**Федеральное государственное автономное образовательное учреждение высшего образования**
**«Московский физико-технический институт (национальный исследовательский университет)»**

**УТВЕРЖДЕНО**
**Директор физтех-школы прикладной математики и информатики**
**А.М. Райгородский**

**Рабочая программа дисциплины (модуля)**

**по дисциплине:** Конструирование оптимизирующих компиляторов
**по направлению:** Прикладные математика и физика
**профиль подготовки:** Радиотехника и компьютерные технологии
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра системного программирования
**курс:** 3
**квалификация:** бакалавр

Семестры, формы промежуточной аттестации:
6 (весенний) - Дифференцированный зачет
7 (осенний) - Экзамен

Аудиторных часов: 105 всего, в том числе:
лекции: 60 час.
семинары: 0 час.
лабораторные занятия: 45 час.

Самостоятельная работа: 135 час.

Подготовка к экзамену: 30 час.

Всего часов: 270, всего зач. ед.: 6

Программу составил: С.С. Гайсарян, канд. физ.-мат. наук, доцент

Программа обсуждена на заседании кафедры системного программирования 15.05.2020

**Аннотация**

Дисциплина «Конструирование оптимизирующих компиляторов» изучает методы и алгоритмы оптимизации процедур и программ в целом (все процедуры и библиотеки, участвующие в сборке программы), а также применение указанных технологий для обнаружения уязвимостей и других дефектов программ как в исходном коде (на языке высокого уровня), так и в машинном коде современных компьютеров.

В результате изучения дисциплины студенты будут знать:

современные свободные компиляторные среды (GCC, LLVM и др.) и их использование для реализации программных инструментов, базирующихся на компиляторных технологиях;

алгоритмы машинно-независимой оптимизации (статической и динамической);

решения задач обратной инженерии, защиты программного кода, обнаружения дефектов в программах и др.

Студенты научатся:

разрабатывать, обосновывать и реализовывать новые методы и алгоритмы машинно-независимой оптимизации программ, задач обратной инженерии, защиты программного кода и обнаружения дефектов;

разрабатывать и реализовывать новые языки и их оптимизирующие компиляторы для новых архитектур процессоров.

## 1. Цели и задачи

**Цель дисциплины**

- изучение основ построения оптимизирующих статических и динамических компиляторов современных языков программирования, учитывающих особенности архитектур современных компьютеров.

**Задачи дисциплины**

- освоение студентами базовых знаний в области оптимизирующей компиляции программ;
- приобретение теоретических знаний в области теории графов, теории решеток, методов сбора статистики, используемых при разработке методов анализа и трансформации программ;
- оказание консультаций и помощи студентам в проведении собственных исследований и разработок в областях, использующих компиляторные технологии;
- приобретение навыков работы на современных неоднородных распределенных компьютерных системах.

## 2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

| Код и наименование компетенции | Индикаторы достижения компетенции |
|---|---|
| ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности | ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности |
| ПК-4 Способен критически оценивать применимость используемых методик и методов | ПК-4.2 Знает источники происхождения и умеет производить оценку погрешности измерений и достоверности экспериментальных результатов |

## 3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- фундаментальные понятия, теории современного системного программирования;
- структуру и состав современных оптимизирующих компиляторных сред (примеры – GCC, LLVM и др.);
- цели, задачи и методы машинно-независимой, машинно-ориентированной статической и динамической оптимизации программ в процессе их компиляции;
- принципы применения компиляторных сред для решения других задач программной инженерии: выявление дефектов и аудит программ, запутывание программ и др.

уметь:

- разрабатывать, обосновывать и реализовывать новые методы и алгоритмы машинно-независимой оптимизации программ;
- разрабатывать и реализовывать новые языки и их оптимизирующие компиляторы для новых архитектур процессоров, в том числе специализированных;
- применять компиляторные методы и компиляторные среды для решения задач обратной инженерии, защиты программного кода, обнаружения дефектов в программах и др.

владеть:

- навыками освоения большого объема информации;
- навыками самостоятельной работы в Интернете;
- культурой разработки и реализации системного программного обеспечения современных компьютеров;
- навыками грамотной разработки новых языков программирования и их программного обеспечения.

**4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий**

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

| № | Тема (раздел) дисциплины | Трудоемкость по видам учебных занятий, включая самостоятельную работу, час. | | | |
|---|---|---|---|---|---|
| | | Лекции | Семинары | Лаборат. работы | Самост. работа |
| 1 | Постановка задачи оптимизации программ в компиляторах. Локальная и глобальная оптимизации. Глобальная оптимизация. Анализ потока данных. | 10 | | 10 | 25 |
| 2 | Методы ускорения анализа потока данных. Выделение областей графа потока управления. | 10 | | 10 | 25 |
| 3 | Межпроцедурный анализ указателей. Другие оптимизирующие преобразования. Применение статического анализа потоков данных в задачах инженерии программ. | 10 | | 10 | 25 |
| 4 | Особенности архитектуры современных компьютеров и задача генерации оптимального кода. Выдача команд. | 10 | | 5 | 20 |
| 5 | Распределение и назначение регистров. Планирование кода. | 10 | | 5 | 20 |
| 6 | Параллельное выполнение циклов. Динамическая и адаптивная оптимизация в компиляторах времени выполнения. | 10 | | 5 | 20 |
| Итого часов | | 60 | | 45 | 135 |
| Подготовка к экзамену | | 30 час. | | | |

| | |
|---|---|
| Общая трудоёмкость | 270 час., 6 зач.ед. |

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 6 (Весенний)

1. Постановка задачи оптимизации программ в компиляторах. Локальная и глобальная оптимизации. Глобальная оптимизация. Анализ потока данных.

Задача компиляции. Неоптимизирующий компилятор. Основные фазы компиляции. Выявление ошибок в процессе компиляции и сообщения о них. Промежуточное представление программы – трехадресный код (четверки). Граф потока управления и алгоритм его построения. Локальная и глобальная оптимизации. Метод нумерации значений (локальный) как основа локальной оптимизации. Недостаточность локальной оптимизации.

Состояние программы. Путь выполнения (трасса). Прямой и обратный обход программы. Передаточная функция инструкции. Композиция передаточных функций. Передаточная функция базового блока. Определение переменной. Постановка задачи о достигающих определениях. Понятие консервативности анализа. Передаточные функции задачи о достигающих определениях (передаточные функции класса gen-kill). Замкнутость класса передаточных функций gen-kill относительно композиции. Система уравнений для задачи о достигающих определениях и ее решение методом итераций. Итеративный алгоритм для вычисления достигающих определений.

Применение достигающих определений: нахождение выражений, инвариантных относительно циклов. Алгоритм обнаружения кода, инвариантного относительно цикла. Анализ живых переменных: передаточные функции, система уравнений и итерационный алгоритм. Анализ доступных выражений: передаточные функции, система уравнений и итерационный алгоритм.

Анализ потока данных. Обоснование итерационных алгоритмов. Понятие полурешетки. Связь между операцией сбора и полурешеточным отношением порядка. Понятие верхнего (наибольшего) элемента полурешетки. Реализация полурешеток с помощью конечных множеств. Операции объединения и пересечения множеств как реализации операции сбора. Диаграммы полурешеток. Декартовы произведения полурешеток. Понятие структуры потока данных. Понятие замкнутости семейства передаточных функций. Замкнутость семейств передаточных функций для достигающих определений, живых переменных и доступных выражений.

Монотонные и дистрибутивные структуры потока данных. Дистрибутивность структур потока данных для достигающих определений, живых переменных и доступных выражений. Обобщенный итеративный алгоритм и его свойства. Понятие максимальной фиксированной точки. Сходимость обобщенного итеративного алгоритма к максимальной фиксированной точке. Идеальное решение уравнений потока данных. Решение сбором по всем путям для дистрибутивных и монотонных структур потока данных. Консервативность максимальной фиксированной точки.

Пример недистрибутивной, но монотонной структуры потока данных – распространение констант. Полурешетка для проблемы распространение констант. Семейство передаточных функций, его монотонность и недистрибутивность. Итерационный алгоритм распространения констант.

Исключение частично избыточных выражений методом анализа ожидаемых выражений. Четырехэтапный алгоритм отложенного перемещения кода. Достоинства и недостатки подхода. Предварительный этап – ликвидация критических ребер. Первый этап – анализ ожидаемых выражений. Второй этап – анализ доступных выражений. Третий этап – анализ откладываемых выражений. Четвертый этап – анализ используемых выражений (исключение мертвого кода).

2. Методы ускорения анализа потока данных. Выделение областей графа потока управления.

Структурный анализ графа потока управления. Глубинное остовное дерево и его обход. Нумерация узлов графа потока управления. Классификация ребер графа потока управления. Алгоритм построения глубинного остовного дерева и упорядочения графа потока управления в глубину. Нумерация узлов графа потока управления (в глубину). Доминаторы. Свойства отношения доминирования. Итеративный алгоритм вычисления доминаторов. Дерево доминаторов и алгоритм его построения. Классификация ребер графа потока управления.

Понятие естественного цикла. Алгоритм построения естественного цикла для заданного обратного ребра. Вложенность естественных циклов. Гнезда циклов. Сильно связанные компоненты. Алгоритм построения всех максимальных сильно связанных компонентов заданного графа потока управления. Приводимые графы потока управления. Неприводимые области (собственные и несобственные) графа потока управления. Примеры неприводимых областей. Глубина графа потока управления.

Понятие области. Виды областей Алгоритм построения иерархии областей для приводимых графов потока управления. Дерево управления. Алгоритм построения восходящего порядка областей графа потока управления. Другие способы структурирования графов потока управления: интервальный анализ, «структурный анализ» с помощью шаблонов и др.

Анализ потоков данных на основе областей. Построение передаточных функций областей с помощью операций композиции, сбора и замыкания. Замкнутость структуры потока данных относительно операций композиции, сбора и замыкания. Трехэтапный алгоритм анализа потока данных на основе областей (на примере достигающих определений). Метод расщепления узлов для обработки неприводимых графов потока управления.

Форма статического единственного присваивания (SSA-форма). Определение SSA-формы. Понятие □-функции. Свойства □-функции. Базовый алгоритм преобразования промежуточного представления программы в SSA-форму. Недостатки базового алгоритма. Пример применения алгоритма.

Форма статического единственного присваивания (SSA-форма). Квазиоптимальная SSA-форма. Граница доминирования. Алгоритм построения границ доминирования. Построение множества глобальных имен и других вспомогательных множеств. Размещение f-функций. Переименование переменных. Восстановление кода из SSA-формы. Проблема потери копий

Глобальная нумерация значений. Два подхода к реализации глобальной нумерации значений: нумерация значений, основанная на хэшировании и нумерация значений, основанная на классификации значений. Нумерация значений, основанная на хэшировании. Нумерация значений в расширенном базовом блоке. Повторное использование результатов для блоков, входящих в несколько путей. Механизм контекстно-ориентированных хэш-таблиц. Алгоритм нумерации значений на основе доминаторов.

Нумерация значений, основанная на классификации значений. Понятие конгруэнтности ориентированных графов.

Объединение нумерации значений с построением SSA-формы.

3. Межпроцедурный анализ указателей. Другие оптимизирующие преобразования. Применение статического анализа потоков данных в задачах инженерии программ.

Внутрипроцедурный (глобальный). Проблемы, связанные с обработкой членов структур, элементов массивов и данных, доступных по указателям, в том числе – динамических данных. Понятие алиаса. Алиасы в языке Си. Алиасы в языке Java. Глобальный (внутрипроцедурный) анализ алиасов: первая фаза – обнаружение алиасов, вторая фаза – распространение алиасов (задача анализа потока данных). Недостаточность глобального анализа алиасов.

Межпроцедурный анализ алиасов. Способы задания графа вызовов. Чувствительность к потоку и контексту вызова. Контекстно-нечувствительный межпроцедурный анализ. Контекстно-чувствительный анализ на основе аннотаций. Контекстно-чувствительный анализ на основе классификации и клонирования. Контекстно-чувствительный анализ ссылок

Оптимизация циклов: классификация и обработка индуктивных переменных, развертка циклов, исключение ненужных (избыточных) проверок условий окончания цикла. Оптимизация потока управления. Оптимизация возвратов из рекурсивных процедур. Открытое вставление процедур. Порядок применения оптимизирующих преобразований. Режимы компиляции.

Распознавание программ: восстановление документации разработчика по исходному коду программы. Запутывание (обфускация) программ на языках высокого уровня. Нахождение критических ошибок и уязвимостей.

Семестр: 7 (Осенний)

4. Особенности архитектуры современных компьютеров и задача генерации оптимального кода. Выдача команд.

Особенности архитектуры современных компьютеров, учитываемые при генерации объектного кода (обзор). RISC и CISC. Конвейер потока команд: блокировки конвейера, система сброса конвейера, конфликты по данным. Векторные регистры и векторные команды. Вырезки из массивов. Конвейер данных. Диспетчер и выдача команд. Блок предсказания переходов. Конвейерное выполнение. Out-of-Order-процессоры. VLIW и EPIC. Кэш, локальность, упреждающая выборка (prefetching). Проблемы генерации оптимизированного кода.

Промежуточное представление низкого уровня (последовательность трехадресных инструкций). Операции низкого уровня. Модель целевой машины (целевой язык). Набор команд. Псевдорегистры. Режимы адресации: прямая, косвенная, индексированная адресации. Стоимость команд и стоимость программы. Задачи генератора кода: распределение памяти, выбор команд, распределение регистров, выбор оптимального порядка команд (планирование кода).

Распределение памяти: статическое и динамическое. Статическое выделение памяти. Дескрипторы регистров и адресов.

Генерация кода для базового блока. Генерация кода для вызовов процедур и возвратов из них (соглашения о связях).

Выбор команд – построение отображения программы в промежуточном представлении на последовательность команд целевой машины.

Выбор команд путем переписывания дерева. Действия, шаблоны, схема трансляции дерева. Поиск соответствий с помощью синтаксического анализа. Числа Ершова. Алгоритм генерации кода для размеченного дерева выражения. Вычисление выражений при недостаточном количестве регистров. Замощение дерева. Генерация кода с использованием динамического программирования.

Покадровая оптимизация: устранение лишних инструкций; оптимизация потока управления; алгебраические упрощения; использование машинных идиом. Исключение недостижимого кода.

5. Распределение и назначение регистров. Планирование кода.

Распределение и назначение регистров в пределах базового блока (Функция getReg ()). Глобальное распределение регистров. Интервалы жизни значений переменных. Построение интервалов жизни. Оценка стоимости сброса. Конфликтные ситуации и граф конфликтов. Построение графа конфликтов. Раскраска графа конфликтов сверху вниз и снизу вверх.

Распределение и назначение регистров. Алгоритм раскраски графа конфликтов снизу вверх. Структура распределителя регистров. Примеры глобального распределения регистров

Анализ зависимостей по данным. Зависимости по управлению. Граф зависимостей. Планирование списков базовых блоков. Опережающее (спекулятивное) выполнение. Использование предикатные команд.

Глобальное планирование кода. Эквивалентность по управлению. Перемещение кода вверх по пути управления. Перемещение кода вниз по пути управления.

Глобальное планирование кода. Алгоритм глобального планирования кода на основе областей. Развертка циклов. Уплотнение окрестностей. Агрессивные алгоритмы перемещения кода. Конвейеризация циклов. Модель процессора. Последовательное и параллельное выполнение итераций цикла. Частичная развертка цикла. Циклы с зависимыми итерациями. Ограничения программной конвейеризации. Ограничения, связанные с зависимостями по данным

Алгоритм программной конвейеризации. Планирование ациклических графов зависимости данных. Планирование графов с циклическими зависимостями.

6. Параллельное выполнение циклов. Динамическая и адаптивная оптимизация в компиляторах времени выполнения.

Симметричные мультипроцессоры с общей памятью (SMP). Многоядерные процессоры. Пример параллельно выполняемого цикла. Закон Амдаля. Понятие локальности данных: пространственная и временная локальность. Формальная постановка задачи распараллеливания циклов. Демонстрация некоторых приемов распараллеливания на примере параллельной программы умножения матриц.

Распараллеливание циклов. Пространство итераций. Построение пространств итераций для гнезд циклов. Управление порядком выполнения циклов гнезда. Алгоритм исключения Фурье-Моцкина. Алгоритм вычисления границ циклов для заданного порядка выполнения. Повторное использование данных. Собственные и групповые повторные использования. Анализ зависимостей по данным. Обнаружение параллельности, не требующей синхронизации. Ограничения разбиений пространства и их решение.

Структура JIT-компилятора. Примеры JIT-компиляторов. Уровни оптимизации. Профилирование в JIT-компиляторах. Хранение данных о профилях. Выборочная оптимизация.

Динамическая и адаптивная оптимизация в JIT-компиляторах. Профилирование с помощью «семплов». Архитектура JIT-компилятора Jikes RVM: подсистема измерений, подсистема перекомпиляции, подсистема управления процессом динамической компиляции (контроллер). Фазы JIT-компиляции. Выбор уровня перекомпиляции. Адаптивная оптимизация. Использование результатов предыдущего выполнения.

**5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)**

Необходимое оборудование для лекций: компьютер и мультимедийное оборудование (проектор, звуковая система).

**6.Перечень рекомендуемой литературы**

Основная литература

1. Компиляторы: принципы, технологии и инструментарий [Текст] : [учеб. пособие для вузов] / Ахо, А. В. [и др.] ; [пер. с англ. и ред. И. В. Красикова] .— 2-е изд. — М. : Вильямс, 2011 .— 1184 с.

Дополнительная литература

**7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)**

Не используются

**8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)**

Программное обеспечение и информационные технологии не требуются.

**9. Методические указания для обучающихся по освоению дисциплины (модуля)**

Студент, изучающий дисциплину, должен с одной стороны, овладеть общим понятийным аппаратом, а с другой стороны, должен научиться применять теоретические знания на практике.

В результате изучения дисциплины студент должен знать основные определения, понятия, аксиомы, алгоритмы.

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя:

– чтение и конспектирование рекомендованной литературы,

– проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств;

– подготовку к дифференциальному зачету и экзамену.

Руководство и контроль за самостоятельной работой студента осуществляется в форме индивидуальных консультаций.

Важно добиться понимания изучаемого материала, а не механического его запоминания. При затруднении изучения отдельных тем, вопросов, следует обращаться за консультациями к лектору.

**ПРИЛОЖЕНИЕ**

# ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

**по направлению:** Прикладные математика и физика
**профиль подготовки:** Радиотехника и компьютерные технологии
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра системного программирования
**курс:** 3
**квалификация:** бакалавр

Семестры, формы промежуточной аттестации:
6 (весенний) - Дифференцированный зачет
7 (осенний) - Экзамен

**Разработчик:** С.С. Гайсарян, канд. физ.-мат. наук, доцент

**1. Компетенции, формируемые в процессе изучения дисциплины**

| Код и наименование компетенции | Индикаторы достижения компетенции |
|---|---|
| ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности | ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности |
| ПК-4 Способен критически оценивать применимость используемых методик и методов | ПК-4.2 Знает источники происхождения и умеет производить оценку погрешности измерений и достоверности экспериментальных результатов |

**2. Показатели оценивания компетенций**

В результате изучения дисциплины «Конструирование оптимизирующих компиляторов» обучающийся должен:

**знать:**

- фундаментальные понятия, теории современного системного программирования;
- структуру и состав современных оптимизирующих компиляторных сред (примеры – GCC, LLVM и др.);
- цели, задачи и методы машинно-независимой, машинно-ориентированной статической и динамической оптимизации программ в процессе их компиляции;
- принципы применения компиляторных сред для решения других задач программной инженерии: выявление дефектов и аудит программ, запутывание программ и др.

**уметь:**

- разрабатывать, обосновывать и реализовывать новые методы и алгоритмы машинно-независимой оптимизации программ;
- разрабатывать и реализовывать новые языки и их оптимизирующие компиляторы для новых архитектур процессоров, в том числе специализированных;
- применять компиляторные методы и компиляторные среды для решения задач обратной инженерии, защиты программного кода, обнаружения дефектов в программах и др.

**владеть:**

- навыками освоения большого объема информации;
- навыками самостоятельной работы в Интернете;
- культурой разработки и реализации системного программного обеспечения современных компьютеров;
- навыками грамотной разработки новых языков программирования и их программного обеспечения.

**3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю**

С целью контроля освоения обучающимися учебного материала проводится устный опрос в начале занятия по теме прошлой лекции или в конце занятия по пройденной теме.

**4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся**

Перечень контрольных вопросов для дифференцированного зачета в конце 6 семестра:

1. Промежуточное представление программы. Граф потока управления и алгоритм его построения. Локальная оптимизация. Представление базового блока в виде ориентированного ациклического графа. Локальный метод нумерации значений. Виды локальной оптимизации.
2. Понятие потока данных. Состояние программы. Передаточная функция инструкции. Передаточная функция базового блока. Пути выполнения.

3. Определение и использование переменной в программе. Понятие дости-гающего определения. Передаточные функции достигающих определений. Итеративный алгоритм вычисления достигающих определений.

4. Применение достигающих определений. Обнаружение кода, инвариант-ного относительно цикла и вынесение его за пределы цикла.

5. Понятие живой (активной) переменной. Итеративный алгоритм анализа живых переменных. Понятие доступного выражения. Итеративный алго-ритм вычисления доступных выражений.

6. Полурешетки. Основные свойства полурешеток. Примеры полурешеток. Наибольшая нижняя граница и ее связь с операцией сбора. Диаграммы полурешеток.

7. Структура потока данных. Замкнутость семейства передаточных функций для достигающих определений, живых переменных и доступных выраже-ний. Монотонные и дистрибутивные структуры. Дистрибутивность структур достигающих определений, живых переменных и доступных выражений.

8. Обобщенный итеративный алгоритм. Сходимость итеративного алгоритма к решению уравнений потоков данных. Максимальная фиксированная точка. Сравнение максимальной фиксированной точки с идеальным решением уравнений потока данных и решением сбором по всем путям.

9. Распространение констант как задача потока данных. Передаточные функции структуры распространения констант, ее монотонность и недист-рибутивность. Итеративный алгоритм распространения констант.

10. Обход графа потока управления. Глубинное остовное дерево. Алгоритм упорядочения графа потока в глубину. Классификация ребер графа потока управления. Доминаторы. Свойства отношения доминирования. Алгоритм вычисления доминаторов. Дерево доминаторов.

11. Обратные ребра и естественные циклы. Построение естественного цикла обратного ребра.

12. Структурный анализ графа потока управления. Понятие области. Выделение областей в графе потока. Виды областей. Построение иерархии областей для приводимых графов потока. Дерево управления. Алгоритм построения восходящего порядка областей графа потока.

13. Алгоритм анализа на основе областей: построение иерархии областей (снизу вверх) и обработка иерархии областей (сверху вниз). Пересчет передаточных функций. Пример применения алгоритм анализа достига-ющих на основе областей. Обработка неприводимых графов потоков.

14. Форма статического единственного присваивания (SSA-форма). Функции объединения значений (f-функции). Определение f-функции. Свойства f-функций. Базовый алгоритм построения SSA-формы.

15. Алгоритм построения квазиоптимальной SSA-формы. Алгоритм постро-ения границы доминирования. Алгоритм переименования переменных.

16. Алгоритм восстановления программы по ее квазиоптимальной SSA-форме.

17. Глобальная нумерация значений (постановка задачи). Два подхода к реализации глобальной нумерации значений. Первый подход: использо-вание хэш-функций.

18. Глобальная нумерация значений: конгруэнтные подграфы.

19. Анализ алиасов: определение алиасов, виды алиасов. Глобальный (внутрипроцедурный) анализ алиасов.

20. Недостаточность глобального анализа алиасов. Межпроцедурный анализ алиасов. Контекстно-нечувствительный межпроцедурный анализ и его недостатки. Построение графа вызовов.

21. Межпроцедурный анализ алиасов. Чувствительность к контексту. Строки вызовов. k-ограниченный контекстно-чувствительный анализ.

22. Контекстно-чувствительный анализ на основе клонирования и на основе аннотаций.

23. Симметричные мультипроцессоры с общей памятью (SMP). Многоядерные процессоры. Закон Амдаля. Понятие локальности данных: пространственная и временная локальность. Формальная постановка задачи распараллеливания циклов.

24. Распараллеливание циклов. Пространство итераций. Построение пространств итераций для гнезд циклов. Управление порядком выполнения циклов гнезда. Алгоритм исключения Фурье-Моцкина. Алгоритм вычисления границ циклов для заданного порядка выполнения.

25. Повторное использование данных. Собственные и групповые повторные использования. Анализ зависимостей по данным. Обнаружение параллельности, не требующей синхронизации. Ограничения разбиений пространства и их решение.

Перечень контрольных вопросов для экзамена в конце 7 семестра:

1. Генерация кода для базового блока
2. Распределение и назначение регистров
3. Глобальное распределение и назначение регистров. Построение интервалов жизни
4. Глобальное распределение и назначение регистров. Конфликтные ситуации и граф конфликтов. Построение графа конфликтов
5. Глобальное распределение и назначение регистров. Раскраска графа конфликтов сверху вниз. Раскраска графа конфликтов снизу вверх.
6. Глобальное распределение и назначение регистров. Структура распределителя регистров. Алгоритм в целом (на примере)
7. Выбор команд путем переписывания дерева.
8. Генерация оптимального кода для выражений. Числа Ершова.
9. Рекурсивный алгоритм генерации кода для размеченного дерева выражения
10. Генерация кода с использованием динамического программирования
11. Покадровая оптимизация
12. Планирование кода. Анализ зависимостей.
13. Планирование списков базовых блоков
14. Спекулятивное планирование. Опережающее выполнение.
15. Глобальное планирование кода. Алгоритм глобального планирования на основе областей
16. Глобальное планирование кода. Агрессивные алгоритмы перемещения кода
17. Программная конвейеризация циклов.
18. JIT-компиляторы. Профилирование. Сэмплирование. Выбор «горячих» участков
19. JIT-компиляторы. Архитектура JIT-компилятора
20. Параллельное выполнение циклов. Закон Амдаля. Понятие локальности данных
21. Формальная постановка задачи распараллеливания циклов.
22. Распараллеливание циклов. Блокирование: Разбиение на блоки
23. Изменение порядка циклов.
24. Обнаружение параллельности, не требующей синхронизации (на примере)

Примерный перечень билетов:

Билет №1.

1. Генерация кода с использованием динамического программирования
2. Планирование кода. Анализ зависимостей.

Билет №2.

1. Покадровая оптимизация
2. Планирование списков базовых блоков

Критерии оценивания

Оценка отлично 10 балов - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины, проявляющему интерес к данной предметной области, продемонстрировавшему умение уверенно и творчески применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка отлично 9 балов - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка отлично 8 балов - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, правильное обоснование принятых решений, с некоторыми недочетами.

Оценка хорошо 7 балов - выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но недостаточно грамотно обосновывает полученные результаты.

Оценка хорошо 6 балов - выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности.

Оценка хорошо 5 балов - выставляется студенту, если он в основном знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач достаточно большое количество неточностей.

Оценка удовлетворительно 4 бала - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он освоил основные разделы учебной программы, необходимые для дальнейшего обучения, и может применять полученные знания по образцу в стандартной ситуации.

Оценка удовлетворительно 3 бала - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, допускающему ошибки в формулировках базовых понятий, нарушения логической последовательности в изложении программного материала, слабо владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и с трудом применяет полученные знания даже в стандартной ситуации.

Оценка неудовлетворительно 2 бала - выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных принципов и не умеет использовать полученные знания при решении типовых задач.

Оценка неудовлетворительно 1 бал - выставляется студенту, который не знает основного содержания учебной программы дисциплины, допускает грубейшие ошибки в формулировках базовых понятий дисциплины и вообще не имеет навыков решения типовых практических задач.

## 5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Во время проведения дифференцированного зачета и экзамена обучающиеся могут пользоваться программой дисциплины, а также справочной литературой, вычислительной техникой, конспектами лекций.

Дифференцированный зачет может проводиться по итогам текущей успеваемости и сдачи заданий, или путем организации специального опроса, проводимого в устной форме. Экзамен проводится путем организации специального опроса, проводимого в устной форме.