Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"

Факультет компьютерных наук
Департамент программной инженерии

Рабочая программа дисциплины
Разработка компиляторов

для образовательной программы «Программная инженерия»
направления подготовки 09.03.04 «Программная инженерия»
уровень - бакалавр

Разработчик программы:
доцент, к.ф.-м.н. Гайсарян С.С. sgaisaryan@hse.ru

Одобрена на заседании департамента программной инженерии «___»_____ 2017 г.
Руководитель департамента Авдошин С.М. _____

Утверждена Академическим советом образовательной программы
«___»_____ 2017 г., № протокола_____

Академический руководитель образовательной программы
Шилов В.В. _____

Москва, 2017

1. Область применения и нормативные ссылки

Настоящая программа учебной дисциплины устанавливает минимальные требования к знаниям и умениям студента и определяет содержание и виды учебных занятий и отчетности.

Программа предназначена для преподавателей, ведущих данную дисциплину, учебных ассистентов и студентов образовательной программы «Программная инженерия» направления подготовки 09.03.04 «Программная инженерия», изучающих дисциплину «Разработка компиляторов»

Программа разработана в соответствии с образовательным стандартом Национального исследовательского университета «Высшая школа экономики» по направлению 09.03.04 «Программная инженерия».

2. Цели освоения дисциплины

Цель курса – Целью курса является изучение основ построения оптимизирующих статических и динамических компиляторов современных языков программирования, учитывающих особенности архитектур современных компьютеров; ознакомление студентов с предметом и основными подходами к формальной спецификации и верификации программ.

Задачами данного курса являются:

- 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;
- освоение студентами базовых современных достижений формальных методов в разработке программного обеспечения для критических по безопасности систем;
- формирование практических навыков применения формальных методов при проектировании и разработке программного обеспечения.

3. Компетенции обучающегося, формируемые в результате освоения дисциплины

В результате освоения дисциплины студент должен:

1. 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. ;
- основные виды формальных спецификаций, виды моделей программных систем, формализация требований к программному обеспечению;
- основные подходы к анализу свойств программ;
- методы верификации и тестирования ПО на основе формальных моделей;
- методы автоматизированного доказательства теорем;

2. 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.;
- формализовывать требования к программному обеспечению в виде алгебраических, имплицитных и эксплицитных спецификаций;
- осуществлять построение моделей программ, аналитическую верификацию последовательных программ;
- осуществлять формальное доказательство свойств программ в инструментах автоматизированного доказательства теорем;

3. Иметь навыки (приобрести опыт):

- освоения большого объема информации;
- самостоятельной работы в Интернете;
- культурой разработки и реализации системного программного обеспечения современных компьютеров;
- грамотной разработки новых языков программирования и их программного обеспечения;
- использования формальных методов для доказательства корректности программ;

В результате освоения дисциплины студент должен обладать следующими компетенциями:

Универсальные компетенции:

1 Способен решать проблемы в профессиональной деятельности на основе анализа и синтеза (УК-3)
2 Способен оценивать потребность в ресурсах и планировать их использование при решении задач в профессиональной деятельности (УК-4)
3 Способен работать с информацией: находить, оценивать и использовать информацию из различных источников, необходимую для решения научных и профессиональных задач (в том числе на основе системного подхода) (УК-5)
4 Способен вести исследовательскую деятельность, включая анализ проблем, постановку целей и задач, выделение объекта и предмета исследования, выбор способа и методов исследования, а также оценку его качества (УК-6)

Профессиональные компетенции:

5 Способен использовать методы и инструментальные средства исследования объектов профессиональной деятельности (ПК-3)
6 Способен обосновать принимаемые проектные решения, осуществлять постановку и выполнение экспериментов по проверке их корректности и эффективности (ПК-4)
7 Способен проектировать, конструировать и тестировать программные продукты (ПК-10)
8 Способен читать, понимать и выделять главную идею прочитанного исходного кода, документации (ПК-11)
9 Способен использовать различные технологии разработки программного обеспечения (ПК-16)
10 Способен применять основные методы и инструменты разработки программного обеспечения (ПК-17)

4. Место дисциплины в структуре образовательной программы

Изучение данной дисциплины базируется на знаниях, полученных студентами при освоении учебных дисциплин:

- «Дискретная математика»,
- «Программирование»,
- «Алгоритмы и структуры данных»,
- «Архитектура вычислительных систем»,
- «Операционные системы».

5. Тематический план учебной дисциплины

№	Название раздела	Всего часов	Аудиторные часы			Самостоятельная работа
			Лекции	Семинары	Практические занятия	
1	Постановка задачи анализа программ в компиляторах.	36	2		4	30
2	Глобальная оптимизация: анализ потока данных.	36	2		4	30
3	Методы ускорения анализа потока данных. Выделение областей графа потока управления	42	6		6	30
4	Методы ускорения анализа потока данных.	52	6		6	40
5	Межпроцедурный анализ указателей	52	6		6	40
6	Другие оптимизирующие преобразования. Применение статического анализа потоков данных в задачах инженерии программ.	48	2		4	42
		266	**24**		**30**	**212**

6. Формы контроля знаний студентов

Тип контроля	Форма контроля	3 год	
		1 модуль	2 модуль
Текущий	Домашнее задание	*	*
Итоговый	Экзамен		*

7. Критерии оценки знаний, навыков

Оценки по всем формам текущего контроля выставляются по 10-ти балльной шкале.

8. Порядок формирования оценок по дисциплине

Оценка по курсу состоит из оценки за выполнение домашних заданий $O_{прак.}$ (10 баллов), и оценки за итоговый устный экзамен (10 баллов). В диплом выставляет результирующая оценка по учебной дисциплине, которая формируется по следующей формуле:

$$O_{результ} = 0{,}5*O_{прак.} + 0{,}5*O_{экз}$$

9. Содержание дисциплины

Разделы и темы лекционных занятий	Содержание
1. Постановка задачи анализа программ в компиляторах.	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.
2. Глобальная оптимизация. Анализ потока данных.	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. Дистри-

	бутивность 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).
3. 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.
4. Методы ускорения анализа по-	Форма статического единственного присваивания (SSA-форма). Определение SSA-формы. Понятие ϕ-функции. Свойства ϕ-

тока данных.	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 □-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.
5. Межпроцедурный анализ указателей	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.
6. Другие оптимизирующие преобразования. Применение статического анализа потоков данных в задачах инженерии программ.	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.

10.Оценочные средства для текущего контроля и аттестации студента

Перечень контрольных вопросов для экзамена

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 (ϕ-функции). Определение ϕ-функции. Свойства ϕ-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.

11. Учебно-методическое и информационное обеспечение дисциплины

a. Базовый учебник

1. А.В. Ахо, М.С. Лам, Р. Сети, Дж.Д. Ульман. Компиляторы: принципы, технологии и инструментарий. Издание второе. / М.: ООО «И.Д. Вильямс», 2008, ISBN: 978-5-8459-1349-4
2. Р. Андерсон, "Доказательство правильности программ", Москва, Мир, 1982

b. Основная литература

1. K.D. Cooper and L. Torczon. Engineering a Compiler. / Morgan Kaufman Publishers, 2004, ISBN: 1-55860-698-X
2. ANSI/ISO C Specification Language. http://frama-c.com/acsl.html

c. Дополнительная литература

1. Y. N. Srikant, P. Shankar. The Compiler Design Handbook. 2nd edition. CRC press – 2008 ISBN: 978-1-4200-4382-2
2. The **LLVM** Compiler Infrastructure. // http://llvm.org/
3. A GNU Manual. // http://gcc.gnu.org/
4. The SUIF 2 Compiler System. // http://suif.stanford.edu/suif/suif2/
5. Free Compiler Construction Tools. // http://www.thefreecountry.com/programming/compilerconstruction.shtml