Министерство образования и науки Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
Институт **Компьютерные технологии и защита информации**

Кафедра **Компьютерные системы**

УТВЕРЖДАЮ
Ответственный за ОП
Верш И.С.Вершинин
«31» 08 2017 г.
Регистрационный номер 4010 – 17/11 – 105

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

для проведения промежуточной аттестации обучающихся по дисциплине

Распознавание и защита данных в геоинформационных системах
(наименование дисциплины, практики)

Индекс по учебному плану: **Б1.В.ДВ.03.01**

Направление подготовки **09.04.01 «Информатика и вычислительная техника»**

Квалификация: **магистр**

Магистерская программа: **Высокопроизводительные вычислительные системы**

Виды профессиональной деятельности: **научно-исследовательская**

Заведующий кафедрой КС И.С. Вершинин
Разработчики :доцент кафедры КС Р.Ф. Гибадуллин ,

Казань 2017 г.

Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине (модулю)

Распознавание и защита данных в геоинформационных системах

(наименование дисциплины)

Содержание фонда оценочных средств (ФОС) соответствует требованиям федерального государственного образовательного стандарта высшего образования (ФГОС ВО) по направлению подготовки 09.04.01 «Информатика и вычислительная техника», учебному плану по направлению подготовки 09.04.01 «Информатика и вычислительная техника».

Разработанные ФОС обладают необходимой полнотой и являются актуальными для оценки компетенций, осваиваемых обучающимися при изучении дисциплины «Распознавание и защита данных в геоинформационных системах». Разработанные ФОС полностью соответствуют задачам будущей профессиональной деятельности обучающихся, установленных ФГОС ВО по направлению подготовки 09.04.01 «Информатика и вычислительная техника». В составе ФОС присутствуют оценочные средства в виде тестовых заданий и контрольных вопросов различного уровня сложности, которые позволяют провести оценку порогового, продвинутого и превосходного уровней освоения компетенций по дисциплине.

ФОС обладают необходимой степенью приближенности к задачам будущей профессиональной деятельности обучающихся, связанным со способностью организовывать работу малых коллективов исполнителей, принимать управленческие решения в сфере профессиональной деятельности, разрабатывать предложения по совершенствованию модулей распознавания и защиты данных в геоинформационных системах.

Существенные недостатки отсутствуют.

Заключение. Учебно-методическая комиссия делает вывод о том, что представленные материалы соответствуют требованиям ФГОС ВО по направлению подготовки 09.04.01 «Информатика и вычислительная техника» и рекомендуются для использования в учебном процессе.

Рассмотрено на заседании учебно-методической комиссии института КТЗИ от «31» августа 2017 г., протокол № 8.

Председатель УМК института КТЗИ _____ В.В. Родионов

Содержание

Введение

Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине «Распознавание и защита данных в геоинформационных системах» – это комплект методических и контрольно-измерительных материалов, предназначенных для определения уровня сформированности компетенций, оценивания знаний, умений, владений на разных этапах освоения дисциплины для проведения промежуточной аттестации обучающихся по дисциплине.

ФОС ПА является составной частью учебного и методического обеспечения программы магистратуры по направлению подготовки 09.04.01 «Информатика и вычислительная техника».

Задачи ФОС по дисциплине «Распознавание и защита данных в геоинформационных системах»:

– оценка запланированных результатов освоения дисциплины обучающимися в процессе изучения дисциплины, в соответствии с разработанными и принятыми критериями по каждому виду контроля;

– контроль и управление процессом приобретения обучающимися необходимых знаний, умений, навыков и формирования компетенций, определенных в ФГОС ВО по направлению подготовки

ФОС ПА по дисциплине «Распознавание и защита данных в геоинформационных системах» сформирован на основе следующих основных принципов оценивания:

– пригодности (валидности) (объекты оценки соответствуют поставленным целям обучения);

– надежности (использования единообразных стандартов и критериев для оценивания запланированных результатов);

– эффективности (соответствия результатов деятельности поставленным задачам).

ФОС ПА по дисциплине «Распознавание и защита данных в геоинформационных системах» разработан в соответствии с требованиями ФГОС ВО по направлению подготовки 09.04.01 «Информатика и вычислительная техника» для аттестации обучающихся на соответствие их персональных достижений требованиям поэтапного формирования соответствующих составляющих компетенций и включает контрольные вопросы (или тесты) и типовые задания, необходимые для оценки знаний, умений и навыков, характеризующих этапы формирования компетенций.

1. Формы промежуточной аттестации по дисциплине

Дисциплина «Распознавание и защита данных в геоинформационных системах» изучается в 1 семестре при очной форме обучения и завершается промежуточной аттестацией в форме экзамена.

2. Оценочные средства для промежуточной аттестации

Оценочные средства для промежуточной аттестации по дисциплине «Распознавание и защита данных в геоинформационных системах» при очной форме обучения.

Таблица 1

Оценочные средств для промежуточной аттестации
(очная форма обучения)

№ п/п	Семестр	Форма промежуточной аттестации	Оценочные средства
1.	1	Экзамен	ФОС ПА

3. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

Перечень компетенций и их составляющих, которые должны быть сформированы при изучении темы соответствующего раздела дисциплины «Распознавание и защита данных в геоинформационных системах», представлен в таблице 2.

Таблица 2

Перечень компетенций и этапы их формирования
в процессе освоения дисциплины

№ п/п	Этап формирования (семестр)	Наименование раздела	Код формируемой компетенции (составляющей компетенции)		Форма промежуточной аттестации
1	1	Начальные понятия и предпосылки	ОПК-6	ОПК-6.З, ОПК-6.В	Экзамен
2	1	Управление картографическими базами данных	ПК-4	ПК-4.З, ПК-4.В	Экзамен
3	1	Модели, инструментальные средства	ОПК-6 ПК-4	ОПК-6.З, ОПК-6.У, ОПК-6.В ПК-4.З, ПК-4.У, ПК-4.В	Экзамен

4. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описания шкалы оценивания

Показатели и критерии оценивания сформированности компетенций на экзамене, приведены в таблице 3.

Таблица 3

Показатели и критерии оценивания сформированности компетенций на экзамене

№ п/п	Этап формирования (семестр)	Код формируемой компетенции (составляющей компетенции)		Критерии оценивания	Показатели оценивания (планируемые результаты обучения)		
					Пороговый уровень	Продвинутый уровень	Превосходный уровень
1.	1	ОПК-6 ПК-4	ОПК-6.З ОПК-6.У ПК-4.З ПК-4.У	Теоретические навыки	- знать тенденции развития и классификацию параллельных систем; - знать элементы теории коммутационных сетей; - знать состав библиотек параллельного программирования - умение объяснить достоинства и недостатки представителей параллельных систем; - умение объяснить принципы построения многоуровневой памяти; - умение объяснить структуру библиотек параллельного программирования	- знать принципы построения и функционирования архитектуры параллельных программ: - знать стандарт OpenMP; - знать принципы управления задачами - умение объяснить функционирование пакетов MPICH и MPI.NET	- знать функционирования DAP и ПМА; - знать принципы использования прикладных библиотек NVIDIA CUDA и шаблонов Thrust - умение объяснить принципы программирования видеокарт
2.	1	ОПК-6 ПК-4	ОПК-6.В ПК-4.В	Практические навыки	*Владеть навыками*: - оценки производительности параллельных систем; - анализа архитектур суперпроцессоров; - работы с разделяемыми данными и параллельными циклами	*Владеть навыками*: - решения типовых задач с использованием пакетов MPICH и MPI.NET	*Владеть навыками*: - программирования видеокарт nVidia

Формирование оценки при промежуточной аттестации по итогам освоения дисциплины зависит от уровня освоения компетенций, которые обучающийся должен освоить по данной дисциплине. Связь между итоговой оценкой и уровнем освоения компетенций (шкала оценивания) представлена в таблице 4.

Таблица 4

Описание шкалы оценивания

Шкала оценивания		Описание оценки в требованиях к уровню и объему компетенций
Словесное выражение	Выражение в баллах	
Отлично	от 86 до 100	Освоен **превосходный** уровень всех компетенций (составляющих компетенций)
Хорошо	от 71 до 85	Освоен **продвинутый** уровень всех компетенций (составляющих компетенций)
Удовлетворительно	от 51 до 70	Освоен **пороговый** уровень всех компетенций (составляющих компетенций)
Неудовлетворительно	до 51	Не освоен **пороговый** уровень всех компетенций (составляющих компетенций)

5. Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Формирование оценки по результатам текущего контроля успеваемости и промежуточной аттестации по итогам освоения дисциплины «Распознавание и защита данных в геоинформационных системах» приведено в таблице 5.

Таблица 5

Формирование оценки по итогам освоения дисциплины

Наименование контрольного мероприятия	Рейтинговые показатели				
	I аттестация	II аттестация	III аттестация	по результатам текущего контроля	по итогам промежуточной аттестации (зачета /экзамена)
Разделы 1. Начальные понятия и предпосылки	**10**			**10**	
Тест текущего контроля по разделу	10			10	
Разделы 2. Управление картографическими базами данных		**10**		**10**	
Тест текущего контроля по разделу		5		5	
Защита лабораторных работ		5		5	
Раздел 3. Модели, инструментальные средства			**30**	**30**	
Тест текущего контроля по разделу			10	10	
Защита лабораторных работ			20	20	
Промежуточная аттестация (Экзамен):					**50**
– тест промежуточной аттестации по дисциплине					20
– ответы на контрольные вопросы в письменной форме по билетам					30

6. Контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения дисциплины

6.1. Тестовые задания

Тестовые задания на тестирование 1

1

Параллелизм может быть введен в архитектуру ЭВМ способами:
1. Конвейерная обработка
2. Функциональная обработка
3. Операционная обработка
4. Матричная обработка
5. Мультипроцессорная обработка

2

Предоставление нескольким независимым устройствам возможности выполнения различных функций
1. Конвейерная обработка
2. Функциональная обработка
3. Матричная обработка
4. Мультипроцессорная обработка

3

Наличие совокупности идентичных процессорных элементов с общей системой управления, где все элементы в каждый момент времени выполняют одну и ту же операцию, но с разными данными
1. Конвейерная обработка
2. Функциональная обработка
3. Матричная обработка
4. Мультипроцессорная обработка

4

SMP-системы
1. Многопроцессорные системы с массовым параллелизмом
2. Мультипроцессорная система с общей памятью
3. Связанный набор полноценных компьютеров, используемый в качестве единого ресурса

5

MPP-системы
1. Многопроцессорные системы с массовым параллелизмом
2. Мультипроцессорная система с общей памятью
3. Связанный набор полноценных компьютеров, используемый в качестве единого ресурса

6

Систематика Флинна
1. Классификация параллельных архитектур на основе того, как в машине ее команды взаимодействуют с обрабатываемыми данными
2. Классификация параллельных архитектур с учетом составных частей машины
3. Классификация параллельных архитектур, принимающая за основу: число устройств команд, способы введения параллелизма, тип используемых команд, тип управления, виды конвейеров и исполнительных устройств, степень связности

7

FLOPS
1. Величина показывающая, сколько эквивалентных скалярных операций с плавающей запятой в секунду выполняет данная вычислительная система
2. Величина показывающая, сколько эквивалентных скалярных операций с фиксированной запятой в секунду выполняет данная вычислительная система
3. Величина показывающая, сколько операций деления в секунду выполняет данная вычислительная система
4. Величина показывающая, сколько операций умножения в секунду выполняет данная вычислительная система

8

Пусть f – доля трудозатрат последовательной программы, которую нельзя распараллелить, $0 \le f \le 1$. Тогда достижимое ускорение S выполнения программы на параллельном компьютере из N процессоров
1. $S \le \frac{1}{f + (1-f)/N}$
2. $S \le \frac{1}{f/N + f}$
3. $S \le \frac{N}{f + (1-f)/N}$

9

Пусть операция сложения в машине происходит за пять тактов. Требуется сложить два вектора, состоящих из 100 чисел каждый. Через какое число тактов завершится сложение векторов с использованием конвейерного устройства, состоящего из пяти ступеней, срабатывающих за один такт каждая, если перед началом сложения векторов конвейер пустой?

1. 20
2. 54
3. 100
4. 104

10

Пусть в программе есть лишь 2% последовательных операций. Какого максимально возможного ускорения работы программы можно достичь при наличии неограниченного числа вычислительных ресурсов?

1. 10-кратного ускорения
2. 50-кратного ускорения
3. 100-кратного ускорения
4. 500-кратного ускорения

11

Тест LINPACK – тест для оценки производительности

1. Только процессора
2. Только файловой системы
3. Только коммутационной сети
4. Только СУБД
5. Кобинированный тест

12

К какому классу вычислительных систем относятся векторные и матричные процессоры?

1. SISD
2. SIMD
3. MISD
4. MIMD

13

К какому классу вычислительных систем относятся системы с общей памятью (мультипроцессоры)?

1. SISD
2. SIMD
3. MISD
4. MIMD

14

Если время такта компьютера равно 2,5 нс, и за каждый такт он может выполнять две операции, чему равна пиковая производительность этого компьютера?

1. 2×10^8
2. 4×10^8
3. 8×10^8
4. 4×10^9

15

Логико-запоминающая среда

1. Итеративная двумерная структура, выполняющая произвольный набор процедур независимо от размеров среды
2. Итеративная двумерная структура, выполняющая заданный набор процедур независимо от размеров среды
3. Итеративная двумерная структура, выполняющая заданный набор процедур в зависимости от размеров среды

16

Задача синтеза ЛЗС сводится к определению

1. Количества элементов памяти
2. Структуры связей
3. Схемы элемента
4. Граничных сигналов

17

Для поиска ближайшего большего на основе ЛЗС необходимо выполнение операций

1. Деления массива на три части
2. Поиска максимума
3. Поиска минимума

18

Пиковая производительность характеризует

1. Уровень технологии
2. Архитектуру
3. То и другое вместе
4. Степень параллелизма в архитектуре

19

Длина полупроизводительности характеризует

1. Уровень технологии
2. Архитектуру
3. То и другое вместе
4. Степень параллелизма в архитектуре

20

Наибольшее влияние на эффективность вычислительного эксперимента оказывает
1. Математическая модель
2. Численный метод
3. Программа
4. Компилятор
5. Операционная система
6. Суперкомпьютерная платформа

21

Системная производительность учитывает
1. Процессорное время
2. Организацию обменов
3. Переформирование данных
4. Организацию циклов и переходов

22

Устанение зернистого шума полутонового изображения достигается применением
1. Фильтра средних значений
2. Медианного фильтра
3. Фильтра улучшения контрастности

23

Устранение дефектов бинарного линейного изображения достигается операциями
1. Увеличения
2. Уменьшения
3. Поочередным использованием обеих операций

24

При передаче полутоновых изображений используется кодирование
1. С линейным прогнозированием
2. С переменной длиной кода
3. Комбинация двух предыдущих методов

25

Наиболее важная и трудоемкая операция над цепочкой символов –
1. Конкатенация
2. Сопоставление
3. Замещение
4. Выборка

26

trie-дерево – средство
1. Эффективного поиска в словаре
2. Обработки морфем
3. Синтаксической обработки
4. Семантической обработки

27

Обработка естественных языков связана с обработкой
1. Слов
2. Предложений
3. Текста

28

CFG –
1. Правила анализа контекста
2. Универсальный язык программирования
3. Специальная грамматика английского языка

29

Операции ассоциативной памяти
1. Хранение
2. Сравнение
3. Мультизапись
4. Чтение
5. Арифметические
6. Символьные
7. Логические

30

Память отклика АПП
1. Память констант

2. Память теговых разрядов
3. Память признаков

31

Приоритетная схема АПП реализует
1. Выбор первого ответчика
2. Чтение первого признака
3. Запись первого операнда

32

Команды АПП
1. SET
2. COMPARE
3. WRITE
4. READY
5. FIRST
6. COUNT
7. COMPOZITE
8. REPORT
9. MOVE

33

Обработка в АПП ведется
1. Последовательно по разрядам, параллельно по словам
2. Последовательно по словам, параллельно по разрядам
3. Параллельно по словам и по разрядам

34

Ортогональная память предназначена для построения
1. Конвейерно-векторных систем
2. SMP-систем
3. MPP-систем
4. Ассоциативных параллельных процессоров

35

По эффективности арифметической обработки предпочтительны
1. СВО
2. СГО
3. Обе системы равноценны

36

В системе ICL DAP реализовано
1. Подключение DAP к Host на уровне внешнего устройства
2. Подключение на уровне спецпроцессора
3. Прямой доступ в Host

37

Необходимость введения акселераторов в СВО вызвана
1. Зависимостью времени выполнения операций от числа разрядов операнда
2. Требованием снижения времени маршрутизации
3. Ограничениями по вводу / выводу

38

ПМА реализует команды
1. Структурной обработки
2. Логической обработки
3. Арифметической обработки
4. Символьной обработки
5. Регистровые
6. Работы с оперативной памятью
7. Управления

39

Для выполнения символьной обработки в ПМА используется
1. Указатель байта
2. Схема блокировки байта
3. ALU
4. AC

40

Наибольшее ускорение на комплексе Host-ПМА достигается при выполнении
1. Представительских программ
2. Матричных команд и подпрограмм ПМА
3. Выполнение тех и других равноэффективно

41

Первый проход сортировки на ПМА реализует
1. Сортировку отдельных слоев памяти данных
2. Попарную сортировку слоев памяти данных

3. Сортировку части слоев памяти данных

42

Второй проход сортировки на ПМА реализует *(n – число строк операционной матрицы, N – число слоев памяти данных)*

1. Попарную сортировку слоев памяти данных
2. Слияние в блоках по ‘n’ ранее отсортированных слоев памяти данных
1. 3. Слияние в блоках по N/2 ранее отсортированных слоев памяти данных

43

Третий проход сортировки на ПМА реализует *(n – число строк операционной матрицы, N – число слоев памяти данных)*

1. Слияние N/n ранее отсортированных блоков памяти данных
2. Слияние двух блоков по N/2 ранее отсортированных слоев памяти данных
3. Слияние всех ранее отсортированных слоев памяти данных

44

Укажите возвращаемый тип функции *cudaFree()*.

Выберите один из 4 вариантов ответа:

1)		int
2)		cudaError_t
3)		float
4)		void

45

Укажите действие, которое произойдет при выполнении кода:
*float *a;*
cudaMalloc(a, 10);

Выберите один из 5 вариантов ответа:

1)		Выделится память для массива из 10 float в памяти GPU.
2)		Программа отработает некорректно.
3)		Выделится память для массива из 10 float в оперативной памяти.
4)		Выделится 10 байт памяти в памяти GPU.
5)		Выделится 10 байт памяти в оперативной памяти.

46

Укажит_е значения полей переменной Dg после исполнения кода:
dim3 Db(32, 32);
dim3 Dg(1024 / Db.x, 1024 / Db.y, 1024 / Db.z);

Выберите один из 4 вариантов ответа:

1)		32, 32, 1024
2)		1024, 32, 32
3)		32, 32, 1
4)		1, 32, 32

47

Выберите верное утверждение о GPU:

Выберите один из 5 вариантов ответа:

1)		Ничто из перечисленного.
2)		В чипах современных видеокарт содержится от 1 до 15 вычислительных ядер.
3)		Управлением контекстом нитей исполнения на GPU занимается операционная система.
4)		Для проведения общих вычислений на GPU необходимы установленный CUDA Toolkit и видеокарта поколения Tesla.
5)		Оперативная память GPU оптимизирована для коллективного доступа.

48

SM – это:

Выберите один из 5 вариантов ответа:		
1)		Signal Multiplier
2)		Streaming Multiprocessor
3)		Sampling Multicore
4)		Ничто из перечисленного
5)		Steaming Multiplex

49		
Укажите верное вычисление линейного индекса нити в случае двумерной сетки (NxN) из двумерных блоков (32x32) (подходит любой вариант, при котором у различных нитей различные индексы и есть нить, индекс которой равен любому числу от 0 до 1024 * N * N - 1):		
Выберите один из 7 вариантов ответа:		
1)		blockIdx.x * blockDim.x + threadIdx.x
2)		(blockIdx.y * blockDim.y + threadIdx.y) + (blockIdx.x * blockDim.x + threadIdx.x)
3)		(blockIdx.y * blockDim.y + threadIdx.y) * (blockIdx.x * blockDim.x + threadIdx.x)
4)		(blockIdx.y * blockDim.y + threadIdx.y) * (blockDim.y + blockDim.x) + (blockIdx.x * blockDim.x + threadIdx.x)
5)		((blockIdx.y * blockDim.y + threadIdx.y) * gridDim.x * blockDim.x) + blockIdx.x * blockDim.x + threadIdx.x
6)		Ничто из перечисленного
7)		((blockIdx.y * blockDim.y + threadIdx.y) * gridDim.x * blockDim.x) + gridDim.x * blockIdx.x * blockDim.x + threadIdx.x

---

50		
Укажите вывод программы (“comp”, если программа не скомпилируется и “err”, если произойдет ошибка при исполнении):		

```
  global   void kernel(int *a){
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    if (tid == 0)
        (*a)++;
}
int main(){
    int ha = 2;
    int *da;
    cudaMalloc(&da, sizeof(int));
    cudaMemcpy(da, &ha, sizeof(int), cudaMemcpyHostToDevice);
    kernel <<<2, 5>>> (da);
    cudaMemcpy(&ha, da, sizeof(int), cudaMemcpyDeviceToHost);
    printf("%d\n", ha);
    return 0;
}
```

Выберите один из 5 вариантов ответа:		
1)		err
2)		3
3)		4
4)		comp
5)		12

51		
Укажите вывод программы (“comp”, если программа не скомпилируется и “err”, если произойдет ошибка при исполнении):		

```
  global   void kernel(int *a){
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    if (tid == 0)
        (*a)++;
}
int main(){
    int ha = 2;
```

```
    int *da;
    cudaMalloc(&da, sizeof(int));
    cudaMemcpy(da, &ha, sizeof(int), cudaMemcpyHostToDevice);
    kernel <<<2, 5>>> (ha);
    cudaMemcpy(&ha, da, sizeof(int), cudaMemcpyDeviceToHost);
    printf("%d\n", ha);
    return 0;
}
```

Выберите один из 5 вариантов ответа:

1)		3
2)		comp
3)		err
4)		12
5)		2

52

К какой модели исполнения относятся GPU NVIDIA?

Выберите один из 5 вариантов ответа:

1)		SIMT
2)		SIMD:SMP
3)		Ни одно из вышеперечисленных
4)		MIMD
5)		SIMD

53

Выберите неверное утверждение о GPU:

Выберите один из 5 вариантов ответа:

1)		Порядок исполнения блоков на SM-ах непредсказуем
2)		Нити warp-а не нуждаются в синхронизации
3)		Все нити имеют одинаковые привилегии на доступ к памяти
4)		Существует локальная синхронизация внутри блока нитей
5)		Количество нитей в блоке должно быть кратно размеру warp-а

54

Какова максимальная по размеру единица исполнения, которая выполняется на потоковом мультипроцессоре?

Выберите один из 4 вариантов ответа:

1)		Блок
2)		Сетка
3)		Нить
4)		Warp

55

Укажите тип, который возвращает большинство функций из CUDA Runtime библиотеки:

Выберите один из 5 вариантов ответа:

1)		error_t
2)		cudaError_t
3)		cudaError
4)		Ни один из перечисленных
5)		cudaSuccess

56

Сколько warp-ов в линейном блоке размера 1024 будут исполнять несколько ветвей условного оператора:

```
switch(threadIdx.x / 32){
    case 0:
        …
        break;
    case 1:
        …
        break;
    …
    case 31:
        …
    break;
}
```

Выберите один из 5 вариантов ответа:

1)		8
2)		16
3)		1
4)		0
5)		32

57

Сколько всего warp-ов будет создано при вызове ядра:
kernel <<<dim3(12, 14), dim3(24, 27)>>> ();

Выберите один из 5 вариантов ответа:

1)		3402
2)		32
3)		3528
4)		8
5)		16

58

Выберите верное описание действия функции cudaDeviceSynchronize():

Выберите один из 4 вариантов ответа:

1)		Выполняет глобальную синхронизацию сетки в GPU коде
2)		Вызывается в Host-коде, выполняет параллельное завершение всех запущенных задач на GPU
3)		Выполняет локальную синхронизацию блока в GPU коде
4)		Вызывается в Host-коде, выполняет синхронизацию всех запущенных задач на GPU

59

Завершится ли указанный код с ошибкой, при условии его выполнения в системе с GPU Compute Capability >= 2.0?

```
  global   void kernel(int *A, int *B, int *C){
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    C[tid] = A[tid] + B[tid];
}
int main(){
    int n = 2000;
    int *A, *B, *C;
    cudaMalloc(&A, n*n*sizeof(int));
    cudaMalloc(&B, n*n*sizeof(int));
    cudaMalloc(&C, n*n*sizeof(int));
    kernel <<<n, n>>>(A, B, C);
    cudaDeviceSynchronize();
```

```
    printf("%s\n", cudaGetErrorString(cudaGetLastError()));
    return 0;
}
```

Выберите один из 2 вариантов ответа:

1)		Да.
2)		Нет.

60

В каком типе памяти может быть расположен массив A при вызове функции:

```
__global__ void axpy(double *A, double *B, double *C, int n, double a){
  int tid = threadIdx.x + blockIdx.x * blockDim.x;
  C[tid] = A[tid] * a + B[tid];
}
```

Выберите один из 5 вариантов ответа:

1)		Ни одна из перечисленных
2)		Локальная память GPU
3)		Все перечисленные
4)		Регистровая память GPU
5)		Глобальная память GPU

61

Выберите верное утверждение о GPU:

Выберите один из 5 вариантов ответа:

1)		Количество регистров, которые используются во всех нитях должно быть одинаковым
2)		С помощью записей в глобальную память можно организовать синхронизацию двух блоков
3)		Регистры GPU именуются подобно регистрам CPU
4)		В случае нехватки регистровой памяти начинается использование DRAM GPU
5)		С помощью записей в регистровую память можно организовать синхронизацию двух блоков

62

cuda-memcheck – это:

Выберите один из 4 вариантов ответа:

1)		Утилита, позволяющая выявить некоторые ошибки при использовании GPU памяти
2)		Ни одно из перечисленных
3)		Функция runtime-библиотеки CUDA, позволяющая выявить некоторые ошибки при использовании GPU памяти. Возвращает тип cudaError_t
4)		Спецификатор функции-ядра, указывающий на автоматическую проверку ошибок работы с памятью при его исполнении

63

Выберите верное утверждение о скоростях передачи по интерфейсу PCI-E и скорости чтения из глобальной памяти

Выберите один из 6 вариантов ответа:

1)		PCI-E обычно в 5-10 раз медленнее DRAM GPU
2)		PCI-E обычно в 100-200 раз медленнее DRAM GPU
3)		PCI-E обычно в 5-10 раз быстрее DRAM GPU
4)		PCI-E обычно такой же по скорости, что и DRAM GPU
5)		PCI-E обычно в 20-30 раз медленнее DRAM GPU
6)		Ни одно из перечисленного

64

В коде, исполняемом на GPU, обычно заменяют многомерные массивы одномерными, поскольку:

Выберите один из 5 вариантов ответа:

1)		Уменьшается количество обращений в память
2)		Используя cudaMalloc невозможно выделить многомерные массивы
3)		CUDA-программисты любят запутывать код
4)		Появляется возможность для обработки под-массивов в одном блоке нитей
5)		В таком случае проще работать с двухуровневой иерархией нитей

65

Выберите верное утверждение о GPU:

Выберите один из 4 вариантов ответа:

1)		При одновременной работе нескольких процессов, использующих GPU, ни один из них не может прочитать данные другого
2)		При вызове функции cudaMalloc, выделяемая память автоматически инициализируется нуля-
3)		Есть возможность проводить копирование данных на GPU и производить вычисления
4)		После окончания работы программы память, которая выделена на GPU автоматически освобождается и обнуляется

66

Что выведет на экран данная программа?

```
__global__ void kernel(int *A, int *B, int *C){
    int tid = threadIdx.x + blockIdx.x * blockDim.x;
    C[tid] = A[tid] + B[tid];
}
int main(){
    int n = 512;
    int *A, *B, *C;
    A = (int *) malloc (n * n * sizeof(int));
    B = (int *) malloc (n * n * sizeof(int));
    C = (int *) malloc (n * n * sizeof(int));
    kernel <<<n, n>>>(A, B, C);
    cudaDeviceSynchronize();
    printf("%s\n", cudaGetErrorString(cudaGetLastError()));
    return 0;
}
```

Выберите один из 4 вариантов ответа:

1)		invalid configuration argument
2)		no error
3)		uninitialized block of memory
4)		an illegal memory access was encountered

67

Укажите максимальный объем (в килобайтах на один SM) разделяемой памяти для устройств с Compute Capability 3.5

Выберите один из 4 вариантов ответа:

1)		48
2)		16
3)		8
4)		32

68

Укажите необходимые приставки (спецификаторы) для объявления динамического массива в разделяемой памяти:

Выберите один из 6 вариантов ответа:

1)		__device__ и volatile
2)		__device__ и extern
3)		__shared__ и extern
4)		__device__ и static
5)		__shared__ и static
6)		__shared__ и volatile

69

С какой структурой разделяемая память делит пространство?

Выберите один из 7 вариантов ответа:

1)		Глобальная память
2)		L2 кэш
3)		L1 кэш
4)		Read-only кэш
5)		Константная память
6)		Локальная память
7)		Регистровый файл

70

Укажите на ошибку в программе, которая использует это ядро (подразумевается, что размер массива *a* равен n*512, а размер массива *b* равен n):

```
#define BLOCK_SIZE 512
__global__ void kernel(double *a, double *b){
  __shared__ double buf[BLOCK_SIZE];
double sum = 0.0;
buf[threadIdx.x] = a[threadIdx.x + blockIdx.x*blockDim.x];
if (threadIdx.x == 0){
for (int i = 0; i < BLOCK_SIZE; i++)
sum += buf[i];
}
  syncthreads();
if (threadIdx.x == 0)
b[blockIdx.x] = sum;
}
```

Выберите один из 7 вариантов ответа:

1)		Нельзя выделять статические массивы в разделяемой памяти
2)		Нет синхронизации между первым “if (...)” и “for (...)”
3)		Неверная адресация при считывании данных из массива *a*
4)		syncthreads() должна быть помещена внтурь первого “if(...)”
5)		Нет синхронизации между “buf[..] = …” и первым “if (...)”
6)		Ошибок нет
7)		Неверная адресация при записи данных в массив *b*

71

В реализации умножения матриц, приведенной в видео-лекции, используются двумерные массивы в разделяемой памяти. Это происходит, поскольку:

Выберите один из 5 вариантов ответа:

1)		Соответствующие одномерные не могут быть выделены в разделяемой памяти
2)		Поскольку использование таких структур проще и они не оказывают большого влияния на производительность

3)		Такие структуры уменьшают время доступа к памяти
4)		Ни одно из приведеных объяснений не является верным
5)		Такие структуры (с учетом размера 16x16 или 32x32) оптимизируют исполнение warp-ов

72

Укажите неверное утверждение о разделяемой памяти:

Выберите один из 5 вариантов ответа:

1)		Имеет смысл использовать разделяемую память, если есть данные, которые необходимы нескольким нитям из блока
2)		В случае, если на потоковом мультипроцессоре исполняются несколько блоков одновременно, объем разделяемой памяти делится между ними
3)		Функция __syncthreads() - единственный доступный примитив синхронизации для устройств с Compute Capability меньше 2.0
4)		Ни одно из перечисленного
5)		При статическом выделении разделяемой памяти в коде функции-ядра (например, и_shared_inta[10]) указатели на эту область памяти являются одинаковыми для всех нитей блока

Тестовые задания на тестирование 2

1

Вычислительный кластер – это
1. Однородная система
2. Гетерогенная система
3. Возможно то или другое

2

Система BBN Butterfly относится к
1. SMP-системе
2. MPP-системе
3. Системе, созданной по NUMA-технологии

3

Система HP Superdome – это
1. SMP-система
2. MPP-система
3. NUMA-система

4

Состав ячейки HP Superdome:
1. Процессоры, банки памяти, контроллер ячейки
2. Процессоры, объединенные сетью Ethernet
3. Процессоры, контроллер ввода/вывода, коммутатор
4. Процессоры, банки памяти, коммутатор

5

«Узкие места» HP Superdome:
1. Неоднородность доступа к данным
2. Конфликты при обращении к памяти
3. Переход в третье измерение

6

Семейство Cray T3D/T3E относится к
1. SMP-системе
2. SIMD-системе
3. MPP-системе
4. Кластерной системе

7

Компьютеры семейства Cray T3D/T3E в максимальной конфигурации объединяют
1. До 500 процессоров
2. До 1000 процессоров
3. Более 2000 процессоров

8

Разновидности функциональных узлов (functional Node) семейства Cray T3D/T3E:
1. Управляющие узлы
2. Узлы операционной системы
3. Вычислительные узлы
4. Интерфейсные узлы

9

Сетевые маршрутизаторы семейства Cray T3D/T3E расположены в узлах
1. Одномерной решетки
2. Двумерной решетки
3. Трехмерной решетки

10

Обобщенная соединительная сеть – это:
1. Сеть, которая выполняет **N!** отображений.
2. Сеть, которая выполняет $\mathbf{N^N}$ отображений.
3. Ни та, ни другая.

11

Полная координатная сеть реализует:
1. $\mathbf{N^N}$ отображений.
2. **N!** отображений.
3. Ни то, ни другое.

12

Число возможных связей между входами и выходами полной координатной сети:
1. **N**. 2. $\mathbf{N^2}$. 3. **N!**.

13

В процессорных матрицах реализуются:
1. Обобщенные соединительные сети.
2. Неполные координатные сети.
3. Полные координатные сети.

14

Инверсией k-бита двоичного представления номера узла реализуется:
1. Перестановка типа полной тасовки.
2. Перестановка с замещением.
3. Перестановка типа «бабочка».
4. Разрядно-реверсивная перестановка.
5. Перестановка со смещением.

15

Циклическим сдвигом влево на 1 разряд двоичного представления номера узла реализуется:
1. Перестановка с замещением.
2. Перестановка типа «бабочка».
3. Перестановка типа полной тасовки.
4. Разрядно-реверсивная перестановка.
5. Перестановка со смещением.

16

Взаимным замещением первого и последнего битов двоичного представления номера узла реализуется:
1. Перестановка с замещением.
2. Перестановка типа полной тасовки.
3. Перестановка типа «бабочка».
4. Разрядно-реверсивная перестановка.
5. Перестановка со смещением.

17

Омега-сеть применяется:
1. В SMP-системах.
2. В MPP-системах.
3. В кластерных системах.

18

В кластерных системах применяются сети коммутации типа:
1. Двумерный тор.
2. Звезда.
3. Двоичный гиперкуб.

19

Метод коммуникации с наибольшим быстродействием:
1. Метод МПС.
2. Метод МПП.
3. Метод МПС-МПП.

20

Конфликты на уровне банков памяти в SMP-системах лучше всего устраняются при включении коммутационной сети:
1. Между процессорами и общей памятью.
2. Между исполнительными и управляющим процессорами.
3. Между общей памятью и процессором управления.

21

Память типа DDR3 в сранении с равночастотной DDR2 имеет:
1. Меньшее напряжение питания.
2. Большее напряжение питания.
3. Меньшую емкость.

4. Большую емкость.
5. Меньшее число логических банков.
6. Большее число логических банков.

22

Реализация механизма когерентности осуществляется отслеживанием запросов по шине, свзывающей:
1. Процессор и **I/O**.
2. Память и **I/O**.
3. Процессор, память и **I/O**.
4. Ни то, ни другое, ни третье.

23

Кэш-память:
1. Ортогональная память.
2. Ассоциативная память.
3. Регистровая память.

24

При использовании алгоритма DASH с каждой строкой кэш в резидентном для нее модуле памяти связаны 3 глобальных состояния:
1. Некэшированная; удаленно-разделенная; удаленно-измененная.
2. Невозможная к использованию; разделяемая; измененная.
3. Некэшированная; разделяемая; измененная.
4. Ни то, ни другое, ни третье.

25

При использовании алгоритма DASH строка кэш может находиться в одном из трех локальных состояний:
1. Некэшированная; удаленно-разделенная; удаленно-измененная.
2. Невозможная к использованию; разделяемая; измененная.
3. Некэшированная; разделяемая; удаленно-измененная.
4. Ни то, ни другое, ни третье.

26

Основное назначение RAID-массива:
1. Увеличение объема дискового пространства.
2. Повышение надежности и производительности дисковой системы.
3. Обеспечение бесперебойной работы дискового контроллера.

27

RAID 3 обеспечивает:
1. Независимый доступ к данным.
2. Параллельный доступ.
3. Механизм зеркалирования.
4. Код Хэмминга для коррекции одиночных ошибок.
5. Механизм диагностики кратных ошибок.
6. Автоматическое восстановление данных неисправного диска при считывании.
7. Хранения контрольных сумм на одном диске
8. Распределенное хранение контрольных сумм.

28

GPU – это:
1. Сопроцессор для ускорения трехмерной графики.
2. Ускоритель, применяемый в вычислительных устройствах общего назначения.
3. Специализированное устройство с ограниченным набором команд для увеличения скорости расчета текстур.

29

Задача обработки 3D-графики:
1. Синтез в реальном времени изображения из описания сцены.
2. Анализ сцены из описания ее графических примитивов и характера освещения.
3. Ни то, ни другое.

30

Текстура – это:
1. Растровое изображение.
2. Векторное изображение.
3. Изображение, накладываемое на поверхность.

31

Для синтеза трехмерного изображения необходимо:
1. Определить, какие объекты должны присутствовать в сцене.
2. Определить местоположение вершин, которые задают каждый из объектов сцены.
3. Построить грани по заданным вершинам.
4. Заполнить полигоны текстурами.

32

Для современных GPU характерно:
1. Использование скалярных процессоров общего назначения с плавающей точкой.

2. Использование векторных процессоров.
3. Использование тех и других.
4. Использование унифицированных процессоров для реализации вершинных и пиксельных шейдеров.

33

Особенности архитектуры современных GPU:
1. Множество унифицированных процессоров (ядер) разбито на N блоков (узлов) по 16(32) ядер.
2. Каждый блок (узел) представляет собой 2 многоядерных шейдерных процессора.
3. Каждый блок – это MPP-узел.
4. Каждый блок – это SMP-узел со своей локальной памятью.

34

Подход GPGPU:
1. Построение специализированных устройств трехмерной графики.
2. Построение мощных вычислительных устройств общего назначения.
3. Построение персональных суперкомпьютеров.
4. Построение гибридных кластеров.

35

Эффективность подхода GPGPU связана:
1. С применением технологии параллельных вычислений NVIDIA CUDA.
2. С автоматическим преобразованием последовательных программ в параллельные.
3. С совершенствованием компиляторов.

36

Архитектура GPU-FERMI
1. SMP
2. Мультикластер
3. Квазикластер

37

Архитектура потоковых мультипроцессоров GPU-FERMI
1. Кластерная
2. SMP
3. MPP

38

Архитектура синергетических процессоров в составе CELL
1. RISC
2. Кластерная
3. SMP

39

Когерентность данных в процессоре CELL
1. Поддерживается автоматически
2. Обеспечивается пользователем
3. Обеспечивается администратором системы

40

Шина EIB процессора CELL
1. Одно кольцо шириной 32 бит
2. Два концентрическиз кольца шириной 64 бит на кольцо
3. Четыре концентрическиз кольца шириной 128 бит на кольцо

41

Что выведет на экран данная программа?

```
__global__ void kernel(int *a) {
  extern __shared__ int buf1[];
  extern __shared__ int buf2[];
  buf1[0]=0; buf2[0]=2;
    __syncthreads();
  a[threadIdx.x]=buf1[0]+buf2[0];
}
int main() {
  int *ha, *da;
  ha=(int *)malloc(32*sizeof(int));
  cudaMalloc(&da, 32*sizeof(int));
  kernel <<<1, 32, 4*sizeof(int)>>>(da);
  cudaMemcpy(ha, da, 32*sizeof(int), cudaMemcpyDeviceToHost);
```

```
  printf("%d\n", ha[31]);
  return 0;
}
```

Выберите один из 5 вариантов ответа:

1)		2
2)		64
3)		0
4)		4
5)		an illegal memory access was encountered

42

Что выведет на экран данная программа?

```
 _global__ void kernel(int *a)
{
    shared   int buf1[2];
    shared   int buf2[2];
  buf1[0]=0; buf2[0]=2;
    syncthreads();
  a[threadIdx.x]=buf1[0]+buf2[0];
}
int main(){
  int *ha, *da;
  ha=(int *)malloc(32*sizeof(int));
  cudaMalloc(&da, 32*sizeof(int));
  kernel <<<1, 32>>>(da);
  cudaMemcpy(ha, da, 32*sizeof(int), cudaMemcpyDeviceToHost);
  printf("%d\n", ha[31]);
  return 0;
}
```

Выберите один из 5 вариантов ответа:

1)		64
2)		2
3)		0
4)		an illegal memory access was encountered
5)		4

43

Выберете верное утверждение

Выберите один из 5 вариантов ответа:

1)		Все библиотеки с поддержкой CUDA разрабатываются при участии NVIDIA
2)		CUDA toolkit не включает в себя библиотеки
3)		Использование библиотек позволяет сократить время работы программы
4)		С помощью CUBLAS можно перемножать матрицы как на CPU, так и на GPU
5)		Случайные числа с помощью CURAND можно генерировать как на CPU, так и на GPU

44

Укажите, какой тип распределения не может быть использован при генерации случайных чисел в CURAND:

Выберите один из 5 вариантов ответа:

1)		Распределение Бернулли
2)		Нормальное распределение
3)		Распределение Пуассона
4)		Равномерное распределение
5)		Все могут

45

Укажите, какой тип данных возвращают функции генерации случайных чисел в CURAND:

Выберите один из 6 вариантов ответа:

1)		curandError_t
2)		cudaError
3)		curandStatus_t
4)		cudaStatus_t
5)		cudaError_t
6)		curandError

46

Выберете неверное утверждение:

Выберите один из 5 вариантов ответа:

1)		Функции CUBLAS работают только с памятью GPU, выделенной cublasAlloc()
2)		В библиотеке CUBLAS есть реализация функций библиотеки BLAS на GPU
3)		Многомерные структуры данных в CUBLAS хранятся по столбцам
4)		cublasSetMatrix() копирует данные с хоста на GPU
5)		CUBLAS включает операции «вектор-вектор», «матрица-вектор», «матрица-матрица»

47

Какой функции нехватает для корректной работы?

```
#include <stdlib.h>
#include <stdio.h>
#include "cublas.h"

main ()
{
	double *a, *b, *c;
	double *d_a, *d_b, *d_c;
	int Ida, Idb, Idc,n;

	cudaMallocHost( (void**)&a, Ida * sizeof(double) );
	cudaMallocHost( (void**)&b, Idb * sizeof(double) );
	cudaMallocHost( (void**)&c, Idc * sizeof(double) );

	cublasAlloc(Ida, sizeof(double), (void**)&d a );
	cublasAlloc(Idb, sizeof(double), (void**)&d_b );
	cublasAlloc(Idc, sizeof(double), (void**)&d_c );

	cublasSetMatrix(Ida, Ida, sizeof(double), a, Ida, d a, Ida);
	cublasSetMatrix(Ida, Ida, sizeof(double), b, Idb, d_b, Idb);

	cublasDgemm('N', 'N', n, n, n, 1.0, d_a, Ida, d_b, Idb, 0.0, d_c, Idc);

	cublasGetMatrix( n, n, sizeof(double), d_c, Idc, c, Idc);

	cublasFree( d a);
	cublasFree( d b);
	cublasFree( d_c);

	cudaFreeHost(a);
	cudaFreeHost(b);
	cudaFreeHost(c);

	cublasShutdown();
	return 0;
}
```

Выберите один из 4 вариантов ответа:

1)		cublasInit
2)		cublasStatus
3)		syncthreads

4)		cudaDeviceSynchronize

48

Какая библиотека, поставляющаяся с CUDA Toolkit, предназначена для обработки сигналов?

Выберите один из 5 вариантов ответа:

1)		ArrayFire
2)		Все перечисленные
3)		IPP
4)		OpenCV
5)		NPP

49

Укажите неверное утверждение:

Выберите один из 5 вариантов ответа:

1)		Разреженную матрицу эффективнее представлять в виде специальных индексных форматов
2)		Для корректной работы CUSPARSE необходимо проинициализировать дескриптор
3)		Все верны
4)		CUSPARSE работает с разреженными матрицами
5)		CUSPARSE работает с комплексными числами

50

Укажите, какая библиотека не является разработкой сотрудников NVIDIA

Выберите один из 5 вариантов ответа:

1)		Thrust
2)		CUBLAS
3)		CULA
4)		CUDA Math Library
5)		CURAND

51

Выберете верное утверждение

Выберите один из 4 вариантов ответа:

1)		Для использования Thrust необходимо скомпоновать (слинковать) приложение с thrust
2)		Thrust реализует весь функционал библиотеки STL
3)		Thrust входит в состав CUDA Toolkit
4)		Thrust - проект с закрытым кодом

52

Сколько байт будет передано по шине PCI?
thrust::host_vector<int> h_vec(1024);
thrust::generate(h_vec.begin(), h_vec.end(), rand);
thrust::device_vector<int> d_vec = h_vec;

Выберите один из 4 вариантов ответа:

1)		2048
2)		1024
3)		8192
4)		4096

53

Укажите значение Z[2] после выполнения кода:
thrust::device_vector<float> X(3), Y(3), Z(3);
X[0] = 10; X[1] = 20; X[2] = 30;

Y[0] = 15; Y[1] = 35; Y[2] = 10;
thrust::transform (X.begin(), X.end(), Y.begin(), X.begin(), _1 + _2);

Выберите один из 5 вариантов ответа:

1)		55
2)		40
3)		120
4)		0
5)		25

54

Выберете неверное утверждение:

Выберите один из 5 вариантов ответа:

1)		Thrust обладает меньшими возможностями по сравнению с CUDA C
2)		Thrust является аналогом библиотеки STL
3)		Библиотеку Thrust не нужно компилировать отдельно от вашей программы
4)		Алгоритмы Thrust невозможно запустить из kernel-функций
5)		Алгоритмы Thrust выполняются только на GPU

55

Какое число выведется на экран?
thrust::device_ptr<int> d_ptr = thrust::device_malloc<int>(10);
thrust::sequence(d_ptr, d_ptr + 5);
std::cout << thrust::reduce(d_ptr, d_ptr + 3) << "\n";

Выберите один из 5 вариантов ответа:

1)		5
2)		6
3)		4
4)		3
5)		2

56

Чему будет равно A[0] после выполнения кода?
thrust::host_vector<int> A(4);
A[0] = 20; A[1] = 14;
A[2] = 38; A[3] = 46;
thrust::device_vector<int> dA(4);
dA = A;
sort(A.begin(), A.end());
A = dA;

Выберите один из 4 вариантов ответа:

1)		14
2)		20
3)		46
4)		38

57

Чему будет равен init?

```
int main(void) {
  thrust::device_vector<float> X(3);
  X[0] = 10; X[1] = 30; X[2] = 20;
  float init = 0.0f;
  float result = thrust::reduce(X.begin(), X.end(), init, thrust::maximum<float>());
  std::cout << "maximum is " << result << "\n";
  std::cout << "init is " << init << "\n";
  return 0;
```

```
}
```

Выберите один из 5 вариантов ответа:

1)		0
2)		20
3)		10
4)		2
5)		30

58

Что выведется на экран?

```
__global__ void kernel(int* Arr)
{
   Arr[0] += 2;
}
int main(void)
{
   size_t N = 10;
   int * raw_ptr;
   cudaMalloc((void **) &raw_ptr, N * sizeof(int));
   thrust::device_ptr<int> dev_ptr = thrust::device_pointer_cast(raw_ptr);
   thrust::fill(dev_ptr, dev_ptr + N, (int) 1);
   kernel <<<1,1>>> (raw_ptr);
   std::cout << dev_ptr[0]--;
   cudaFree(raw_ptr);
   return 0;
}
```

Выберите один из 4 вариантов ответа:

1)		1
2)		4
3)		3
4)		2

59

Какое максимальное количество блоков может исполняться конкурентно в одном SMX чипа GK110?

Выберите один из 4 вариантов ответа:

1)		32
2)		8
3)		16
4)		4

60

Какое максимальное количество блоков может конкурентно исполнять SMX чипа GK110 для следующей функции ядра (критерий – объем разделяемой памяти):

```
#define BLOCK SIZE 32
myKernelCUDA(float *C, float *A)
{
    int bx = blockIdx.x;
    int by = blockIdx.y;
    int tx = threadIdx.x;
    int ty = threadIdx.y;
      shared   double As[BLOCK SIZE][BLOCK SIZE];
    As[tx][ty]=A[bx*blockDim.x+tx][by*blockDim.y+ty];
   __synchtreads();
   C[bx*blockDim.x+tx][by*blockDim.y+ty] =As[tx][ty]*8+15;
}
```

Выберите один из 6 вариантов ответа:

1)		8
2)		16

3)		4
4)		3
5)		6
6)		12

61

Какое количество 32-битных регистров может использовать каждая нить блока, если в блоке 512 нитей, и конкурентно исполняются 4 блока, при условии что нет спиллинга регистров (для чипа GK110)?

Выберите один из 5 вариантов ответа:

1)		21
2)		16
3)		255
4)		63
5)		32

62

Какое максимальное количество варпов может конкурентно исполняться на одном SM в архитектуре Fermi (CC 2.0)?

Выберите один из 4 вариантов ответа:

1)		48
2)		64
3)		32
4)		96

63

В случае, если нет ограничения по регистрам и разделяемой памяти, какой размер блока позволит получить 100% device occupancy для устройств на архитектуре Fermi(CC 2.0)?

Выберите один из 4 вариантов ответа:

1)		16 * 32
2)		256 * 4
3)		32 * 32
4)		16 * 8

64

В случае, если нет ограничения по регистрам и разделяемой памяти, какой размер блока не позволит получить 100% device occupancy для устройств на архитектуре Kepler (CC 3.5)?

Выберите один из 4 вариантов ответа:

1)		48 * 32
2)		16 * 8
3)		32 * 32
4)		256 * 4

65

Как будет выглядеть директива *parallel* в языке C с атрибутами, говорящими о размере сетки в 64 блока и блока в 4

Выберите один из 4 вариантов ответа:

1)		#pragma acc parallel gang (64) vector(4)
2)		#pragma acc parallel gang (64) worker (4)
3)		#pragma acc parallel vector (64) gang (4)
4)		#pragma acc parallel worker (64) gang (4)

66

Какую директиву необходимо использовать, чтобы выделить память на GPU для массива **A** без передачи данных?		
Выберите один из 4 вариантов ответа:		
1)		#pragma acc data present (A[:])
2)		#pragma acc data create (A[:])
3)		#pragma acc data copy (A[:])
4)		#pragma acc data private (A[:])

67

Укажите неверное утверждение об OpenACC.		
Выберите один из 4 вариантов ответа:		
1)		Хост исполняет большую часть кода и только в нужных случаях инициализирует работу части кода на ускори-
2)		OpenACC – закрытый стандарт, разработанный NVIDIA для упрощения языка CUDA
3)		В OpenACC атрибуты директив можно условно разделить на основные атрибуты и атрибуты данных
4)		Модель исполнения имеет три уровня: gang, worker и vector

---

68

Директива *#pragma acc loop independent collapse (2)* вызовет:		
Выберите один из 4 вариантов ответа:		
1)		Параллельное исполнение цикла с размером сетки – 2 блока
2)		Последовательное исполнение 2-х вложенных циклов
3)		Параллельное исполнение следующего цикла с использованием 2-х функций-ядер
4)		Параллельное исполнение цикла первого уровня и вложенного цикла

69

Какие ограничения имеют директивы *parallel* и *kernels* в спецификации 1.0?		
Выберите один из 4 вариантов ответа:		
1)		Не могут включать в себя вложенные регионы *parallel* или *kernels*
2)		Не могут иметь заданный пользователем размер сетки
3)		Не могут включать в себя операции с двумерными массивами
4)		Не могут включать в себя циклы *for*

70

Выберите оптимальный вариант атрибутов директивы *parallel* для следующего кода (с точки зрения минимизации объема копируемых данных и корректности работы кода)

```
#pragma acc parallel
for (int i = 0; i < n; i++)
  a[i] = b[i] + i * 3 + 3,5;
```

Выберите один из 4 вариантов ответа:		
1)		#pragma acc parallel copy (a[n],b[n])
2)		#pragma acc parallel copyin (b[n]), copyout (a[n])
3)		#pragma acc parallel create (b[n]), copyout (a[n])
4)		#pragma acc parallel create (b[n]), copy (a[n])

Оценка практических умений и навыков:

Пример типовой задачи.

При сложении чисел в ассоциативном параллельном процессоре оба слагаемых располагатся в одной строке ассоциативной памяти. Пусть слагаемое А расположено в разрядах 1-10, слагаемое – в разрядах 21-30. Результат сохраняется в разрядах 21-30.

N вар. (дейст.)	Нач. СВА	Рез. С В	Комментарий
0	0 0 0	0 0	Нет действий
1(2)	0 0 1	0 1	Делать после 3
2	0 1 0	0 1	Нет действий
3 (1)	0 1 1	1 0	Делать перед 1
4 (3)	1 0 0	0 1	Делать перед 6
5	1 0 1	1 0	Нет действий
6 (4)	1 1 0	1 0	Делать после 4
7	1 1 1	1 1	Нет действий

Один разряд каждой строки отведем под флажок переноса C_i. При поразрядном сложении возможны 8 вариантов (см. рис.). Но проводить опросы (действия) надо только для четырех из них в порядке, который указан в скобках. Почему?

6.2. Контрольные вопросы / вопросы экзамена

1. Необходимость, ретроспектива и тенденции развития параллельных систем
   Понятие СуперЭВМ. Сфера применений ВВС. Основа повышения производительности ВС. Этапы численного эксперимента. Введение параллелизма в архитектуру ЭВМ. Ретроспектива развития параллельных систем. Современные тенденции.
2. Систематика параллельных систем
   Задачи систематики. Систематика Флина. Понятие структурной систематики.
3. Абстрактные оценки производительности
   Пиковая производительность. Длина полупроизводительности. Экспериментальное определение параметров.
4. Системная производительность
   Основные недостатки абстрактной оценки. Закон Амдала. Концепция маршрутизации. Пример каскадного суммирования. Относительная оценка производительности. Тестовые оценки производительности.
5. Обработка изображений
   Основные понятия. Виды обработки. Улучшение изображений. Кодирование изображений и обработка графики.
6. Обработка символов
   Обработка цепочек символов. Обработка естественных языков. Поиск в словаре и обработка морфем. Синтаксическая обработка. Правила CFG.
7. Понятие ассоциативного параллельного процессора
   Ассоциативная память. Память отклика. Команды АПП. Логические алгоритмы для АПП.
8. Арифметические алгоритмы для АПП. Ортогональная память
   Усложнение памяти отклика. Дополнительные микрокоманды. Алгоритм добавления единицы. Алгоритм сложения полей. Причины громоздкости АПП. Организация ортогональной памяти.
9. Понятие процессорных матриц
   Ассоциация понятий ОКМД-систем, процессорных матриц и параллельных процессоров в строгом смысле. Физические ограничения на процессорные элементы. Потенциальные возможности ОКМД-систем. Закон Гроша и гипотеза Минского. Альтернатива СВО-СГО.
10. Системы вертикальной обработки
    Обобщенная структура. Пример ICL DAP. Метод доступа. Процессорный элемент. Разрядная зависимость производительности. Акселераторы. Внешние устройства. Современные СВО.
11. Матричный процессор ассоциативного типа
    Регистровая структура. Процессорная матрица и ее обрамление. Процессорный элемент. Скалярный процессор управления. Характеристика состава команд. Пример организации входного потока ПМА. Оценка производительности ПМА.
12. Алгоритм сортировки для ПМА
    Первый проход. Второй проход. Третий проход. Оценка числа обращений к памяти.
13. Мэйнфреймовые архитектуры
    Специфика программного обеспечения. Рассматриваемые архитектуры. Системы Cm и BBN Butterfly. Система HP Superdome. Семейство CRAY T3D/T3E.*
14. Кластерные архитектуры
    Понятие кластера. Однородность и гетерогенность. Два подхода к организации. Первые проекты. Beowulf-технологии. Технологии соединения компьютеров в кластер. Задачи кластерных систем. Вопросы разработки параллельных программ.
15. Элементы теории коммутационных сетей
    Понятие отображений, соединительных и координатных сетей. Базовые перестановки. Коммутации в процессорных матрицах. Оценки эффективности переключателей.
16. Сети коммутации в мэйнфреймах и кластерах
    Основные типы сетей. Временные параметры сетевых передач. Методы МПС и МПП.
17. Организация главной памяти
    Случай ОКМД-систем. Исключение коллизий. Подход фирмы Burroughs. Память типа DDR-SDRAM.
18. Когерентность по данным
    Позитивизм многоуровневой организации памяти. Кэш-память. Понятие когерентности для данных.
19. Реализация неявных алгоритмов когерентности в MPP-системах
    Случай распределенной памяти. Алгоритм DASH.
20. RAID-массивы
    Понятие RAID. Параллельный и независимый доступы. Уровни RAID. Формирование контрольных слов. RAID-контроллеры.

21. Классика графических процессоров
    Понятие GPU и текстуры. Задача синтеза 3D-изображения. Конвейеризация. Недостатки классического подхода. Проблема балансировки нагрузки.
22. Архитектура современных графических процессоров
    Унифицированный шейдерный процессор как скалярный прцессор общего назначения. Эффективность перехода в GPU от векторных к скалярным вычислениям. Подход GPGPU. Архитектура FERMI.
23. Суперпроцессор Cell
    Вехи истории. Текущая версия Cell. Высокопроизводительные системы на Cell. Кластер RoadRunner. Архитектура CBEA.
24. Разработка параллельных программ с использованием MPI.
    Инициализация и завершение MPI программ. Определение количества и ранга процессов. Передача сообщений. Прием сообщений. Определение времени выполнение MPI-программы.
25. Коллективные операции передачи данных в MPI.
    Передача данных от одного процесса всем процессам программы. Передача данных от всех процессов одному процессу. Операции редукции. Синхронизация вычислений.
26. Производные типы данных в MPI.
    Понятие производного типа данных. Способы конструирования производных типов данных. Объявление производных типов и их удаление.
27. Управление группами процессов и коммуникаторами в MPI.
    Управление группами. Управление коммуникаторами.
28. Виртуальные топологии в MPI.
    Декартовы топологии (решетки). Топологии графа.
29. Стандарт MPI-2.
    Динамическое порождение процессов. Одностороннее взаимодействие процессов. Параллельный ввод/вывод. Расширенные коллективные операции.
30. Основные конструкции OpenMP.
    Принципиальная схема программирования в OpenMP. Синтаксис директив в OpenMP. Особенности реализации директив OpenMP.
31. Загрузка и синхронизация в OpenMP.
    Синхронизации типов atomic, critical, barrier, master, ordered, flush.
32. Программирование с использованием Task Parallel Library (TPL).
    Применение идентификатора задачи. Методы ожидания задачи. Класс TaskFactory. Применение лямбда-выражения в качестве задачи. Создание продолжения задачи. Возврат значения из задачи. Отмена задачи. Отмена задачи. Методы For() и ForEach().
33. Модель исполнения CUDA.
    SIMT: виртуальные нити, блоки. Глобальная синхронизация. Ветвление (branching). Распределение по warp-ам. Масштабирование.
34. Иерархия памяти в CUDA.
    Глобальная, локальная, регистровая, разделяемая память.
35. Прикладные CUDA библиотеки.
    Библиотеки NVIDIA. Сторонние библиотеки. CURAND, CUBLAS, CUSPASE, CUFFT, NPP, ArrayFire.
36. Библиотека Thrust.
    Основные возможности. Компоненты. Распространение. Класс-функтор. Типы трансформаций.
37. Оптимизация CUDA программ.
    Конфигурация сетки. Занятость устройства. Размеры блоков. Occupancy calculator.
38. Стандарт директивного программирования OpenACC.
    Модель исполнения OpenACC. Синтаксис дирректив. Конструкция Parallel. Атрибуты конструкции Parallel. Конструкция Kernels. Конструкция Loop. Использование региона Data.
39. Запросы Parallel LINQ.
    Предохранение порядка результатов. Принудительное параллельное выполнение. Ограничение степени параллелизма. Обработка исключений.
40. Линейная алгебра с плотными матрицами в CUDA.
    Инициализация и освобождение контекста. Работа с памятью. Потокобезопасность. Функции уровней 1,2,3.
41. Линейная алгебра с разреженными матрицами в CUDA.
    Индексирование и форматы данных. Функции уровней 1,2,3.
42. Преобразование Фурье в CUDA.
    Типы преобразований. Многомерные преобразования. Потокобезопасность.
43. Генерация псевдослучайных чисел в CUDA.
    Типы генераторов. Опции генераторов. Пример использования cuRAND.

44. Основные концепции и понятия профилирования с применением пакета Intel Parallel Studio.
Понятие критического пути. Состояния потоков. Понятие категорий времени.
45. Проблемы производительности, определяемые при помощи пакета профилирования Intel Parallel Studio.
Распределение вычислительной нагрузки. Синхронизация и производительность.
46. Использование Intel Thread Profiler.
Изучение профилируемого приложения. Подготовка приложения для профилирования. Профилирование приложения

Лист регистрации изменений и дополнений

№ п/п	№ раздела внесения изменений	Дата внесения изменений	Содержание изменений	«Согласовано» Заведующий кафедрой, реализующей дисциплину	«Согласовано» Председатель УМК института, в состав которого входит выпускающая кафедра