельствует о недостаточных знаниях обучающегося и о его ограниченном умении решать профессиональные задачи.
- оценка «неудовлетворительно» выставляется обучающемуся, если два вопроса не имеют ответа и решено менее 25% задач на практических занятиях. Содержание ответов свидетельствует о слабых знаниях обучающегося и о его неумении решать профессиональные задачи.

6. Учебно-методическое и информационное обеспечение дисциплины (модуля)

6.1 Основная литература

1. Сперанский В.С. Сигнальные микропроцессоры и их применение в системах телекоммуникаций и электроники: учебное пособие. – М.: Горячая линия-Телеком, 2008.

2. Сергиенко А.Б. Цифровая обработка сигналов: учебное пособие. – СПб.: БХВ-Петербург, 2011.

3. Юров В.И. Assembler: [учебник для вузов] — Санкт-Петербург [и др.] : Питер, 2010

6.2 Дополнительная литература

1. Журналы и монографии по специальности

[2] Полный ответ – развёрнутый ответ, охватывающий все технические аспекты вопроса; неполный ответ – ход рассуждения правильный, но конечный результат (вывод) неверный; нет ответа – отсутствует ответ или ход рассуждения выбран неправильно, наличие грубых ошибок.

6.3. Интернет-ресурсы (электронные учебно-методические издания, лицензионное программное обеспечение)

Каждый обучающийся (аспирант) в течение всего периода обучения обеспечен индивидуальным неограниченным доступом к следующим электронно-библиотечным системам (ЭБС «Лань» (http://e.lanbook.com/, ЭБС Ассоциации «Электронное образование Республики Башкортостан» http://e-library.ufa-rb.ru, Консорциум аэрокосмических вузов России http://elsau.ru/, Электронная коллекция образовательных ресурсов УГАТУ http://www.library.ugatu.ac.ru/cgi-bin/zgate.exe?Init+ugatu-fulltxt.xml,simple-fulltxt.xsl+rus), содержащим все издания основной литературы, перечисленные в рабочих программах дисциплин (модулей), практик, НИ сформированным на основании прямых договорных отношений с правообладателями.

Электронно-библиотечная система и электронная информационно-образовательная среда обеспечивают возможность индивидуального доступа для каждого обучающегося из любой точки, в которой имеется доступ к сети Интернет, как на территории университета, так и вне ее.

Библиотечный фонд укомплектован печатными изданиями из расчета не менее 50 экземпляров каждого из изданий основной литературы, перечисленной в рабочих программах дисциплин (модулей), практик и не менее 25 экземпляров дополнительной литературы на 100 обучающихся.

Обучающимся обеспечен доступ к электронным ресурсам и информационным справочным системам, перечисленным в таблице 1.

Таблица 1

№	Наименование ресурса	Объем фонда электронных ресурсов	Доступ	Реквизиты договоров с правообладателями
1.	Электронная библиотека диссертаций РГБ	885352 экз.	Доступ с компьютеров читальных залов библиотеки, подключенных к ресурсу	Договор №1330/0208-14 от 02.12.2014
2.	СПС «КонсультантПлюс»	2007691 экз.	По сети УГАТУ	Договор 1392/0403 -14 от 10.12.14
3.	СПС «Гарант»	6139026 экз.	Доступс компьютеров читальных залов библиотеки, подключенных к ресурсу	ООО «Гарант-Регион, договор № 3/Б от 21.01.2013 (пролонгирован до 08.02.2016.)
4.	ИПС «Технорма/Документ»	36939 экз.	Локальная установка:библиотека УГАТУ -5 мест; кафедра стандартизации и метрологии-1 место; кафедра начертательной геометрии и черчения-1 место	Договор № АОСС/914-15 № 989/0208-15 от 08.06.2015.
5.	Научная электронная библиотека eLIBRARY* http://elibrary.ru/	9169 полнотекстов ых журналов	С любого компьютера, имеющего выход в Интернет, после регистрации в НЭБ на площадке библиотеки УГАТУ	ООО «НАУЧНАЯ ЭЛЕКТРОННАЯ БИБЛИОТЕКА». № 07-06/06 от 18.05.2006
6.	Тематическая коллекция полнотекстовых журналов «Mathematics» издательства Elsevier http://www.sciencedirect.com	120 наимен. журнал.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	Договор №ЭА-190/0208-14 от 24.12.2014 г.

7.	Научные полнотекстовые журналы издательства Springer* http://www.springerlink.com	1900 наимен. журнал.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	Доступ открыт по гранту РФФИ
8.	Научные полнотекстовые журналы издательства Taylor& Francis Group* http://www.tandfonline.com/	1800 наимен. журнал.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и Государственной публичной научно-технической библиотекой России (далее ГПНТБ России)
9.	Научные полнотекстовые журналы издательства Sage Publications*	650 наимен. журнал.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и ГПНТБ России
10.	Научные полнотекстовые журналы издательства Oxford University Press* http://www.oxfordjournals.org/	275 наимен. журналов	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и ГПНТБ России
11.	Научный полнотекстовый журнал Science The American Association for the Advancement of Science http://www.sciencemag.org	1 наимен. журнала.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и ГПНТБ России
12.	Научный полнотекстовый журнал Nature компании Nature Publishing Group* http://www.nature.com/	1 наимен. журнала	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и ГПНТБ России
13.	Научные полнотекстовые журналы Американского института физики http://scitation.aip.org/	18 наимен. журналов	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и ГПНТБ России
14.	Научные полнотекстовые ресурсы Optical Society of America* http://www.opticsinfobase.org/	22 наимен. журн.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	В рамках Государственного контракта от 25.02.2014 г. №14.596.11.0002 между Министерством образования и науки и ГПНТБ России
15.	База данных GreenFile компании EBSCO* http://www.greeninfoonline.com	5800 библиографич. записей, частично с полными текстами	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	Доступ предоставлен компанией EBSCO российским организациям-участникам консорциума НЭЙКОН (в том числе УГАТУ - без подписания лицензионного договора)

16.	Архив научных полнотекстовых журналов зарубежных издательств*- Annual Reviews (1936-2006); Cambridge University Press (1796-2011); цифровой архив журнала Nature (1869- 2011) ; Oxford University Press (1849–1995) ;SAGE Publications (1800-1998); цифровой архив журнала Science (1880 -1996); Taylor & Francis (1798-1997); Институт физики Великобритании The Institute of Physics (1874-2000)	2361 наимен. журн.	С любого компьютера по сети УГАТУ, имеющего выход в Интернет	Доступ предоставлен российским организациям-участникам консорциума НЭЙКОН (в том числе УГАТУ - без подписания лицензионного договора)

* Периодические издания получены по Гранту и на баланс библиотеки не принимались.

Кафедра, реализующая образовательную программу подготовки научно-педагогических кадров высшей квалификации, обеспечена необходимым комплектом программного обеспечения:

Программный комплекс – операционная система Microsoft Windows (№ договора ЭФ-193/0503-14, 1800 компьютеров, на которые распространяется право пользования)

Программный комплекс –Microsoft Office (№ договора ЭФ-193/0503-14, 1800 компьютеров, на которые распространяется право пользования)

Kaspersky Endpoint Security для бизнеса (« лицензии 13C8-140128-132040, 500 users).

Dr.Web® Desktop Security Suite (КЗ) +ЦУ (AH99-VCUN-TPPJ-6k3L, 415 рабочих станций)

Среда разработки программ на ассембере SimpleASM (свободное программное обеспечение).

6.4 Методические указания к практическим занятиям

Занятия 1, 2. Команды пересылки данных, режимы адресации

Основные команды пересылки данных

Две основные команды пересылки данных:

mov <операнд-приемник>, <операнд-источник>

xchg <операнд1>, < операнд2>

Команда mov пересылает данные из <операнда-источника> в <операнд-приемник>. Команда xchg обменивает данные заданные операндами 1 и 2.

Режимы адресации

Режимы адресации, используемые как с командами пересылки данных, так и с большинством других команд:

1. Регистровая адресация. При использовании такой адресации процессор извлекает операнд из регистра или загружает его в регистр. Например, команда mov eax, ecx копирует содержимое регистра ecx в регистр eax.

2. Непосредственная адресация позволяет использовать константы в качестве операнда-источника. Эта константа содержится в команде (куда она помещается Ассемблером), а не в регистре или ячейке памяти. Например, команда mov cx, 500 загрузит значение 500 в регистр cx.

Константа также может быть определена оператором equ:

const equ 500

mov cx, const

3. Прямая адресация. При данном режиме адресации адрес (относительно сегментного регистра) является составной частью команды (так же, как значения при непосредственной адресации). Обычно прямая адресация применяется, если операндом служит метка. Например:

```
section .data
data1 db 0ffh
…
section .text
mov al, [data1]
```

В данном фрагменте последняя команда загрузит в регистр al данные (байт 0ffh), расположенные по адресу, соответствующему метке data1.

Основные операторы ассемблера, используемые для описания данных в памяти: db, dw, dd, dq – соответственно для байтов (8 бит), слов (16 бит), двойных слов (32 бита), учетверенных слов (64 бита).

4. При косвенной регистровой адресации адрес содержится в любом регистре общего назначения, кроме ebp и esp. В предыдущем примере можно заменить команду mov al, [data1] следующей последовательностью:

```
mov ecx, data1
mov al, [ecx]
```

Сначала адрес метки data1 загружается в регистр ecx (синтаксис соответствует ассемблеру nasm). В некоторых ассемблерах такую операцию нужно записывать следующим образом: mov ecx, offset data1.

5. При косвенной базовой адресации со смещением адрес формируется как сумма содержимого регистра (как при косвенной регистровой адресации) и смещения, задаваемого в команде.

```
section .data
data2 dw 1234h, 5678h
…
section .text
mov ecx, data2
mov bx, [ecx+2]
```

Приведенная последовательность команд загрузит в регистр bx слово 5678h.

6. Косвенная индексная адресация со смещением. Данный вид адресации отличается от предыдущего использованием масштабирования содержимого индексного регистра в 2, 4, 8 раз. Это позволяет адресовать элементы массивов по 1, 2, 4, 8 байт. Например, можно заменить две последние строки предыдущего примера:

```
mov ecx, 1
mov bx, [data2+ecx*2]
```

В ecx помещается индекс элемента в массиве (в примере=1). Адрес начала массива – data2.

7. Косвенная базовая индексная адресация. При этом виде адресации адрес формируется как сумма содержимого двух регистров общего назначения: базового и индексного. Можно использовать масштабирование содержимого индексного регистра. Для примера можно модифицировать пример из пункта 5, заменив последние две строки фрагментом:

```
mov esi, data2
mov ebx, 1
mov bx, [ebx+esi*2]
```

В других трансляторах ассемблера возможны альтернативные варианты записи данного вида адресации.

8. Косвенная базовая индексная адресация со смещением. В данном виде адресации адрес формируется как сумма трех составляющих: базового регистра, индексного регистра и смещения, заданного в команде.

```
section .data
data3 db 12h, 34h, 56h, 78h
…
section .text
mov edx, data3
mov esi, 2
```

mov al, [edx+esi+1]

Приведенный фрагмент загрузит в регистр al байт 78h.

Основные ограничения команд mov и xchg:

1. Нельзя одной командой mov осуществить пересылку из одной ячейки памяти в другую, или одной командой xchg обменять содержимое двух ячеек памяти. Если такая необходимость возникает, то нужно использовать в качестве промежуточного буфера регистр общего назначения.
2. Нельзя загрузить в сегментный регистр значение непосредственно из памяти. Для выполнения такой загрузки нужно использовать промежуточный объект. Это может быть регистр общего назначения или стек.
3. Нельзя переслать одной командой содержимое одного сегментного регистра в другой сегментный регистр. Выполнить такую пересылку можно, используя в качестве промежуточных регистры общего назначения (или через стек).
4. Нельзя использовать сегментный регистр cs в качестве операнда назначения.

Важная особенность команды mov в случае, если адрес в регистре является приемником, а источник – константа. Например:

mov [ecx], 12h

В записи данной команды есть неоднозначность – каков размер операнда источника? Для устранения данной неоднозначности можно привести модификатор byte, word, dword.

mov byte [ecx], 12h

В некоторых ассемблерах кроме размера операнда нужно указывать и оператор ptr:

mov byte ptr[ecx], 12h

Команды работы со стеком

Стек – область памяти, специально выделяемая для временного хранения данных программы. Запись и чтение данных в стеке осуществляется в соответствие с принципом LIFO (Last In First Out – «последним пришел, первым ушел»). По мере записи данных в стек указатель стека esp/sp уменьшается, при чтении – увеличивается. Регистр esp/sp всегда указывает на вершину стека, т.е. содержит смещение, по которому в стек был занесен последний элемент или откуда будет прочитан очередной элемент (команды работы со стеком неявно изменяют этот регистр).

Команда push <источник> записывает значение источника в вершину стека.

Команда pop <приемник> запись значения из вершины стека в приемник, т.е. по отношению к push это обратная операция.

Для работы с регистром флагов есть две специальные команды:

pushf – сохранение регистра флагов в стеке. Можно явно указать размер операнда: pushfw – сохранение регистра flags (16 бит), pushfd – сохранение регистра eflags (32 бита).

popf – чтение регистра флагов из стека. Можно явно указать размер операнда: popfw – чтение регистра flags (16 бит), popfd – сохранение регистра eflags (32 бита).

Работа с системой SASM

При изучении данной дисциплины используется система программирования на ассемблере SASM (SimpleASM). Это интегрированная среда, включающая редактор, трансляторы ассемблера (NASM, GAS, FASM, MASM), отладчик (GDB). При выполнении заданий предполагается использование ассемблера NASM (Netwide Assembler), хотя нет принципиальных препятствий для использования других вариантов.

Для корректной работы с отладчиком в начале каждой секции .text нужно размещать следующий фрагмент:

```
section .text
global CMAIN
CMAIN:
mov ebp, esp
```

Заканчиваться программа должна командой ret.

В программах, приведенных в данных методических указаниях вышеприведенные фрагменты каждый раз не приводятся, но их наличие подразумевается.

Для просмотра содержимого памяти в отладчике нужно выбрать Отладка>Память (Ctrl+M). В поле «Переменная или выражение» указывается имя переменной или массива. Если это массив, нужно задать его размер в соответствующем поле. Возможно выбрать формат отображения (по умолчанию Smart). Также нужно задать размер отдельного элемента (b – байт, w – слово, d – двойное слово, q – слово длиной 32 байта).

Отладчик позволяет выполнять трассировку с заходом в подпрограммы (F11) или без захода (F10). Можно определять точки останова (F8 или клик мышью рядом с номером соответствующей строки). При работе отладчика рекомендуется включить отображение регистров (Ctrl+R) и, при необходимости, памяти (Ctrl+M).

Примеры заданий с решениями:

1. Приведите пример команды обмена содержимого двух 8-битных регистров.

Ответ: xchg al, bl

2. Приведите последовательность команд, пересылающих содержимое одной 32-битной ячейки памяти в другую с использованием прямой адресации.

Ответ:

```
section .data
data1 dd 12345678h
data2 dd 0
…
section .text
mov eax, [data1]
mov [data2], eax
```

3. Приведите последовательность команд, в которой регистр eax используется как промежуточный для записи некоторого значения в память. Используйте команды работы со стеком в обрамлении данной последовательности, чтобы в результате содержимое eax осталось неизменным.

Ответ:

```
section .data
data1 dd 0
…
section .text
push eax
mov eax, 1
mov [data1], eax
pop eax
```

Занятие 3. Арифметические команды

На данном практическом занятии рассматриваются команды двоичной арифметики, которые есть практически в любом процессоре. Не будут описываться уникальные для x86 команды, не имеющие аналогов в DSP.

Команды сложения

Команда add <операнд1>, <операнд2> выполняется следую-щим образом: <операнд1>=<операнд1>+<операнд2>.

Команда adc <операнд1>, <операнд2> отличается учетом флага переноса: <операнд1>=<операнд1>+<операнд2>+cf.

inc <операнд> – операция инкремента, то есть <операнд>=<операнд>+1.

Команды вычитания и сравнения

Команда sub <операнд1>, <операнд2> выполняется следую-щим образом: <операнд1>=<операнд1>-<операнд2>.

Команда sbb <операнд1>, <операнд2> производит вычитание с учетом заема: <операнд1>=<операнд1>-<операнд2>-cf.

dec <операнд> – операция декремента, то есть <операнд>=<операнд>-1.

cmp <операнд1>, <операнд2> выполняется аналогично команде sub, но, в отличие от sub, значение операнда1 не изменяется. Используется для сравнения чисел. По результатам сравнения устанавливаются соответствующие флаги.

Команды умножения

Команда mul <сомножитель1> служит для умножения чисел без знака. Местоположение второго сомножителя и результата задается неявно и зависит от размера сомножителей.

Размер сомножителя 1	Сомножитель 2	Результат
Байт	al	ax
Слово	ax	dx:ax (старшие 16 бит в dx. младшие – в ax)
Двойное слово	eax	edx:eax (старшие 32 бита в edx. младшие – в eax)

Для умножения чисел со знаком используется команда

imul <операнд1>[, <операнд2>, <операнд3>]

Команда imul имеет три формы, различающиеся количеством операндов:

1) С одним операндом – требует явного указания местоположения только одного сомножителя, который может быть расположен в ячейке памяти или регистре. Местоположение второго сомножителя и результата задается неявно и зависит от размера сомножителей (соответствует вышеприведенной таблице для команды mul).

2) С двумя операндами – первый операнд определяет местоположение первого сомножителя (регистр), на его место будет записан результат. Второй операнд определяет местоположение второго сомножителя.

3) С тремя операндами – первый операнд определяет местоположение результата (регистр), второй операнд – местоположение первого сомножителя, третий операнд может быть непосредственно заданным значением размером в байт, слово или двойное слово.

При использовании двух- и трехоперандных вариантов может возникнуть ситуация, когда результат не полностью поместится в регистре-приемнике, при этом флаги cf=of=1. Если результат помещается в регистр приемник, то флаги cf=of=0.

Команды деления

Команда div <делитель> служит для деления чисел без знака. Делитель может находится в памяти или в регистре и иметь размер 8, 16 или 32 бита. Местонахождение делимого фиксировано и зависит от размера операндов.

Делимое	Размер делителя	Частное	Остаток
ax	8 бит	al	ah
dx:ax (старшие 16 бит в dx. младшие – в ax)	16 бит	ax	dx
edx:eax (старшие 32 бита в edx. младшие – в eax)	32 бита	eax	edx

После выполнения команды содержимое флагов неопределенно и возможно возникновение прерывания с номером 0, называемого «деление на ноль». Этот вид прерывания относится к так называемым исключениям. Эта разновидность прерываний возникает внутри микропроцессора из-за некоторых аномалий во время вычислительного процесса. Прерывание 0 «деление на ноль»при выполнении команды div может возникнуть по одной из следующих причин:

1) Делитель равен нулю.

2) Частное не входит в отведенную под него разрядную сетку.

Для деления чисел со знаком предназначена команда
idiv <делитель>
Размещение делителя и результата аналогично команде div. При делении на 0 и в случае, если частное не входит в отведенную для него разрядную сетку возникнет прерывание 0.

Примеры заданий с решениями:

1. Приведите последовательность команд, выполняющую сложение двух 16-битных чисел, непосредственно заданных в командах.

Ответ:
mov ax, 1234h
add ax, 5678h

2. Приведите последовательность команд, выполняющую вычитание 16-битных чисел, непосредственно заданных в командах.

Ответ:
mov ax, 5678h
sub ax, 1234h

3. Приведите последовательность команд, вычисляющую в регистре ax результат беззнакового умножения ax=bl*al.

Ответ:
mov al, 2
mov bl, 3
mul bl

4. Приведите последовательность команд, вычисляющую в регистрах dx:ax результат знакового умножения dx:ax=bx*ax.

Ответ:
mov ax, -2
mov bx, 3
imul bx

Занятие 3. Логические команды и команды сдвига

Логические команды

Команда and <операнд1>, <операнд2> выполняет поразрядно логическую операцию И (конъюнкцию) над битами операндов 1 и 2. Результат записывается на место операнда 1.

С использованием данной команды можно обнулить заданные разряды в операнде1. При этом операнд2 будет выполнять роль маски – нулевым значениям маски будут соответствовать выключенные разряды операнда1.

Команда or <операнд1>, <операнд2> выполняет поразрядно логическую операцию ИЛИ (дизъюнкцию) над битами операндов 1 и 2. Результат записывается на место операнда 1.

С использованием данной команды можно установить заданные разряды в операнде1 в состояние 1. При этом операнд2 выполняет роль маски – единичным значениям маски будут соответствовать включенные разряды операнда1.

Команда xor <операнд1>, <операнд2> выполняет поразрядно логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над битами операндов 1 и 2. Результат записывается на место операнда 1.

С использованием данной команды можно поменять значение заданных разрядов в операнде1. При этом операнд2 выполняет роль маски – единичным значениям маски будут соответствовать инвертированные разряды операнда1.

Команда test <операнд1>, <операнд2> выполняет поразрядно логическую операцию И (конъюнкцию) над битами операндов 1 и 2. В отличие от команды and, операнды остаются

неизменными, но меняется значение флагов zf, sf, pf. Команда служит для проверки содержимого операндов с использованием конъюнкции.

Команды сдвига

Все эти команды выполняют сдвиг операнда на столько разрядов, сколько задается <числом_сдвигов>. <Число сдвигов> может быть или непосредственным операндом или задаваться в регистре cl. На практике значение <числа_сдвигов> лежит в диапазоне от 0 до 31. Данные команды можно разделить на две группы: линейного и циклического сдвига.

1) Команды линейного сдвига.

shl <операнд>, <число_сдвигов> – логический сдвиг влево. Содержимое операнда сдвигается влево на количество бит, задаваемое значением <число_сдвигов>. Справа в «освободившиеся» разряды вписываются нули.

shr <операнд>, <число_сдвигов> – логический сдвиг вправо. Содержимое операнда сдвигается вправо на количество бит, задаваемое значением <число_сдвигов>. Слева в «освободившиеся» разряды вписываются нули.

sal <операнд>, <число_сдвигов> – арифметический сдвиг влево. Содержимое операнда сдвигается влево на количество бит, задаваемое значением <число_сдвигов>. Справа в «освободившиеся» разряды вписываются нули. Данная команда не сохраняет знак числа и практически аналогична команде shl.

sar <операнд>, <число_сдвигов> – арифметический сдвиг вправо. Содержимое операнда сдвигается вправо на количество бит, задаваемое значением <число_сдвигов>. Слева «освободившиеся» разряды заполняются значением знакового бита, т.е. при арифметическом сдвиге вправо знак числа сохраняется.

Все перечисленные команды пересылают «выдвигаемый» из операнда бит во флаг cf.

При сдвигах влево на 1 разряд также имеет смысл значение флага of:

of=1, если текущее значение флага cf и выдвигаемого слева бита операнда различны;

of=0, если текущее значение флага cf и выдвигаемого слева бита операнда совпадают.

При сдвигах вправо of всегда равен 0.

2) Команды циклического сдвига.

rol <операнд>, <число_сдвигов> – циклический сдвиг влево. Содержимое операнда сдвигается влево на количество бит, задаваемое значением <число_сдвигов>. «Выдвигаемые» слева биты записываются в тот же операнд справа.

ror <операнд>, <число_сдвигов> – циклический сдвиг вправо. Содержимое операнда сдвигается вправо на количество бит, задаваемое значением <число_сдвигов>. «Выдвигаемые» справа биты записываются в тот же операнд слева.

Очередной выдвигаемый двумя вышеприведенными командами бит записывается также во флаг cf.

rcl <операнд>, <число_сдвигов> – циклический сдвиг влево через флаг cf. Содержимое операнда сдвигается влево на количество бит, задаваемое значением <число_сдвигов>. «Выдвигаемые» слева биты сначала попадают во флаг cf, а затем записываются в тот же операнд справа.

rcr <операнд>, <число_сдвигов> – циклический сдвиг вправо через флаг cf. Содержимое операнда сдвигается вправо на количество бит, задаваемое значением <число_сдвигов>. «Выдвигаемые» справа биты сначала попадают во флаг cf, а затем записываются в тот же операнд слева.

Команды rcl и rcr отличаются от rol и ror тем, что сдвигаемый бит не сразу записывается в операнд с другой стороны, а сначала записывается во флаг переноса cf и только при следующем сдвиге попадает в операнд.

Дополнительные команды сдвига

Команда shld <операнд1>, <операнд2>, <число_сдвигов> выполняет сдвиг влево операнда1. Число сдвинутых разрядов определяется числом_сдвигов. Младшие биты операнда1, освободившиеся при сдвиге, заменяется старшими битами операнда2. Старшие биты операнда1, «выдвигаемые» из него, «проходят» через флаг cf. Значение операнда2 не изменяется.

Команда shrd <операнд1>, <операнд2>, <число_сдвигов> выполняет сдвиг вправо операнда1. Число сдвинутых разрядов определяется числом_сдвигов. Старшие биты операнда1, освободившиеся при сдвиге, заменяются младшими битами операнда2. Младшие биты операнда1, «выдвигаемые» из него, «проходят» через флаг cf. Значение операнда2 не изменяется.

Примеры задач с решениями:

1. Какой командой можно обнулить биты 0, 3, 5 регистра al?

Ответ: and al, 11010110b

2. Какой командой можно инвертировать биты 0 и 1 регистра dl?

Ответ: xor dl, 11b

3. Приведите последовательность команд для умножения знакового числа в регистре al на 2 с размещением результата в регистре ax?

Ответ:

sal al, 1

sbb ah, ah

Занятие 4. Команды передачи управления

Рассмотренные на предыдущих занятиях команды позволяют создавать программы, работающие линейно. Выполняется некоторое действие по преобразованию или пересылке данных, после чего процессор передает управление следующей команде. На практике в программах практически всегда есть точки, в которых нужно принять решение о том, какая команда будет выполняться следующей. Это решение может быть:

безусловным – в данной точке необходимо передать управление не следующей команде, а другой, находящейся на некотором удалении от следующей команды;

условным – решение о том, какая команда будет выполняться следующей, принимается на основе анализа некоторых условий или данных.

На данном практическом занятии не затрагиваются особенности разных вариантов команд передачи управления, связанные с уникальными для x86 сегментными регистрами.

Команды безусловной передачи управления

Команда безусловного перехода jmp <адрес_перехода>. После её выполнения следующая выполняемая команда будет выбрана начиная с адреса <адрес_перехода>. Адрес перехода м.б. задан непосредственно или содержаться в регистре или ячейке памяти (два последних случая соответствуют косвенному переходу). При программировании на ассемблере вместо непосредственного задания адреса обычно указывается метка, которую программа ассемблера при трансляции преобразует в адрес.

Для организации процедур служат две команды:

call <адрес_процедуры> – вызов процедуры с сохранением адреса возврата в стеке. Вместо адреса в программах на ассемблере обычно указывается метка, соответствующая началу процедуры.

ret – возврат из процедуры. Из стека извлекается адрес, помещенный туда командой call и выполнение программы продолжается с этого адреса.

Условные переходы

Команды условного перехода имеют следующий синтаксис:

jcc <метка_перехода>

Вместо cc подставляется конкретное условие, анализируемое командой.

Значение аббревиатур в команде jcc

Мнемоническое обозначение	Английское значение	Русский перевод	Тип операндов
E или e	equal	равно	любые
N или n	not	не	любые

G или g	greater	больше	числа со знаком
L или l	less	меньше	числа со знаком
A или a	above	выше («больше»)	числа со знаком
B или b	below	ниже («меньше»)	числа со знаком

Переходы по результатам сравнения двух операндов можно описать следующим образом:

Команды условных переходов после команд cmp и sub

Типы операндов	Мнемокод команды условного перехода	Критерий условного перехода	Значения флагов для осуществления перехода
любые	je	операнд1=операнд2	zf=1
любые	jne	операнд1<>операнд2	zf=0
со знаком	jl/jnge	операнд1<операнд2	sf<>of
со знаком	jle/jng	операнд1<=операнд2	sf<>of или zf=1
со знаком	jg/jnle	операнд1>операнд2	sf=of и zf=0
со знаком	jge/jnl	операнд1=>операнд2	sf=of
без знака	jb/jnae	операнд1<операнд2	cf=1
без знака	jbe/jna	операнд1<=операнд2	cf=1 или zf=1
без знака	ja/jnbe	операнд1>операнд2	cf=0 и zf=0
без знака	jae/jnb	операнд1=>операнд2	cf=0

Условные переходы могут осуществляться не только по результатам арифметического сравнения двух операндов, но и, например, если результат операции нулевой и т.п.

Команды условного перехода и флаги

Название флага	Команда условного перехода	Значение флага для осуществления перехода
флаг переноса cf	jc	cf=1
флаг четности pf	jp	pf=1
флаг нуля zf	jz	zf=1
флаг знака sf	js	sf=1
флаг переполнения of	jo	of=1
флаг переноса cf	jnc	cf=0
флаг четности pf	jnp	pf=0
флаг нуля zf	jnz	zf=0
флаг знака sf	jns	sf=0
флаг переполнения of	jno	of=0

Команда условного перехода проверяющая состояние регистра ecx/cx

Вышеприведенные команды условных переходов проверяли состояние флагов, установленных предыдущими командами. Есть специальная команда, проверяющая регистр ecx/cx на равенство нулю и выполняющая переход если это условие выполняется:

jcxz/jecx <метка_перехода> – переход, если cx/ecx равен нулю.

Команды организации циклов

Для организации циклов с числом повторений, заданных в регистре cx/ecx служит команда loop <метка_перехода>

Работа команды заключается в выполнении следующих действий:

1) Декремент регистра cx/ecx;

2) Сравнение cx/ecx с нулем. Если (cx/ecx)>0, то управление передается на метку перехода. Если (cx/ecx)=0, то управление передается на следующую после loop команду.

Для принудительного указания разрядности используемого для счетчика регистра нужно указывать префикс a16 или a32:

a16 loop <метка_перехода> – использует cx в качестве счетчика;

a32 loop <метка_перехода> – использует ecx в качестве счетчика. По умолчанию при использовании SASM в Windows в loop используется именно ecx, поэтому префикс a32 можно не указывать.

Также есть две команды, дополнительно проверяющие состояние флага zf.

loope/loopz <метка_перехода> – повторить цикл пока cx/ecx>0 и zf=1. Команды loope и loopz – синонимы.

loopne/loopnz <метка_перехода> – повторить цикл пока cx/ecx>0 и zf=0. Команды loopne и loopnz – синонимы.

Команды loope/loopz и loopne/loopnz расширяют действие команды loop тем, что дополнительно анализируют флаг zf, что дает возможность организовать досрочный выход из цикла, используя этот флаг в качестве индикатора.

Примеры задач с решениями:

1. Приведите последовательность команд, сравнивающую два знаковых числа в регистрах eax и ebx (eax и ebx не должны измениться). Если eax<ebx, то вызвать процедуру, задающую значение регистра ecx=1.

Ответ:

```
cmp eax, ebx
jnl metka1
call proc1
metka1:
…
proc1:
mov ecx, 1
ret
```

2. Приведите последовательность команд, прибавляющую 3 к содержимому регистра dl до тех пор, пока dl не станет равен нулю.

Ответ:

```
metka1:
add dl, 3
jnz metka1
```

3. Приведите последовательность команд с использованием команды организации цикла, выполняющую 10 раз прибавление числа 2 к содержимому регистра dx.

Ответ:

```
mov ecx, 10
metka1:
add dx, 2
loop metka1
```

Занятие 6. Свертка дискретных сигналов, реализация вычисления свертки на ассемблере

Круговая (периодическая) свёртка сигналов.

Если последовательности $x_1(n)$ и $x_2(n)$ периодические с периодом N, то их круговая свёртка будет определяться следующим выражением:

$$y(n)=\sum_{m=0}^{N-1}x_1(m)x_2(n-m)=\sum_{m=0}^{N-1}x_1(n-m)x_2(m)$$

Последовательность $y(n)$ также является периодической с периодом N.

Круговую свёртку также можно вычислить при помощи ДПФ:

1) Вычислить ДПФ $\{x_1(n)\} = X_1(k)$ и ДПФ $\{x_2(n)\} = X_2(k)$.

2) Вычислить произведение $Y(k) = X_1(k) \cdot X_2(k)$.

3) Найти результат круговой свертки $y(n) = \text{ОДПФ}\{Y(k)\}$

Т.е. ДПФ свёртки равно произведению ДПФ сворачиваемых сигналов. На практике для уменьшения объема вычислений при реализации указанного алгоритма используются алгоритмы быстрого преобразования Фурье (БПФ).

Линейная (апериодическая) свёртка линейных сигналов

Пусть $x_1(n)$ и $x_2(n)$ конечные последовательности, длиною N_1 и N_2, тогда их линейная свёртка определяется выражением:

$$y(n) = \sum_{m=0}^{n} x_1(m) \cdot x_2(n-m) = \sum_{m=0}^{n} x_1(n-m) \cdot x_2(m)$$

Номер (индекс) максимального элемента:

$n_{\max} = N_1 + N_2 - 2$

Длина линейной свёртки равна $N_1 + N_2 - 1$.

Вышеприведенный алгоритм вычисления круговой свертки с использованием ДПФ может быть применен и для вычисления линейной свертки. Для этого нужно дополнить исходные последовательности $x_1(n)$ и $x_2(n)$ длиной N_1 и N_2 нулями до длины N_1+N_2-1. Затем можно вычислить свертку с использованием алгоритма с ДПФ.

Команды MMX

Для цифровых сигнальных процессоров характерно наличие специализированных команд, повышающих эффективность реализации алгоритмов ЦОС. В процессорах x86 подобные команды появились впервые в процессорах Pentium MMX. В следующих поколениях процессоров x86 набор команд для обработки сигналов развивался и расширялся.

Рассмотрим некоторые команды набора MMX, позволяющие увеличить скорость выполнения алгоритмов ЦОС. Увеличение скорости выполнения операций происходит за счет использования идеологии SIMD (Single Instruction Multiple Data), как и в некоторых сигнальных процессорах – одна команда выполняет операцию над вектором данных. В технологии MMX полный размер обрабатываемого вектора – 64 бита. Эти 64 бита могут по разному трактоваться в различных командах – как 8 8-битных значений, 4 16-битных или 2 32-битных. В качестве регистров MMX используются регистры сопроцессора, обозначаемые в данном случае как mm0-mm7. Набор команд MMX довольно обширен и полное его рассмотрение затруднительно в рамках данного практического занятия, поэтому рассмотрим только несколько полезных для ЦОС команд.

Команда pmaddwd <приемник>, <источник> выполняет одну из базовых операций ЦОС – знаковое умножение с накоплением. Исходные данные – 16-разрядные числа со знаком. Источник – MMX-регистр или ячейка памяти, приемник – MMX-регистр. Работу данной команды можно описать так (цифры – номера разрядов):

<приемник0-31>=<приемник0-15>*<источник0-15>+
+<приемник16-31>*<источник16-31>;
<приемник32-63>=<приемник32-47>*<источник32-47>+
+<приемник48-63>*<источник48-63>.

Команда movq <приемник>, <источник> пересылает 64 бита из источника в приемник. Источник и приемник – MMX-регистр или ячейка памяти, но одновременно только один из операндов может адресовать ячейку памяти.

Команда movd <приемник>, <источник> пересылает 32 бита из источника в приемник. Один из операндов, источник или приемник, но не одновременно, должен быть MMX-регистром. Другой операнд должен быть 32-разрядным регистром общего назначения 32-разрядной ячейкой памяти.

Команда packsswb <приемник>, <источник> – команда упаковки со знаковым насыщением двух 32-разрядных слов из приемника и двух 32-разрядных слов из источника в 4 16-разрядных слова в приемнике. Работу данной команды можно описать так (цифры – номера разрядов):

<приемник0-15>=<приемник0-31>;
<приемник16-31>=<приемник32-63>;
<приемник32-47>=<источник0-31>;
<приемник48-63>=<источник32-63>.

Команда pshufw <приемник>, <источник>, <маска> перераспределяет упакованные слова из источника в приемник в соответствии с маской. Источник – MMX-регистр или ячейка памяти, приемник – MMX-регистр. Маска представляет собой байт, в котором:

биты 0 и 1 определяют, какое из четырех слов источника нужно разместить в 0 и 1 байте приемника;

биты 2 и 3 определяют, какое из четырех слов источника нужно разместить в 2 и 3 байте приемника;

биты 4 и 5 определяют, какое из четырех слов источника нужно разместить в 4 и 5 байте приемника;

биты 6 и 7 определяют, какое из четырех слов источника нужно разместить в 6 и 7 байте приемника.

Команда emms – очистка MMX-контекста (очистка стека регистров MMX/FPU и установка единичных значений в регистре тегов). Данная команда обязательно должна присутствовать в конце блока команд, использующих MMX-регистры, если в дальнейшем предполагается использование команд математического сопроцессора.

Примеры задач с решениями:

1. Даны периодические последовательности:

$x_1(n)=\{1,2,3\};\ x_2(n)=\{4,5,6\}$

Вычислите круговую свертку последовательностей $x_1(0)$ и $x_2(0)$.

Решение:

$y(0)=x_1(0)\cdot x_2(0)+ x_1(1)\cdot x_2(-1)+ x_1(2)\cdot x_2(-2)=4+12+15=31;$

$y(1)=x_1(0)\cdot x_2(1)+ x_1(1)\cdot x_2(0)+ x_1(2)\cdot x_2(-1)=5+8+18=31;$

$y(2)=x_1(0)\cdot x_2(2)+ x_1(1)\cdot x_2(1)+ x_1(2)\cdot x_2(0)=6+10+12=28.$

$y(n)=\{31; 31; 28\}.$

Можно также представить исходные данные и результат в виде таблицы:

n	-3	-2	-1	0	1	2
$x_1(n)$	1	2	3	1	2	3
$x_2(n)$	4	5	6	4	5	6
$y(n)$	31	31	28	31	31	28

2. Даны конечные последовательности

$x_1(n)=\{1,2,3\};\ x_2(n)=\{4,5,6\}$

Вычислите линейную свертку последовательностей $x_1(0)$ и $x_2(0)$.

Решение:

Длины последовательностей $x_1(0)$ и $x_2(0)$ соответственно $N_1=3$ и $N_2=3.$

Найдем длину результата линейной свертки. Она будет равна 3+3-1=5. Соответственно максимальный индекс $n_{\max}=3+3-2=4.$

$y(0)=x_1(0)\cdot x_2(0)=4$

$y(1)=x_1(0)\cdot x_2(1)+x_1(1)\cdot x_2(0)=5+8=13$

$y(2)=6+10+12=28$

$y(3)=27$

$y(4)=18$

$y(n)=\{4,13,28,27,18\}.$

3. Составьте программу вычисляющую линейную свертку последовательностей $x_1(n)=\{1,2,3\};\quad x_2(n)=\{4,5,6\}$ (результат свертки рассчитан в примере 2 и равен

$y(n)=\{4,13,28,27,18\}$). Исходные данные и результат считать 8-разрядными. Для упрощения программы дополните исходные данные нулями до длины результата.

Ответ:

```
N1 equ 3
N2 equ 3
section .data
x1 db 1, 2, 3, 0, 0
x2 db 4, 5, 6, 0, 0
y db 0, 0, 0, 0, 0

section .text
mov ebx, x1
mov esi, x2
mov edi, y
mov ebp, N1+N2-1
mov ecx, 1
metka2:
mov dl,0
push ecx
push ebx
push esi
metka1:
mov al, [ebx]
inc ebx
imul byte[esi]
dec esi
add dl, al
loop metka1
mov [edi], dl
inc edi
pop esi
inc esi
pop ebx
pop ecx
inc ecx
dec ebp
jnz metka2
```

4. Приведите последовательность команд, загружающую 8 байт из ячейки памяти в mm0, и затем копирующую младшие 32 бита mm0 в eax.

Ответ:

```
section .data
data dq 1234567890abcdefh

section .text
movq mm0, [data]
movd eax, mm0
```

Занятие 7. Быстрое преобразование Фурье, реализация вычисления БПФ на ассемблере

Дискретное преобразование Фурье

Рассмотрим, что представляет собой спектр дискретного *периодического* сигнала.

Итак, пусть последовательность отсчетов {*x*(*k*)} является периодической с периодом *N*:

$x(k + N) = x(k)$ для любого k

Такая последовательность полностью описывается *конечным* набором чисел, в качестве которого можно взять произвольный фрагмент длиной N, например $\{x(k)\ k= 0,1,..., N- 1\}$. Поставленный в соответствие этой последовательности сигнал из смещенных по времени дельта-функций:

$$s(t) = \sum_{k=-\infty}^{\infty} x(k)\delta(t - kT) \tag{7.1}$$

также, разумеется, будет периодическим с минимальным периодом NT. Так как сигнал (7.1) является дискретным, его спектр должен быть *периодическим* с периодом $2\pi/T$. Так как этот сигнал является также и периодическим, его спектр, должен быть *дискретным* с расстоянием между гармониками, равным $2\pi/(NT)$.

Итак, периодический дискретный сигнал имеет периодический дискретный спектр; который также описывается конечным набором из N чисел (один период спектра содержит $\frac{2\pi}{T}/\frac{2\pi}{NT} = N$ гармоник).

Рассмотрим процедуру вычисления спектра периодического дискретного сигнала. Так как сигнал периодический, будем раскладывать его в *ряд Фурье*. Коэффициенты $X(n)$ этого ряда равны

$$\begin{aligned}
\dot{X}(n) &= \frac{1}{NT}\int_0^{NT} s(t)e^{-jw_n t}dt = \\
&= \frac{1}{NT}\int_0^{NT}\sum_{k=0}^{N-1} x(k)\delta(t - kT)e^{-jw_n t} = \\
&= \frac{1}{NT}\sum_{k=0}^{N-1} x(k)\int_0^{NT}\delta(t - kT)e^{-jw_n t} = \frac{1}{NT}\sum_{k=0}^{N-1} x(k)e^{-jw_n t} = \\
&= \frac{1}{NT}\sum_{k=0}^{N-1} x(k)\exp(-j\frac{2\pi nk}{N}).
\end{aligned} \tag{7.2}$$

Таким образом, формула для вычисления комплексных амплитуд гармоник представляет собой линейную комбинацию отсчетов сигнала.

В выражении (7.2) реальный масштаб времени фигурирует только в множителе $1/T$ перед оператором суммирования. При рассмотрении дискретных последовательностей обычно оперируют номерами отсчетов и спектральных гармоник без привязки к действительному масштабу времени и частоты. Поэтому множитель $1/T$ из (7.2) удаляют, то есть считают частоту дискретизации равной единице. Удаляют обычно и множитель $1/N$ (об этом см. замечание ниже). Получившееся выражение называется *дискретным преобразованием Фурье* (ДПФ; английский термин — Discrete Fourier Transform, DFT):

$$\dot{X}(n) = \sum_{k=0}^{N-1} x(k)\exp(-j\frac{2\pi nk}{N}). \tag{7.3}$$

Существует и *обратное* дискретное преобразование Фурье. Переход от дискретного спектра к временным отсчетам сигнала выражается следующей формулой:

$$x(k) = \frac{1}{N}\sum_{n=0}^{N-1} \dot{X}(n)\exp(j\frac{2\pi nk}{N}). \tag{7.4}$$

Это выражение отличается от формулы прямого ДПФ (7.3) лишь знаком в показателе комплексной экспоненты и наличием множителя $1/N$ перед оператором суммирования.

В размещении множителя $1/N$ в формулах (7.3) и (7.4) нет полного единства. В большинстве источников, а также в математических пакетах компьютерных программ (в том числе и в MATLAB) этот множитель фигурирует в формуле *обратного* ДПФ (7.4).

Алгоритм быстрого преобразования Фурье

Для вычисления одного коэффициента ДПФ по формуле (7.3) необходимо выполнить N комплексных умножений и сложений. Таким образом, расчет всего ДПФ, содержащего N коэффициентов, потребует N^2 пар операций «умножение-сложение». Число операций возрастает

пропорционально квадрату размерности ДПФ. Однако, если N не является простым числом и может быть разложено на множители, процесс вычислений можно ускорить, разделив анализируемый набор отсчетов на части, вычислив их ДПФ и объединив результаты. Такие способы вычисления ДПФ называются *быстрым преобразованием Фурье* (БПФ; английский термин — Fast Fourier Transform, FFT) и повсеместно используются на практике.

При реализации БПФ возможно несколько вариантов организации вычислений в зависимости от способа деления последовательности отсчетов на части (*прореживание* по времени либо по частоте) и от того, на сколько фрагментов производится разбиение последовательности на каждом шаге (*основание* БПФ).

БПФ с прореживанием по времени

Рассмотрим идею БПФ с прореживанием по времени (decimation in time, DIT) на примере деления набора отсчетов пополам.

Итак, пусть N — четное число. Выделим в формуле (7.3) два слагаемых, соответствующих элементам исходной последовательности с четными и нечетными номерами:

$$\dot{X}(n) = \sum_{m=0}^{N/2-1} x(2m) e^{-j\frac{2\pi 2mn}{N}} + \sum_{m=0}^{N/2-1} x(2m+1) e^{-j\frac{2\pi 2(m+1)n}{N}}.$$

Введем обозначения $y(m) = x(2m)$ и $z(m) = x(2m+1)$, а также вынесем из второй суммы общий множитель $e^{-j2\pi n/N}$:

$$\dot{X}(n) = \sum_{m=0}^{N/2-1} y(m) e^{-j\frac{2\pi mn}{N/2}} + e^{-j\frac{2\pi n}{N}} \sum_{m=0}^{N/2-1} z(m) e^{-j\frac{2\pi mn}{N/2}}. \qquad (7.5)$$

Две суммы в (7.5) представляют собой ДПФ последовательностей {y(m)} (отсчеты с четными номерами) и {z(m)} (отсчеты с нечетными номерами). Каждое из этих ДПФ имеет размерность $N/2$. Таким образом,

$$\dot{X}(n) = \dot{Y}(n) + e^{-j\frac{2\pi n}{N}} \cdot \dot{Z}(n), \qquad (7.6)$$

где $\dot{Y}(n)$ и $\dot{Z}(n)$ — ДПФ соответственно последовательностей отсчетов с четными и нечетными номерами:

$$\dot{Y}(n) = \sum_{m=0}^{N/2-1} y(m) e^{-j\frac{2\pi mn}{N/2}},$$

$$\dot{Z}(n) = \sum_{m=0}^{N/2-1} z(m) e^{-j\frac{2\pi mn}{N/2}},$$

Так как ДПФ размерности $N/2$ дает лишь $N/2$ спектральных коэффициентов, непосредственно использовать формулу (7.6) можно только при $0 \le n < N/2$. Для остальных n ($N/2 \le n < N$) следует воспользоваться периодичностью спектра дискретного сигнала (и, соответственно, периодичностью результатов ДПФ):

$$\dot{Y}\left(n + \frac{N}{2}\right) = \dot{Y}(n),$$

$$\dot{Z}\left(n + \frac{N}{2}\right) = \dot{Z}(n)$$

С учетом этого при $n > N/2$ формула (7.6) представляется в виде

$$\dot{X}(n) = \dot{Y}\left(n - \frac{N}{2}\right) + e^{-j\frac{2\pi n}{N}} \cdot \dot{Z}\left(n - \frac{N}{2}\right) = \qquad (7.7)$$

$$= \dot{Y}\left(n - \frac{N}{2}\right) - e^{-j\frac{2\pi}{N}\left(n - \frac{N}{2}\right)} \cdot \dot{Z}\left(n - \frac{N}{2}\right)$$

Процесс вычисления 8-точечного ДПФ путем разбиения его на два 4-точечных ДПФ иллюстрируется на рис. 7.1. На этом рисунке использовано следующее обозначение для комплексных экспонент:

$$\dot{\omega}^n = \exp\left(-j\frac{2\pi n}{N}\right).$$

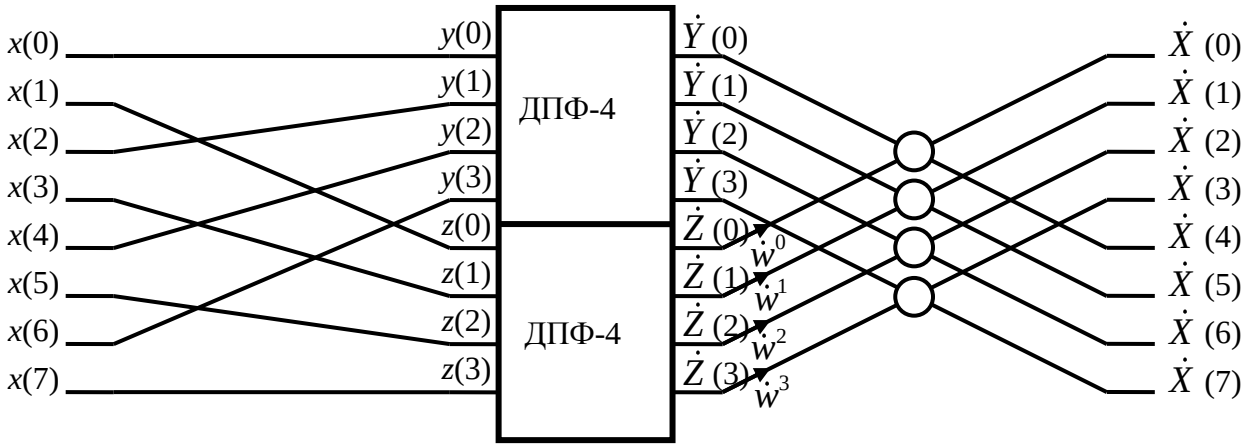


Рис. 7.1. Вычисление 8-точечного ДПФ с помощью двух 4-точечных ДПФ

Блоки, выполняющие на рис. 7.1 объединение результатов двух ДПФ, требуют дополнительных комментариев. Каждый такой блок имеет два входных и два выходных сигнала. Один из входных сигналов умножается на комплексную экспоненту $\dot{\omega}^k$, после чего суммируется со вторым входным сигналом и вычитается из него, формируя таким образом два выходных сигнала. Это соответствует реализации формул (7.6) и (7.7). Данная операция получила название «бабочки» (butterfly). Расшифровка ее структуры представлена на рис. 7.2.

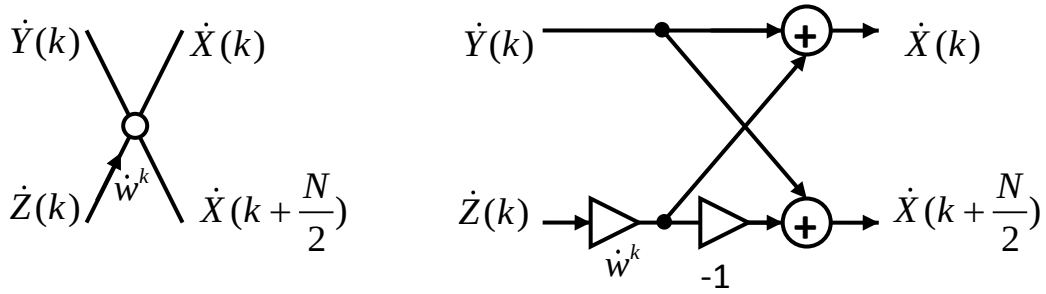


Рис. 7.2. Условное обозначение «бабочки» БПФ с прореживанием по времени (слева) и ее структурная схема (справа)

Оценим количество операций, необходимое для вычисления ДПФ указанным способом. Каждое из двух ДПФ половинной размерности требует $N^2/4$ операций. Кроме того, при вычислении окончательных результатов каждый спектральный коэффициент $\dot{Z}(n)$ умножается на экспоненциальный комплексный множитель. Это добавляет еще $N/2$ операций. Итого получается $2N^2/4 + N/2 = N(N+1)/2$, что почти вдвое меньше, чем при вычислении ДПФ прямым способом.

Если $N/2$ тоже является четным числом (то есть если N делится на 4), можно продолжить описанную процедуру, выразив результат через четыре ДПФ размерности $N/4$. Это позволяет еще больше сократить число требуемых вычислительных операций.

Делить исходную последовательность можно на любое количество частей. Таким образом, приведенный алгоритм позволяет уменьшить число операций в случае любого N, не являющегося простым числом. Степень ускорения вычислений зависит от числа фрагментов последовательности и является максимальной при делении на две части, как в рассмотренном примере.

Наибольшая степень ускорения вычислений может быть достигнута при $N = 2^k$, в этом случае деление последовательностей на две части можно продолжать до тех пор, пока не

получатся двухэлементные последовательности, ДПФ которых рассчитывается вообще без использования операций умножения (достаточно вычислить сумму и разность двух отсчетов). Число требуемых при этом пар операций «умножение — сложение» можно оценить как $N\log_2(N)$. Таким образом, вычислительные затраты по сравнению с непосредственным использованием формулы (7.3) уменьшаются в $N/\log_2(N)$ раз. При больших N это отношение становится весьма велико (например, $1024/\log_2(1024)=102,4$, то есть при N = 1024 достигается более чем 100-кратное ускорение).

БПФ с прореживанием по частоте

Формулы прямого и обратного ДПФ (7.3) и (7.4) отличаются только знаком в показателе экспоненты и множителем перед суммой. Поэтому можно получить еще один вариант алгоритма БПФ, выполнив преобразования, показанные на схеме рис. 7.1, в обратном порядке. Этот способ вычислений называется *прореживанием* по частоте (decimation in frequency, DIF). Покажем, как получить описание этого метода на основе формулы прямого ДПФ (7.3).

Разделим исходную последовательность {x(k)} на две следующие друг за другом половины (как и в предыдущем случае, N должно быть четным числом):

$$\dot{X}(n)=\sum_{m=0}^{N/2-1}x(m)e^{-j\frac{2\pi mn}{N}}+\sum_{m=0}^{N/2-1}x(m+\frac{N}{2})e^{-j\frac{2\pi(m+N/2)n}{N}}$$

Из второй суммы можно выделить множитель

$$e^{-j\frac{2\pi(N/2)n}{N}}=e^{-j\pi n}=(-1)^n.$$

Этот множитель равен 1 или -1 в зависимости от четности номера вычисляемого спектрального отсчета n, поэтому дальше рассматриваем четные и нечетные n по отдельности. После выделения множителя ±1 комплексные экспоненты в обеих суммах становятся одинаковыми, поэтому выносим их за скобки, объединяя две суммы:

$$\dot{X}(2k)=\sum_{m=0}^{N/2-1}\left(x(m)+x(m+\frac{N}{2})\right)e^{-j\frac{2\pi mk}{N/2}},$$

$$\dot{X}(2k+1)=\sum_{m=0}^{N/2-1}\left(x(m)-x(m+\frac{N}{2})\right)e^{-j\frac{2\pi mk}{N/2}}e^{-j\frac{2\pi m}{N/2}}$$

Фигурирующие здесь суммы представляют собой ДПФ суммы и разности половин исходной последовательности, при этом разность перед вычислением ДПФ умножается на комплексные экспоненты $\exp(-j2\pi m/N)$. Каждое из двух используемых здесь ДПФ имеет размерность $N/2$.

Итак, при прореживании по частоте вычисления организуются следующим образом:

1. Из исходной последовательности {x(k)} длиной N получаются две последовательности {y(m)} и {z(m)} длиной $N/2$ согласно следующим формулам:

$$y(m)=x(m)+x\left(m+\frac{N}{2}\right),$$

$$z(m)=\left(x(m)-x\left(m+\frac{N}{2}\right)\right)e^{-j\frac{2\pi m}{N}}.$$

2. ДПФ последовательности {y(m)} дает спектральные отсчеты с четными номерами, ДПФ последовательности {z(m)} — с нечетными:

$$\dot{X}(2k)=\sum_{m=0}^{N/2-1}y(m)e^{-j\frac{2\pi mk}{N/2}},$$

$$\dot{X}(2k+1)=\dot{Z}(k)=\sum_{m=0}^{N/2-1}z(m)e^{-j\frac{2\pi mk}{N/2}}.$$

Все сказанное в предыдущем разделе о возможности деления последовательности на иное, отличное от двух, число частей и об уменьшении числа операций, требуемых для расчетов, относится и к алгоритму с прореживанием по частоте.

Процесс вычислений 8-точечного ДПФ путем разбиения его на два 4-точечных ДПФ с прореживанием по частоте показан на рис. 7.3.

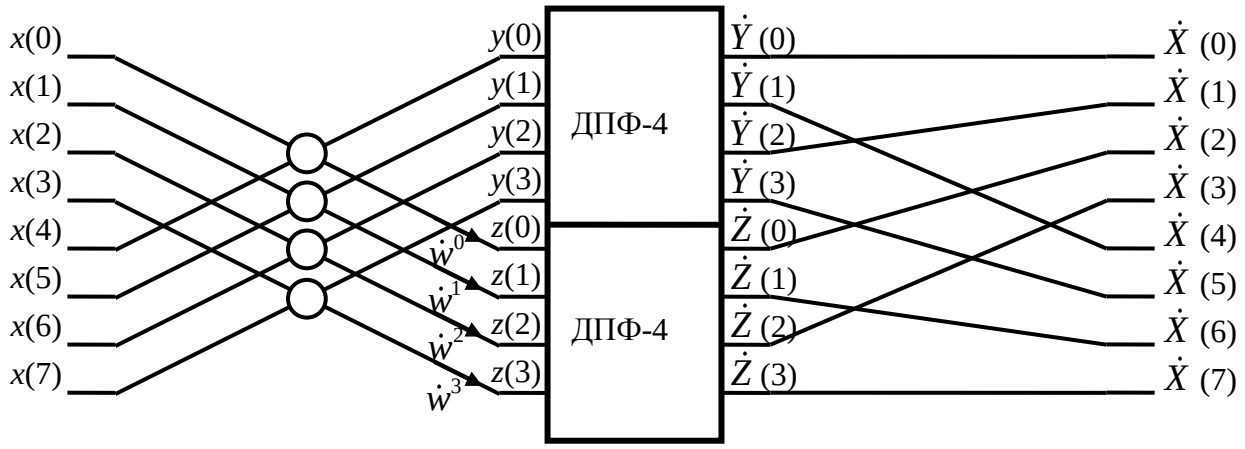


Рис. 7.3. Вычисление 8-точечного ДПФ с помощью двух 4-точечных ДПФ путем прореживания по частоте.

Поскольку комплексный экспоненциальный множитель в данном алгоритме применяется к *результату* вычитания двух сигналов, «бабочка» БПФ с прореживанием по частоте имеет несколько иную структурную схему (рис. 7.4).

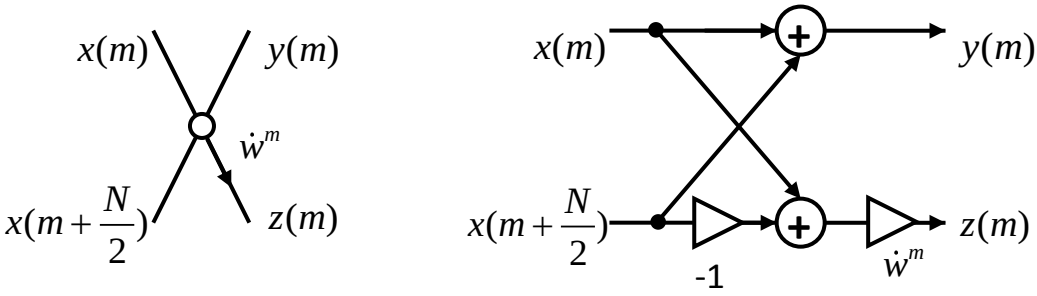


Рис. 7.4. Условное обозначение «бабочки» БПФ с прореживанием по частоте (слева) и ее структурная схема (справа)

Для получения алгоритма обратного БПФ достаточно поменять в приведенных формулах знак в показателях комплексных экспонент и добавить на выходе (или на входе) деление на два (в более общем случае — на используемый коэффициент прореживания).

Возможен также обобщенный подход к рассмотрению алгоритмов БПФ с прореживанием по времени и по частоте.

Основание алгоритма БПФ

В названиях алгоритмов БПФ можно встретить слово «RADIX» («основание» — в математическом смысле). Следующее после него число обозначает число фрагментов, на которое разбивается сигнал на каждом этапе прореживания (а также минимальный размер «кусочков» входного вектора, который достигается в результате его последовательных разбиений).

В алгоритмах «RADIX-2» размер анализируемой последовательности должен быть равен степени двойки, а ее половинное деление производится вплоть до получения двухэлементных последовательностей. Вычисление их ДПФ не требует операций умножения — два спектральных отсчета представляют собой сумму и разность отсчетов временных:

$$\dot{X}(0) = x(0) + x(1),$$

$$\dot{X}(1) = x(0) - x(1).$$

В алгоритмах «RADIX-4» количество отсчетов сигнала должно быть равно степени четверки, при каждом прореживании сигнал делится на четыре фрагмента, а последней стадией

деления являются четырехэлементные последовательности. При вычислении их ДПФ умножение производится только на ±j, а такое умножение сводится к взаимной перестановке вещественной и мнимой частей комплексного числа с изменением знака у одной из них:

$$\dot{X}(0) = x(0) + x(1) + x(2) + x(3),$$
$$\dot{X}(1) = x(0) - jx(1) - x(2) + jx(3),$$
$$\dot{X}(2) = x(0) - x(1) + x(2) - x(3),$$
$$\dot{X}(3) = x(0) + jx(1) - x(2) - jx(3).$$

Использование основания 4 позволяет ощутимо уменьшить число выполняемых умножений.

Выводы

Наибольшее ускорение вычислений благодаря алгоритму БПФ достигается при длине анализируемого вектора, равной степени двойки. При разложении длины вектора на иные множители ускорение также возможно, хотя и не столь значительное. Если длина вектора — простое число, вычисление спектра может быть выполнено только по прямой формуле ДПФ.

Завершая краткое рассмотрение идеи БПФ, необходимо отметить следующее:

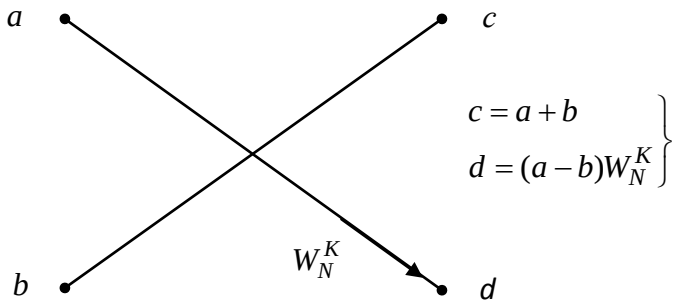
□ БПФ не является приближенным алгоритмом; при отсутствии вычислительных погрешностей он даст точно такой же результат, что и исходная формула ДПФ (7.3). Ускорение достигается исключительно за счет оптимальной организации вычислений;

□ применение БПФ имеет смысл, если число элементов в анализируемой последовательности является степенью числа 2. Как уже отмечалось, некоторое ускорение вычислений возможно и при разложении N на другие множители, однако это ускорение не столь велико, как при $N = 2^k$;

□ алгоритм БПФ предназначен для одновременного расчета всех спектральных отсчетов $\dot{X}(n)$. Если же необходимо получить эти отсчеты лишь для некоторых n, может оказаться предпочтительнее прямая формула ДПФ или алгоритм Герцеля.

БПФ с прореживанием по частоте

Этот алгоритм можно представить направленным графом, имеющим вид бабочки:



Примеры задач с решениями:

1. Вычислить БПФ последовательности $x(n)=\{0; 1; 2; 3; 4; 5; 6; 7\}$ с прореживанием по частоте с основанием 2.

Решение:

$W_8^0 = W_4^0 = W_2^0 = 1;$

$W_8^1 = e^{-j\frac{2\pi}{8}} = e^{-j\frac{\pi}{4}} = 0,7 - 0,7j;$

$W_8^2 = W_4^1 = e^{-j\frac{2\pi}{8}2} = e^{-j\frac{\pi}{2}} = -j;$

$W_8^3 = e^{-j\frac{2\pi}{8}3} = e^{-j\frac{3\pi}{4}} = -0,7 - 0,7j;$

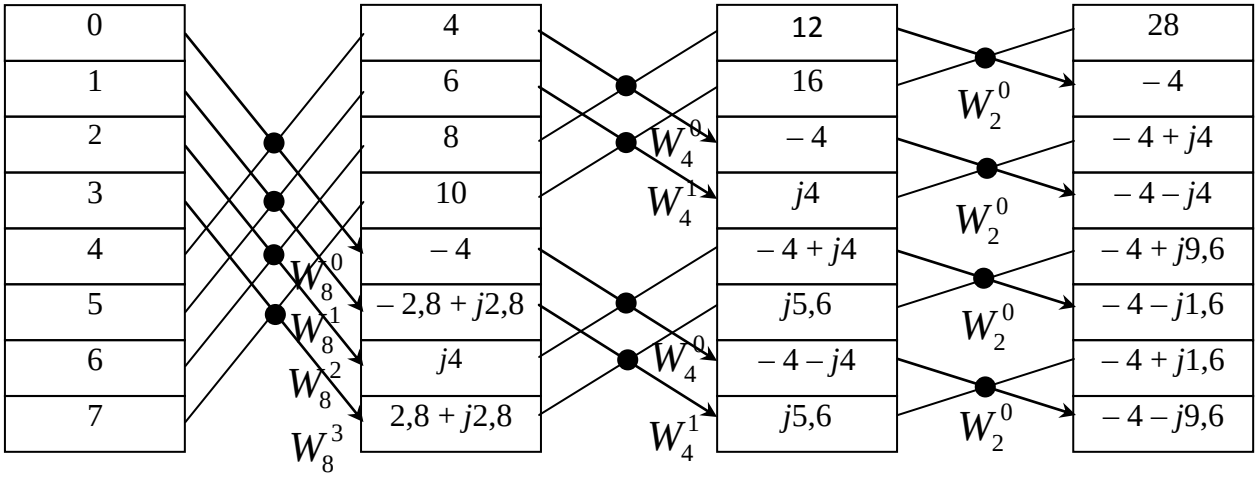
$(-4 + 4j) + (0,7 - 0,7j) \cdot (-4 + 4j) = -4 + 4j + (-2,8 + 2,8j + 2,8j + 2,8) =$

$= -4 + 9{,}6\,j;$

$(-4 + 4\,j) + (-0{,}7 - 0{,}7\,j) \cdot (-4 - 4\,j) = (-4 - 4\,j) + (2{,}8 + 2{,}8\,j + 2{,}8\,j - 2{,}8) =$

$= -4 - 4\,j + 5{,}6\,j = -4 + 1{,}6\,j$

$(-4 + 4\,j) - (-0{,}7 - 0{,}7\,j) \cdot (-4 - 4\,j) = -4 - 4\,j - 5{,}6\,j = -4 - 9{,}6\,j$

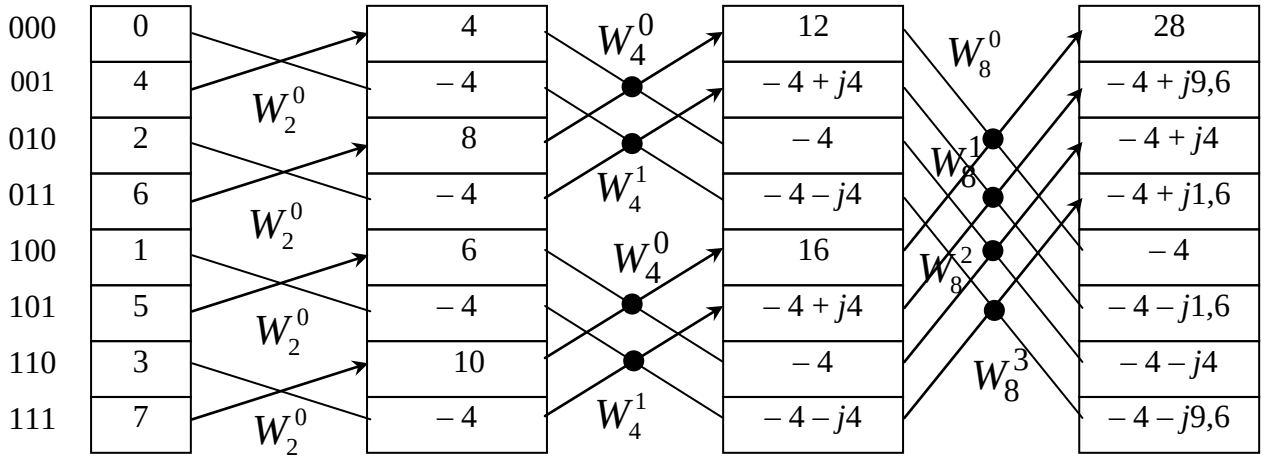


Пояснение к перестановкам

0	000	→	000	0
1	001	→	100	4
2	010	→	010	2
3	011	→	110	6
4	100	→	001	1
5	101	→	101	5
6	110	→	011	3
7	111	→	111	7

28	28
– 4	– 4 + j9,6
– 4 + j4	– 4 + j4
– 4 – j4	– 4 + j1,6
– 4 + j9,6	– 4
– 4 – j1,6	– 4 – j1,6
– 4 + j1,6	– 4 – j4
– 4 – j9,6	– 4 – j9,6

2. Вычислить БПФ последовательности $x(n)=\{0; 1; 2; 3; 4; 5; 6; 7\}$ с прореживанием по времени с основанием 2.



3. Составьте процедуру для сложения комплексных чисел. Исходные данные в регистрах eax и ebx, результат в регистре eax (младшие 16 разрядов – вещественная часть, старшие 16 разрядов – комплексная часть).

Ответ:

```
addcomplex:
add ax, bx; a1+a2
rol eax, 16
```

```
rol ebx, 16
add ax, bx; b1+b2
rol eax, 16
rol ebx, 16
ret
```

4. Составьте процедуру для умножения комплексных чисел. Исходные данные в регистрах eax и ecx, результат в регистре eax (младшие 16 разрядов – вещественная часть, старшие 16 разрядов – комплексная часть).

Ответ:

```
mulcomplex:
mov bp, ax
imul bp, cx; a1*a2
mov edx, eax
rol edx, 16
rol ecx, 16
imul dx, cx; b1*b2
sub bp, dx; a1*a2-b1*b2
imul ax, cx; a1*b2
mov dx, ax
rol eax, 16
rol ecx, 16
imul ax, cx; a2*b1
add ax, dx; a1*b2+a2*b1
rol eax, 16
rol ecx, 16
mov ax, bp
ret
```

5. Напишите программу реализующую вычисление БПФ последовательности длиной 4 с прореживанием по частоте с основанием 2. Разрядность исходных данных – 16 бит. Можно использовать процедуры для выполнения арифметических операций с комплексными числами из задач 1-3.

В качестве исходных данных можно использовать данные из примера 2: $x(n)=\{0; 1; 2; 3\}$, $X(k)=\{6; -2+j2; -2; -2-j2\}$.

Ответ:

```
section .data
x dw 0, 0, 1, 0, 2, 0, 3, 0; исходные данные комплексные
xfft1 dw 0, 0, 0, 0, 0, 0, 0, 0
xfft dw 0, 0, 0, 0, 0, 0, 0, 0

section .text
fft:
mov esi, x
mov edi, xfft1
mov ecx, 1+(0*65536);W4^0=1
call fftstep1
add esi, 4
add edi, 4
mov ecx, 0+(-1*65536);W4^1=-j
call fftstep1
mov esi, xfft1
mov edi, xfft
```

```
mov ecx, 1+(0*65536);W2^0=1
call fftstep2
add esi, 8
add edi, 8
mov ecx, 1+(0*65536);W2^0=1
call fftstep2
mov esi, xfft+4
call swapfft
ret

fftstep1:
mov eax, [esi]
mov ebx, [esi+8]
call addcomplex
mov [edi], eax
mov eax, [esi]
call subcomplex
call mulcomplex
mov [edi+8], eax
ret
fftstep2:
mov eax, [esi]
mov ebx, [esi+4]
call addcomplex
mov [edi], eax
mov eax, [esi]
call subcomplex
call mulcomplex
mov [edi+4], eax
ret
swapfft:
mov eax, [esi]
xchg [esi+4], eax
mov [esi], eax
ret
```

7. Образовательные технологии

При реализации ОПОП ВО подготовки кадров высшей квалификации при реализации различных видов учебной работы применяются информационные технологии (использование мультимедийного сопровождения лекций, электронных мультимедийных учебных материалов и др.) и интерактивные методы и технологии обучения (лекции-визуализации, тренинг), с учетом содержания дисциплины и видов занятий, предусмотренных учебным планом.

В частности, предусмотрено использование следующих образовательных технологий:

1. Классическая лекция, предусматривающая систематическое, последовательное, монологическое изложение учебного материала.
2. Лекция-визуализация – передача информации посредством схем, таблиц, рисунков, видеоматериалов, проводится по ключевым темам с комментариями.

При реализации настоящей рабочей программы предусматриваются интерактивные и активные формы проведения занятий, дискуссии по темам исследования и поставленным научным проблемам.

8. Методические указания по освоению дисциплины

1. Методические указания к практическим занятиям по дисциплинам «Цифровая обработка сигналов и сигнальные процессоры в инфокоммуникационных системах» и «Микропроцессорные устройства в инфокоммуникационных системах» [кафедральное издание] Сост. Городецкий И.И. – Уфа: УГАТУ, 2015.

При изучении дисциплины обучающийся должен посещать все лекции, а также практические занятия. В процессе изучения дисциплины аспирантам рекомендуется подготовка обзорной статьи, в которой будут изложены тенденции развития объекта исследования в диссертации студента. Для обучающегося также рекомендована разработка нового программного решения в области телекоммуникационных систем и сетей связи, исследуемым им в рамках научного исследования.

При самостоятельной подготовке обучающийся должен пользоваться не только учебно-методической литературой, но и периодической научной литературой. В частности журналами «Электросвязь», «Компоненты и технологии» и т.д., а также зарубежной литературой и периодическими изданиями.

9 . Материально-техническое обеспечение дисциплины

Аудиторный фонд кафедры телекоммуникационных систем включает как традиционные учебные аудитории, так и специализированные, обеспечивающие проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работы обучающихся, предусмотренных учебным планом.

Специализированные аудитории оснащены современной вычислительной, мультимедийной, проекционной и аудио-видео техникой.

Материально-техническая база для данной дисциплины обеспечивается наличием:

- лекционных аудиторий с современными средствами демонстрации: 6-401 а) и б), 6-406, 6-407, 6-517;

- кафедральной лаборатории, обеспечивающей реализацию ОПОП ВО: «Компьютерный класс» (6-401 б)).

Технические средства обучения:

Компьютерный класс, оснащенный персональными компьютерами, подключенными к сети Internet. Программное обеспечение включает в себя среду разработки программ на ассембере SimpleASM (свободное программное обеспечение); а также MS Windows XP, Microsoft Office.

10. Адаптация рабочей программы для лиц с ОВЗ

Адаптированная программа разрабатывается при наличии заявления со стороны обучающегося (родителей, законных представителей) и медицинских показаний (рекомендациями психолого-медико-педагогической комиссии). Для инвалидов адаптированная образовательная программа разрабатывается в соответствии с индивидуальной программой реабилитации.

ЛИСТ

согласования рабочей программы

Направление подготовки: 11.06.01 Электроника, радиотехника и системы связи
код и наименование

Направленность подготовки (программа): Системы, сети и устройства телекоммуникаций
наименование

Дисциплина: Микропроцессорные устройства в инфокоммуникационных системах

Учебный год 2015 / 2016

РЕКОМЕНДОВАНА заседанием кафедры телекоммуникационных систем
наименование кафедры

протокол № 11 от "29" 06 2015 г.
Заведующий кафедрой _____ Султанов А.Х.
подпись расшифровка подписи

Исполнитель:
кандидат технических наук,
доцент _____ И.И. Городецкий
должность подпись расшифровка подписи

СОГЛАСОВАНО:

Заведующий кафедрой
_____ Султанов А.Х. 30.06.15
наименование кафедры *личная подпись* *расшифровка подписи* *дата*

Председатель НМС по УГСН 110000 Электроника, радиотехника и системы связи
протокол № 2 от "30" 06 2015 г.

_____ Султанов А.Х.
личная подпись *расшифровка подписи*

Библиотека директор _____ Мустафина С.Ф. 28.08.15
личная подпись *расшифровка подписи* *дата*

Начальник отдела аспирантуры _____ Р.К. Фаттахов 28.08.15
личная подпись *расшифровка подписи* *дата*

Рабочая программа зарегистрирована в ООПМА и внесена в электронную базу данных
Начальник _____ И.А. Лакман 4.09.15
личная подпись *расшифровка подписи* *дата*