ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«24» июня 2025 г.
Протокол №2

Рабочая программа дисциплины (модуля) «Архитектура виртуальных машин»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Искусственный интеллект*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2025*

Москва
2025

Электронный документ

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Архитектура виртуальных машин» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по специальности 02.03.01 Математика и компьютерные науки, профиль Искусственный интеллект, утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Архитектура виртуальных машин» позволяет понять принципы создания и функционирования виртуализированных сред, что является основой для эффективного использования современных вычислительных ресурсов и облачных технологий. Это знание способствует развитию навыков оптимизации работы систем и повышению безопасности при развертывании программного обеспечения в изолированных средах.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 Математика и компьютерные науки, профиль Искусственный интеллект и входит в вариативную часть Блока 1, формируемую участниками образовательных отношений.

Дисциплина (модуль) является выборной и доступна для изучения на 2, 3 или 4 курсе в 4, 5, 6, 7, 8 семестрах на выбор. Прохождение возможно при условии успешного завершения дисциплин: «Архитектура компьютера и операционные системы», «Дизайн компиляторов», «Разработка C++».

Цель изучения дисциплины (модуля): формирование знаний и навыков проектирования, реализации и эффективного использования виртуализированных вычислительных сред для оптимизации работы и управления ресурсами современных информационных систем.

Задачи изучения дисциплины (модуля):

— изучить фундаментальные концепции архитектуры виртуальных машин, включая модели исполнения байт-кода, механизмы интерпретации и компиляции, а также принципы виртуализации аппаратных ресурсов;

— освоить методы анализа и оптимизации производительности виртуальных машин, такие как just-in-time компиляция, управление памятью и стратегии кэширования, с учетом требований к безопасности и эффективности;

— применить теоретические знания на практике путем проектирования, реализации и тестирования простых виртуальных машин, а также анализа существующих систем, таких как JVM или CLR, для решения задач в области программной инженерии.

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

— принципы построения архитектуры команд (ISA) виртуальных машин;

— устройства выполнения кода на виртуальной машине Java;

— алгоритмы автоматического управления динамической памятью (Garbage Collection);

— принципы JIT-компиляции;

уметь:

— реализовать выполнение простого набора команд байткода;

— реализовать алгоритм уборки мусора;

— реализовать генерацию машинного кода;

владеть:

— навыком реализации среды выполнения для простого языка программирования.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
УК-1.	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1.	Знает методы поиска и анализа информации в области разработки, основные принципы критической оценки источников информации и их релевантности
		УК-1.2.	Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем
		УК-1.3.	Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации
ОПК-1.	Способен консультировать и использовать фундаментальные знания в области математического анализа, комплексного и функционального анализа алгебры, аналитической геометрии, дифференциальной геометрии и топологии, дифференциальных уравнений, дискретной математики и математической логики, теории вероятностей, математической статистики и случайных процессов, численных методов, теоретической механики в профессиональной деятельности	ОПК-1.1.	Знает основные концепции и теории в области математического анализа и смежных дисциплин; методы и подходы, используемые в различных областях математики
		ОПК-1.2.	Умеет применять математические методы для решения профессиональных задач
		ОПК-1.3.	Имеет практический опыт разработки и реализация математических моделей в профессиональной деятельности
ОПК-6	Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-6.1.	Знает алгоритмы разработки, компьютерные программы, а также алгоритмы вычислительной математики в области искусственного интеллекта
		ОПК-6.2.	Умеет разрабатывать математические программные продукты и комплексы с использованием современных

			технологий программирования в области искусственного интеллекта
		ОПК-6.3.	Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований в области искусственного интеллекта
ПК-1.	Способен формулировать задачи с математической точностью, обосновывать утверждения строго и анализировать полученные результаты в области математики и компьютерных наук	ПК-1.1.	Обладает базовыми знаниями, полученными в области математических наук, программирования и информационных технологий
		ПК-1.2.	Умеет находить, формулировать и решать стандартные задачи в собственной научно-исследовательской деятельности в математике и информатике
		ПК-1.3.	Имеет опыт работы с задачами в области математики и компьютерных наук, включая применение математических методов для решения практических задач

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Контактная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары (практические занятия)			
1	Архитектура набора команд байткода и его выполнение	6	6		32	Кейс
2	Автоматическое управление памятью	6	6		32	Кейс
3	JIT-компиляция	6	6		32	Кейс
4	Интеграция подсистем и производительность виртуальных машин	6	6		34	Коллоквиум
	Зачет с оценкой			12		Проект
	Итого:	***24***	***24***	***12***	***130***	
Объем дисциплины (модуля) (в ак. ч.)		***190***				
Объем дисциплины (модуля) (в зач. ед.)		***5***				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Архитектура набора команд байткода и его выполнение	Архитектура байткода. Загрузка и верификация классов. Интерпретация команд. Управление потоком исполнения. Оптимизация интерпретатора
2	Автоматическое управление памятью	Основы управления памятью. Поколенческая сборка мусора. Продвинутые алгоритмы GC. Мониторинг и профилировка памяти
3	JIT-компиляция	Основы JIT-компиляции. Архитектура JIT и оптимизация кода. Адаптивные стратегии JIT и деоптимизация
4	Интеграция подсистем и производительность виртуальных машин	Интеграция подсистем виртуальной машины. Производительность и масштабирование. Виртуальные машины других языков: сравнительный анализ

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Клинтон, Л. Дж. Создай свой собственный язык программирования. Руководство программиста по разработке компиляторов, интерпретаторов и доменно-ориентированных языков для решения современных вычислительных задач : практическое руководство / Л. Дж. Клинтон ; пер. с англ. С. В. Минца. - Москва : ДМК Пресс, 2023. - 408 с. - ISBN 978-5-93700-140-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/2109499.

2. Довек, Ж. Введение в теорию языков программирования : научное издание / Ж. Довек, Ж.-Ж. Леви ; пер. с англ. В. Н. Брагилевского, А. М. Пеленицына. - 2-е изд. - Москва : ДМК Пресс, 2023. - 135 с. - ISBN 978-5-89818-582-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/2107945.

3. Орлов, С. А. Теория и практика языков программирования : учебник для вузов / С. А. Орлов. - Санкт-Петербург : Питер, 2021. - 688 с. - (Стандарт 3-го поколения). - ISBN 978-5-4461-9740-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1857045.

4. Кудрявцева, И. А. Программирование: комбинаторная логика : учебник для вузов / И. А. Кудрявцева, М. В. Швецкий. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 524 с. — (Высшее образование). — ISBN 978-5-534-10620-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/565666.

5. Шоу, Э. Внутри CPYTHON: гид по интерпретатору Python : практическое руководство / Э. Шоу. - Санкт-Петербург : Питер, 2023. - 352 с. - (Библиотека программиста). - ISBN 978-5-4461-1925-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/2123388.

6. Малявко А. А. Формальные языки и компиляторы : учебник для вузов / А. А. Малявко. — Москва : Издательство Юрайт, 2025. — 429 с. — (Высшее образование). — ISBN 978-5-534-04288-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/562822.

Дополнительная литература:

7. Новожилов О. П. Архитектура ЭВМ и вычислительных систем : учебник для вузов / О. П. Новожилов. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2025. — 505 с. — (Высшее образование). — ISBN 978-5-534-20365-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/568920.

8. Толстобров А. П. Архитектура ЭВМ : учебник для вузов / А. П. Толстобров. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 162 с. — (Высшее образование). — ISBN 978-5-534-16839-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/566711.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;
— компьютерной техникой;
— механическими калькуляторами;
— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное

Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

Электронный документ

В процессе изучения дисциплины (модуля) «Архитектура виртуальных машин» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, кейсы, коллоквиум, проект, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Семинар — это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Кейс – практическая работа студентов над реальными или смоделированными задачами, что позволяет студенту применять теоретические знания на практике.

Студент самостоятельно разрабатывает стратегию решения поставленной задачи, что способствует развитию навыков критического мышления и самостоятельного принятия решений. Такой подход помогает подготовить будущих специалистов к реальным вызовам в их профессиональной деятельности.

Коллоквиум – устные ответы на вопросы, список которых известен студенту заранее.

В процессе подготовки к коллоквиуму необходимо проанализировать учебные материалы, ознакомившись с лекциями, учебниками и дополнительными источниками, акцентируя внимание на ключевых темах. Рекомендуется создать структурированные конспекты, выделяя основные идеи, термины и формулы.

Проект – исследовательская работа по курсу и презентация результатов.

Для успешной подготовки к проекту: четко определите цели и задачи проекта, распределите роли и обязанности между участниками, а также установите сроки выполнения каждой части работы. Регулярно проводите встречи для обсуждения прогресса и решения возникающих вопросов.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов, планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Архитектура виртуальных машин»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачета с оценкой***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	**Пятибалльная оценка**	**Оценка за зачет**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Зачтено	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	Зачтено	
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно
6	Хорошо	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	Зачтено	
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Архитектура виртуальных машин» оценивается следующим образом:

Активность	Вес	Количество	Описание
Кейс	40%	3	Практическая работа студентов над реальными или смоделированными задачами
Проект	40%	1	Защита итогового проекта
Коллоквиум	20%	1	Устные ответы на вопросы, список которых известен студенту заранее

Формула расчёта итоговой оценки по дисциплине (модулю) «Архитектура виртуальных машин»: «0,4 × среднее за кейсы + 0,4 ×за проект + 0,2 × за коллоквиум».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные задания для кейсов

Архитектура набора команд байткода и его выполнение

Кейс 1: Анализ архитектуры байткода в JVM

В компании разрабатывается новое приложение на Java, и команда сталкивается с проблемой низкой производительности из-за неэффективного байткода. Разработчики анализируют скомпилированный класс-файл с помощью инструментов вроде javap и обнаруживают, что некоторые инструкции (например, aload и invokevirtual) генерируют лишние операции. Необходимо понять, как структура байткода JVM (стековая архитектура, типы инструкций) влияет на исполнение и предложить улучшения. **Вопросы:**

— Как стековая модель байткода JVM отличается от регистровой архитектуры процессоров?
— Какие типы инструкций байткода (load/store, arithmetic, control flow) наиболее распространены и почему?

Задачи:

1. Разберите байткод простого метода (например, вычисления факториала) и опишите последовательность инструкций.
2. Предложите оптимизацию байткода, например, замену на более эффективные инструкции или использование констант.
3. Оцените влияние изменений на размер файла и время исполнения.

Кейс 2: Загрузка и верификация классов в JVM

В процессе развертывания веб-приложения на сервере Tomcat происходит ошибка ClassNotFoundException. Анализ показывает, что загрузчик классов (class loader) не может найти зависимый класс из JAR-файла. Команда проводит верификацию классов с помощью javac и bytecode verifier, обнаруживая несоответствия в сигнатурах методов. Необходимо исследовать механизм загрузки классов и верификации для предотвращения подобных ошибок.

Вопросы:

— Как работает иерархия загрузчиков классов (bootstrap, extension, system/application) в JVM?
— Какие этапы верификации байткода (structural, data flow, bytecode) выполняются и зачем?

Задачи:

1. Моделируйте загрузку классов для простого приложения с зависимостями и опишите порядок загрузки.
2. Выполните верификацию байткода с помощью инструментов (например, java -verify) и исправьте найденные ошибки.
3. Предложите стратегию для безопасной загрузки классов в многопоточной среде.

Кейс 3: Интерпретация команд байткода

Разработчики оптимизируют интерпретатор JVM для мобильного приложения, где батарея быстро разряжается из-за частого декодирования инструкций. Они используют простой интерпретатор (switch-based), который медленно обрабатывает циклы. Необходимо изучить, как интерпретатор декодирует и исполняет байткод, и предложить улучшения.

Вопросы:

— В чем разница между интерпретацией и компиляцией байткода в JVM?
— Как интерпретатор управляет стеком операндов и локальными переменными во время исполнения?

Задачи:

1. Реализуйте простой интерпретатор для подмножества байткода (например, arithmetic operations) и протестируйте на примере.
2. Измерьте производительность интерпретатора (время на 1000 инструкций) и сравните с нативным исполнением.
3. Предложите оптимизацию, такую как кэширование декодированных инструкций или переход на threaded code.

Кейс 4: Управление потоком исполнения в байткоде

В приложении для обработки данных происходит бесконечный цикл из-за ошибки в условных переходах (if_icmpne). Анализ байткода показывает неправильное использование инструкций goto и jsr для обработки исключений. Команда изучает, как JVM управляет потоком (control flow) через стек и метки.

Вопросы:

— Как инструкции control flow (if, goto, tableswitch) изменяют счетчик программ (PC) в JVM?
— В чем роль исключений (try-catch) в управлении потоком и их связь с байткодом?

Задачи:

1. Разберите байткод метода с циклами и условными операторами, выделив точки ветвления.
2. Моделируйте исполнение потока для сценария с исключением и опишите изменения в стеке.
3. Оптимизируйте код, заменив неэффективные переходы на более простые инструкции.

Кейс 5: Оптимизация интерпретатора JVM

В высоконагруженном сервере JVM интерпретатор тратит 30% времени на декодирование инструкций, что приводит к задержкам. Команда внедряет оптимизации, такие как inline caching для вызовов методов. Необходимо проанализировать текущий интерпретатор и предложить улучшения для повышения производительности.

Вопросы:

— Какие стратегии оптимизации интерпретатора (например, superinstructions, fast paths) используются в современных JVM?
— Как измерять эффективность интерпретатора (через профилирование с помощью JMH)?

Задачи:

1. Профилируйте простой интерпретатор на тестовом байткоде и идентифицируйте узкие места.
2. Реализуйте оптимизацию, например, объединение инструкций в супер-инструкции.
3. Сравните производительность до и после оптимизации на метриках (время, CPU usage).

Автоматическое управление памятью

Кейс 1: Основы управления памятью в JVM

В приложении на Java происходит утечка памяти из-за неосвобожденных объектов. Команда анализирует heap с помощью jmap и видит, что объекты String накапливаются. Необходимо понять основы аллокации и освобождения памяти в JVM.

Вопросы:

- Как JVM управляет heap (young/old generation) и stack?
- В чем разница между strong, soft, weak и phantom references?

Задачи:

1. Создайте простое приложение с аллокацией объектов и проанализируйте heap с помощью VisualVM.
2. Моделируйте утечку памяти и опишите, как GC ее обнаруживает.
3. Предложите исправления, такие как использование weak references.

Кейс 2: Поколенческая сборка мусора

Серверное приложение испытывает паузы GC до 5 секунд из-за частой сборки в young generation. Анализ с помощью jstat показывает несбалансированные размеры поколений. Команда изучает поколенческую модель (Eden, Survivor, Old).

Вопросы:

- Как работает minor GC в young generation и major GC в old generation?
- Почему поколенческая сборка эффективнее полной сборки?

Задачи:

1. Настройте JVM параметры (-Xms, -Xmx, -XX:NewRatio) и протестируйте на приложении.
2. Измерьте время пауз GC и сравните с разными размерами поколений.

3. Оптимизируйте конфигурацию для снижения пауз.

Кейс 3: Продвинутые алгоритмы GC

В высокопроизводительном приложении используется G1 GC, но возникают фрагментации. Команда рассматривает переход на ZGC или Shenandoah для низких пауз. Необходимо сравнить алгоритмы (mark-sweep, copying, concurrent).

Вопросы:

- Как G1 GC делит heap на регионы и управляет паузами?
- В чем преимущества concurrent GC (ZGC) над stop-the-world?

Задачи:

1. Реализуйте симуляцию простого GC-алгоритма (например, mark-and-sweep) на тестовых данных.
2. Сравните производительность G1 и Parallel GC на бенчмарке.
3. Предложите выбор GC для сценария с низкими паузами.

Кейс 4: Мониторинг и профилировка памяти

Приложение на сервере имеет высокое потребление памяти (RAM usage 80%). Команда использует JMX и jconsole для мониторинга heap и threads. Необходимо выявить причины и оптимизировать.

Вопросы:

- Какие метрики (heap size, GC count, memory leaks) важны для мониторинга?
- Как инструменты вроде MAT помогают в профилировке?

Задачи:

1. Настройте мониторинг JVM с помощью JFR и проанализируйте дамп heap.
2. Идентифицируйте утечку с помощью MAT и предложите исправления.
3. Создайте отчет с рекомендациями по оптимизации памяти.

Кейс 5: Интеграция GC с приложением

В микросервисах на Spring Boot GC вызывает задержки из-за больших объектов. Команда интегрирует профилировщик (Async Profiler) для анализа. Необходимо оптимизировать управление памятью.

Вопросы:

- Как GC взаимодействует с JIT-компилятором для оптимизации аллокаций?
- В чем роль escape analysis в управлении памятью?

Задачи:

1. Профилируйте приложение с Async Profiler и найдите горячие точки аллокаций.
2. Внедрите оптимизацию, такую как object pooling для больших объектов.
3. Измерьте улучшения в производительности после изменений.

Примерные задания для коллоквиума

1. Объясните, как интегрируются подсистемы JVM, такие как загрузчик классов, интерпретатор и JIT-компилятор. Приведите пример взаимодействия на этапе выполнения метода.
2. Как garbage collector (GC) взаимодействует с JIT-компилятором в JVM? Опишите, как это влияет на оптимизацию кода.
3. Опишите роль байткода в интеграции подсистем JVM. Как он обеспечивает совместимость между интерпретатором и JIT?
4. Как подсистема управления памятью (heap и stack) интегрируется с подсистемой выполнения байткода? Приведите пример с аллокацией объектов.
5. Объясните интеграцию подсистем в многопоточной среде JVM: как работают мониторы и синхронизация с GC и JIT.
6. Как загрузчик классов взаимодействует с верификатором байткода и JIT-компилятором? Почему это важно для безопасности?
7. Опишите, как JVM интегрирует нативные вызовы (JNI) с внутренними подсистемами, такими как GC. Приведите пример.

8. Какие ключевые метрики используются для оценки производительности JVM (например, throughput, latency, memory usage)? Приведите примеры инструментов для их измерения.

9. Объясните, как масштабируется JVM в многопроцессорных системах. Какие ограничения возникают при увеличении числа ядер?

10. Как оптимизировать производительность JVM в контейнеризованных средах (например, Docker)? Опишите влияние на GC и JIT.

11. Опишите стратегии масштабирования JVM для высоконагруженных приложений: горизонтальное vs вертикальное масштабирование.

12. Как профилирование (например, с помощью JFR или Async Profiler) помогает улучшить производительность JVM? Приведите пример анализа узкого места.

13. Объясните влияние размера heap на производительность и масштабирование JVM. Как выбрать оптимальные параметры (-Xmx, -Xms)?

14. Как JVM обеспечивает масштабирование в распределенных системах (например, с использованием микросервисов)? Опишите роль в сериализации и сетевых взаимодействиях.

15. Сравните JVM с виртуальной машиной JavaScript (V8). В чем преимущества и недостатки V8 по сравнению с JIT-компиляцией в JVM?

16. Опишите различия между JVM и .NET CLR (Common Language Runtime). Как CLR управляет памятью по сравнению с GC в JVM?

17. Сравните производительность JVM и CPython (виртуальная машина Python). Почему CPython медленнее в численных вычислениях?

18. Как Lua VM отличается от JVM по архитектуре? Приведите примеры использования и сравните масштабируемость.

19. Проведите сравнительный анализ JVM и виртуальной машины Ruby (YARV). В чем сильные стороны YARV в динамической типизации?

20. Объясните, почему JVM часто считается более производительной для enterprise-приложений по сравнению с VM интерпретируемых языков (например, CPython). Поддержите аргументы примерами.

Примерное описание и критерии оценивания к итоговому проекту

Описание проекта:

В рамках итогового проекта студенты должны разработать прототип виртуальной машины (ВМ), способной выполнять платформонезависимый байткод с использованием оптимизированного набора команд, реализовать автоматическое управление памятью и внедрить базовую JIT-компиляцию для повышения производительности.

Основные задачи проекта:

1. **Архитектура байткода и его выполнение**
   - — Спроектировать компактный и оптимизированный набор команд байткода, включающий операции работы с памятью, стеком и вызовами.
   - — Обеспечить платформонезависимость байткода и переносимость приложений.
   - — Реализовать интерпретатор, выполняющий байткод.
2. **Автоматическое управление памятью**
   - — Реализовать механизм автоматического выделения и освобождения памяти.
   - — Внедрить один из основных алгоритмов сборки мусора (подсчёт ссылок, маркировка или копирование).
   - — Продемонстрировать влияние выбранного алгоритма на производительность и устойчивость работы ВМ.
3. **JIT-компиляция**
   - — Реализовать простой JIT-компилятор, преобразующий байткод в машинный код.
   - — Оптимизировать горячие участки кода для повышения скорости исполнения.

— Обеспечить сочетание переносимости (через байткод) и эффективности (через JIT).
— Подготовить анализ адаптивных оптимизаций и их влияния на производительность.

4. **Документация и демонстрация**
   — Подготовить техническое описание архитектуры ВМ, используемых алгоритмов и решений.
   — Провести тестирование и предоставить сравнительный анализ производительности с и без JIT, а также с разными методами управления памятью.
   — Представить проект в виде презентации с демонстрацией работы.

Критерии оценивания:

- Архитектура байткода
- Реализация интерпретатора
- Автоматическое управление памятью
- JIT-компиляция
- Документация и презентация
- Тестирование и анализ результатов

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Укажите основной формат кода в архитектуре байткода JVM.	байткод	УК-1
2	Назовите процесс проверки классов на соответствие спецификациям JVM перед загрузкой.	верификация	УК-1
3	Укажите метод выполнения байткода без компиляции в машинный код.	интерпретация	УК-1
4	Назовите механизм JVM для обработки условных переходов в потоке исполнения.	управление потоком	УК-1
5	Укажите подход к ускорению интерпретатора байткода через кэширование.	оптимизация интерпретатора	УК-1
6	Назовите базовый принцип выделения памяти в JVM для объектов.	куча (heap)	ОПК-1
7	Укажите количество поколений в стандартной поколенческой сборке мусора JVM.	три (young, old, permanent/metaspace)	ОПК-1
8	Назовите алгоритм GC, использующий копирование для young generation.	копирующий коллектор	ОПК-1
9	Укажите метрику для оценки утечек памяти в JVM.	профилировка памяти	ОПК-1
10	Назовите математический метод оценки времени пауз GC в JVM.	анализ остановок (stop-the-world)	ОПК-1
11	Укажите тип компилятора, преобразующего байткод в машинный код во время выполнения.	JIT-компилятор	ОПК-6
12	Назовите процесс удаления оптимизаций JIT при изменении поведения кода.	деоптимизация	ОПК-6

13	Укажите стратегию JIT для компиляции часто выполняемых методов.	горячая точка (hot spot)	ОПК-6
14	Назовите алгоритм оптимизации кода в JIT для устранения мертвого кода.	dead code elimination	ОПК-6
15	Укажите подход к интеграции подсистем JVM для параллельного выполнения.	многопоточность	ОПК-6
16	Назовите базовую структуру JVM для хранения классов и методов.	классовая иерархия	ПК-1
17	Укажите математическую модель для оценки производительности сборки мусора.	mark-sweep-compact	ПК-1
18	Назовите процесс верификации байткода как задачу в теории графов.	достижимость состояний	ПК-1
19	Укажите метрику масштабирования JVM в многопроцессорных системах.	параллелизм потоков	ПК-1
20	Назовите сравнительный аспект V8 (для JavaScript) по отношению к JVM.	JIT-компиляция	ПК-1