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

Факультет компьютерных наук

Департамент программной инженерии

Рабочая программа дисциплины

**Архитектура вычислительных систем**

для образовательной программы «Программная инженерия»

направления подготовки 09.03.04 «Программная инженерия»

уровень - бакалавр

Разработчик программы:

Баканов В.М., д.т.н.,профессор, vbakanov@hse.ru

Одобрена на заседании департамента программной инженерии «\_25\_»\_мая\_ 2018 г.

Руководитель департамента Авдошин С.М. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утверждена Академическим советом образовательной программы

«\_25\_»\_мая\_ 2018 г., № протокола\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Академический руководитель образовательной программы

Шилов В.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2018

*Настоящая программа не может быть использована другими подразделениями университета и другими вузами без разрешения подразделения-разработчика программы.*

|  |  |
| --- | --- |
| Автор | Баканов Валерий Михайлович |
| Число кредитов | 4 |
| Контактная работа (час.) | 420 |
| Самостоятельная работа (час.) | 153 |
| Курс | 2 |
| Формат изучения дисциплины | Без использования онлайн курса |

**I. ЦЕЛЬ, РЕЗУЛЬТАТЫ ОСВОЕНИЯ ДИСЦИПЛИНЫ И ПЕРЕКВИЗИТЫ**

Цели освоения дисциплины "Архитектуры вычислительных систем":

* формирование у студентов профессиональных компетенций, связанных с использованием теоретических знаний в области архитектур вычислительных систем;
* ознакомление студентов с прогрессивными парадигмами развития архитектур вычислителей с целью заложить основы для последующих курсов, посвящённых созданию современных информационных систем
* получение практических навыков в области выбора архитектуры вычислительной системы, наилучшим образом раскрывающего потенциальные возможности заданного алгоритма с учётом заданных требований к программному обеспечению;
* развитие умений, основанных на полученных теоретических знаниях, позволяющих на творческом и репродуктивном уровне применять и создавать эффективные алгоритмы для решения задач обработки информации применительно к данной архитектуре вычислительной системы;
* получение студентами навыков самостоятельной исследовательской работы, предполагающей изучение специфических методов анализа архитектур вычислительных систем и функционирующего на них программного обеспечения, инструментов и средств, необходимых для решения актуальной, в аспекте программной инженерии, задачи выбора рациональных алгоритмов в зависимости от особенностей применения разрабатываемых программ.

Настоящая дисциплина относится к базовой части профессионального цикла дисциплин.

В соответствии с рабочим учебным планом по направлению "Программная инженерия" дисциплина "Архитектура вычислительных систем" читается студентам второго курса бакалавриата в 1-ом и 2-ом модулях.

Изучение данной дисциплины базируется на знаниях, полученных студентами при освоении учебных дисциплин "Дискретная математика", "Программирование", общих знаниях математики и основ программирования в части базовых алгоритмических конструкций.

Дисциплина является основой для последующего изучения дисциплин: "Проектирование и архитектура программных систем", научно-исследовательского семинара "Современные архитектуры вычислителей".

**II. СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ**

***Тема 1.***История машинного счёта.

***Тема 2.***Определения понятия "Архитектура" в применении к вычислительным системам.

***Тема 3.***Уровни управления процессом вычислений. Архитектура процессоров x86 фирмы Intel.

***Тема 4.***Общие требования к программному коду. Пото́ковые (DATA-FLOW) вычислители. Машинные команды, язык программирования Ассемблер.

***Тема 5.***Недостаток процесса вычислений в позиционной системе счисления и альтернативные решения. Простейшие программы на Ассемблере - простые типы и описания данных.

***Тема 6.***Архитектура параллельных вычислительных систем. Ассемблер - целочисленная арифметика.

***Тема 7.***Суперкомпьютеры. Ассемблер - организация условных и безусловных переходов.

***Тема 8.***Нейронные сети и нейрокомпьютеры.

***Тема 9.***Ассемблер - практика разработки программ. Интерактивные среды разработки. Макроассемблер. Процессор x86 и программирование под MS DOS. Ассемблер для 32-x и 64-бит архитектур. Ассемблер для процессоров архитектуры ARM. Поддержка операций над числами с плавающей запятой.

***Тема 10.***Транспьютеры. Отладка программ на языке Ассемблер.

***Тема 11.***Метакомпьютинг и концепция GRID. Использование ассемблерных программ при программировании на языках высокого уровня.

***Тема 12.***Архитектура GPU фирмы NVIDIA и технология CUDA.

***Тема 13.***Аналоговые вычислительные системы.

***Тема 14.***Вычислители с программируемой архитектурой, пути усовершенствования архитектуры.

***Тема 15.*** Квантовые вычислители и системы передачи данных.

***Тема 16.*** Архитектура систем поиска информации в сети InterNet.

# III. ОЦЕНИВАНИЕ

# III.1. Формы контроля знаний студентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Тип контроля* | *Форма контроля* | *Модули* | | *Параметры* |
| *1* | *2* |
| Текущий  (неделя) | Самостоятельная (домашняя) работа |  | 5÷7-я недели | Разработка программы на языке Ассемблер (по индивидуальному заданию) |
| Экзамен |  | ■ | Устный экзамен 120 мин. |
| Итоговый |  |  |  |  |

# III.2. Критерии оценки знаний, навыков

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

**Текущий контроль** предусматривает домашнее задание, выполняемое во втором модуле.

***Домашнее задание*** включает разработку, кодирование, тестирование и отладку программ реализации задачи на языке Ассемблера (по индивидуальному заданию), исследование и сравнительный анализ алгоритмов ее решения. По домашнему заданию оформляется отчет в электронном виде.

Домашнее задание размещается в LMS в разделе "Проекты". В установленный срок студент загружает в LMS архив, содержащий полностью оформленный отчёт и программу решения контрольного домашнего задания. Оценка за домашнее задание выставляется с учётом полноты выполнения задания и оформления результатов.

При определении итоговой оценки учитываются:

* работа студентов на практических занятиях, а именно: подготовка сообщений по одной из заданных тем, решение тестов по некоторым темам;
* выполнение домашней работы.

В случае несвоевременной сдачи домашнего задания оценка снижается на один балл за каждый день задержки. При задержке по уважительной причине баллы не снимаются.

**Итоговый контроль**: экзамен в конце 2-го модуля (2 час). Экзамен проводится в устной форме.

# III.3. Порядок формирования оценок по дисциплине

По всем видам работ выставляется 10-балльная оценка. Оценивается работа студентов на практических занятиях Оаудиторная, в которую входят:

* Результаты опросов по текущей теме в устной форме.
* Полнота освещения темы доклада. Доклад готовит каждый студент по предложенным преподавателям темам.

Накопленная оценка Онакопл. за текущий контроль учитывает результаты студента по текущему контролю следующим образом:

Онакопл. = 0,5 × Одом.задание + 0,5 × Оаудиторная,

Способ округления — арифметический.

Результирующая оценка во втором модуле определяется соотношением:

Орезульт. = 0,5 × Онакопл. + 0,5 × Оэкз.

Перевод в пятибалльную оценку осуществляется в соответствии со следующей таблицей.

**Таблица соответствия оценок по десятибалльной и пятибалльной системам**

|  |  |
| --- | --- |
| *По десятибалльной шкале* | *По пятибалльной шкале* |
| 1 – неудовлетворительно  2 – очень плохо  3 – плохо | неудовлетворительно – 2 |
| 4 – удовлетворительно  5 – весьма удовлетворительно | удовлетворительно – 3 |
| 6 – хорошо  7 – очень хорошо | хорошо – 4 |
| 8 – почти отлично  9 – отлично  10 – блестяще | отлично – 5 |

# IV. ПРИМЕРЫ ОЦЕНОЧНЫХ СРЕДСТВ

# IV.1. Тематика заданий текущего контроля

### IV.1.1. Домашнее задание

1. Разработка программы на языке Ассемблер для процессоров архитектуры x86 фирмы Intel.

Примерное задание - по заданному алгоритму требуется**:**

1. Разработать программу на языке ассемблера для нахождения наибольшего общего делителя (НОД, GCD - *Greatest Common Divisor*) двух положительных чисел по методу Эвклида вычитанием. Программу разработать в среде Ассемблер-компилятора FASM с использованием 32-ти битовых регистров и оформить в виде ассемблер-процедуры (исходные данные в регистрах EAX,EBX, результат вернуть через EAX).

**Формат входных и выходных данных:**

1. Для набора текста программы применяется интегрированная среда разработки Ассемблерных программ FasmW (Fasm Editor 2.0, Fresh), компиляция производится компилятором FASM.
2. Команды программы находятся в текстовом файле GCD\_Euclid\_32.asm, скомпилированный бинарный файл (версия 32-бит) - GCD\_Euclid\_32.exe. Бинарный файл исполняется аппаратно 32-битовым процессором Intel.

**Формат выходных данных**

Выходные данные – исполняемый (формат PE – Portable Executable, 32 бит) бинарный файл GCD\_Euclid\_32.exe.

**Пример входных и выходных данных**

|  |
| --- |
| *Входные данные – файл GCD\_EUCLID\_32.ASM* |
| format PE GUI 4.0 ; GUI-приложение для Windows (EXE-файл)  ;  include "c:/emu8086/fasm/include/win32ax.inc" ; путь к подкаталогу /include/  ;  .data ; секция данных  formats db "НОД чисел %d и %d суть %d (десятичные)",0  result db 256 dup(?) ; строка для форматного вывода  ;  A dd 81d ; исходные данные (1-е число для нахождения НОД - Наиб. Общего Делителя)  B dd 54d ; 2-е число для того же  ;  .code ; начало секция кода  start: ; точка старта программы  ;  mov eax,dword[A] ; исходные данные присваиваем регистрам  mov ebx,dword[B]  ;  call GCD\_2S\_Euclid\_32 ; вызов процедуры GCD\_2S\_Euclid\_32 (поиск НОД двух чисел)  ;  invoke wsprintf,result,formats, [A],[B],eax ; форматирование строки для вывода  add esp,20 ; восстановим стек (соглашение о вызовах сdecl)  ;  invoke MessageBox,0,result,"Данные расчёта",MB\_OK ; EAX ← код нажатой кнопки  invoke ExitProcess,0 ; возврат в Windows c кодом возврата 0  ;  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  proc GCD\_2S\_Euclid\_32 ; USES EAX,EBX; RESULT IN EAX  ; фактически используется fastcall-соглашение (передача формальных  ; параметров не посредством стека, а через регистры процессора  ; ищет GCD от EAX,EBX по Эвклиду (при EAX==0 || EBX==0 возвращается 0)  ; GCD is abbreviation of 'Greatest Common Divisor'  ; эта процедура изменяет только регистры EAX и EBX  ;  ; while(EAX!=0 && EBX!=0) { if (EAX>EBX) EAX-=EBX else EBX-=EAX; } return EAX+EBX;  ;  cmp eax,0 ; сравниваем EAX с 0  jne ax\_jne ; переход, если EAX!=0  ret ; возвращаем EAX=0 как результат  ax\_jne: ; вариант EAX!=0  ; cmp eax,0 ; сравниваем EAX с 0 (повторять сравнение не надо)  jge ax\_jge ; переход, если EAX>=0  neg eax ; изменили знак EAX (получили модуль EAX)  ax\_jge: ; вариант EAX>=0  ;  cmp ebx,0 ; сравниваем EBX с 0  jne bx\_jne ; переход, если EAX!=0  mov eax,ebx ; готовим возврат EAX ← 0  ret ; возвращаем EAX=0 (result)  bx\_jne: ; вариант EBX!=0  ; cmp ebx,0 ; сравниваем EBX с 0 (повторять сравнение не надо)  jge bx\_jge ; переход, если EBX>=0  neg ebx ; изменили знак EBX (получили модуль EBX)  bx\_jge: ; вариант EAX>=0  ;  cycle: ; начинаем основной цикл метода Эвклида ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  cmp eax,ebx ; сравним EAX и EBX  je ended ; при EAX==EBX (ZF==1) выход с возвратом EAX  ;  jl metka ; при EAX<EBX (SF#OF) переход на metka  sub eax,ebx ; EAX ← EAX-EBX  jmp cycle ; безусловный переход на cycle  metka:  sub ebx,eax ; EBX ← EBX-EAX  jmp cycle ; безусловный переход на cycle  ended: ; EAX ← Result ( GCD )  ;  ret ; возврат из процедуры GCD\_2S\_Euclid\_32  ;  endp ; конец процедуры GCD\_2S\_Euclid\_32  .end start ; конец программы |
| *Выходные данные* |
| При входных данных EAX=81**10** и EBX=54**10** после выполнения процедуры GCD\_Euclid\_32 правильный ответ EAX=1B**16**=27**10** |

**Оценивание домашней работы**

1. Верно выполняется тестовый пример, приведённый выше - 3 балла.
2. Отсутствуют синтаксические ошибки в заданном примере - 4 балла.
3. Заданный пример выполняется не на всех входных данных - (+ до 2 баллов) (6-7 баллов).
4. Заданный пример выполняется на всех входных данных (+ до 2 баллов) (8-9 баллов).

# IV.1.2. Вопросы для оценки качества освоения дисциплины

1. История автоматизированных вычислений. Первые появления понятий “структура” и “архитектура” вычислительных систем.
2. Архитектура вычислительной системы Ч.Беббеджа. Роль Ады Лавлейс в её проектировании и программировании.
3. Понятие “архитектура” в применении к вычислительным системам.
4. Машина Тьюринга. Её архитектура, связь с архитектурой современных вычислительных системам.
5. Язык программирования Ассемблер - история, область применения.
6. Выбор основания системы счисления. “Двои́чные” и “трои́чные” вычислители.
7. Архитектура вычислительных систем фон Неймана. Пять принципов построения вычислителей, достоинства и недостатки “принстонской” и “гарвардской” архитектур.
8. Архитектуры SMP и MPP. Преимущества и недостатки этих архитектур, области применения.
9. Особенности разработки программ в архитектуре SMP – достоинства и недостатки.
10. Особенности разработки программ в архитектуре MPP – достоинства и недостатки.
11. Классификация архитектур по М.Флинну. Параметры классификации, примеры реализации архитектур.
12. Повышение быстродействия вычислительных систем с помощью совершенствования их архитектуры. Конвейерный принцип вычислений.
13. Повышение быстродействия вычислительных систем с помощью совершенствования их архитектуры. Векторные вычислители.
14. Повышение быстродействия вычислительных систем с помощью совершенствования их архитектуры. Вычислители со сверхдлинным командным словом (VLIW).
15. Архитектура систем команд вычислителей. А́дресность команд, CISC и RISC – системы команд.
16. Арифметические операции с числами в позиционной системе счисления. Недостаток этого метода, альтернативы.
17. Упреждающая загрузка и спекулятивное выполнение команд в современных процессорах. Проблемы и решения этого подхода.
18. Основные требования к программному коду. Особенности выполнение его в вычислителях классической фон Неймановской архитектуры и потоковых (DATA-FLOW) вычислителях.
19. Понятие ярусно-параллельной формы (ЯПФ) информационного графа алгоритма и условия готовности операций к выполнению.
20. Архитектура потокового (DATA-FLOW) вычислителя. Проблемы реализации такой вычислительной системы.
21. Понятие гра́нулы (*зерна, блока*) параллелизма. Размер гранул параллелизма в вычислителях различной архитектуры.
22. Архитектура суперкомпьютеров. Задачи, требующие использования супер-ЭВМ; основные параметры супер-ЭВМ.
23. Группы инструкций процессора. Использование ими регистров процессора.
24. Недостатки выполнения арифметических действий на арифметико-логических современной архитектуры, потенциал использования непозиционных систем счисления. Вычислители на основе СОК (системы оста́точных классов), их достоинства и недостатки, реализация, перспективы.
25. Метод проведения математических вычислений на графических процессорах (GPU). Архитектура GPU, конкретные модели, основные параметры, область эффективного применения.
26. Принципы разработки программ для использования в технологии CUDA. Понятия “хоста” (*host*) и “девайса” (*device*), приёмы программирования, используемые ср*е*ды создания приложений для архитектуры CUDA.
27. Использование устройств архитектуры CUDA в современных суперкластерах. Раздел**е**ние задач по эффективности решения на GPU.
28. особенности архитектуры и системы инструкций процессоров ARM.
29. Архитектура вычислительных систем на основе нейронных сетей. Класс реш**ае**мых задач, процесс обучение нейронной сети, метод обратного распространения ошибки.
30. Ана́логовые вычислители – архитектура, составные элементы аналоговых вычислителей, область применения, достоинства и недостатки. Гибридные вычислительные системы.
31. Архитектура вычислителей на основе транспьютеров. Понятие транспьютера, история разработки, потенциал агрегации транспьютеров, современное состояние.
32. Архитектура вычислителей на основе систол*и*ческих матриц. Принцип обработки данных, реализации, области применения.
33. Стековая машина, связь с обратной польской нотацией записи выражений. Достоинства и недостатки стековой машины, примеры реализации.
34. Сравнение архитектур SIMD и SMP (на примере GPU и CPU). Достигнутые вычислительные мощности, стоимость, области эффективного применения, перспективы.
35. Архитектура поиск**о**вых систем в сети InterNet. Принципы поиска информации, формирование ключевых слов, аппаратная реализация поисковых машин.
36. Недостатки архитектур современных вычислителей, направления их совершенствования.
37. Архитектура математического сопроцессора плавающей точки i87 - кольцевой стек, точность представления чисел, принцип разделения команд между центральным процессором и сопроцессором.
38. Квантовые вычислители и сети передачи данных. Основные принципы действия, потенциальные возможности, известные реализации отдельных компонентов.
39. Архитектуры вычислителей с наличием регистра-счётчика команд и без **о**ного. Примеры, достоинства и недостатки.
40. Архитектура метакомпьютера. Отличия метакомпьютера от обычного компьютера, гетерогенность метакомпьютера, реальные системы метакомпьютинга.
41. Архитектура “облачных вычислений” (*cloud computing*). Понятие облачного сервиса, прозрачности и гибкости изменения потребляемых клиентом ресурсов, критика полезности и безопасности технологии облачных вычислений.
42. Взаимодействие параллельных процессов, известные методы программной синхронизации. Аппаратная синхронизация.
43. Принципы управление последовательностью выполнения процессорных инструкций в вычислителях традиционной архитектуры и потоковой (DATA-FLOW) архитектуры.
44. Ограничения производительности вычислителей многопроцессорных архитектур – закон Амдаля, сетевой закон Амдаля. Формулировка, исходные предпосылки, применимость.
45. Особенности использования внутренних регистров процессора командами систем CISC и RISC.
46. Организация и использование стека в процессорах x86. Пролог и эпилог в процедурах на ассемблере.
47. CISC и RISC системы команд. Основные отличия, особенности декодирования и исполнения, преимущества.
48. Абстрактные модели параллельных вычислений. Теорема Брента.
49. Конвейерная архитектура как средство повышения производительности вычислителя. Условия эффективной работы конвейера.
50. Причины необходимости следования концепции однократного присваивания в вычислителях потоковой (DATA-FLOW) архитектуры.
51. Микропрограммная архитектура процессоров x86. Достоинства и недостатки микропрограммной архитектуры.
52. Архитектуры микропроцессоров с предикатной системой выполнения машинных инструкций. Микропроцессоры архитектуры ARM и Intel Itanium.

# V. РЕСУРСЫ

# V.1. Основная литература:

1. Э.Таненбаум, Т.Остин. Архитектура компьютера (издание 6). — СПб.: Питер, 2014. — 811 с.

# V.2. Дополнительная литература

1. Пирогов В.Ю. Ассемблер и дизассемблирование / В.Ю. Пирогов. — СПб.: БХВ-Петербург, 2007. — 464 с.
2. Барский А.Б. Логические нейронные сети: учеб. пособие / А.Б.Барский. — М.: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2007. — 352 с.
3. Барский А.Б. Параллельные информационные технологии: учеб. пособие / А.Б.Барский. — М.: Интернет-Университет Информационных Технологий, 2007. — 503 с.
4. Сандерс, Дж. Технология CUDA в примерах: введение в программирование графических процессоров / Дж.Сандерс, Э.Кэндрот; предисл. Дж.Донгарра; пер. с англ. А.А.Слинкина; науч. ред. А.В.Боресков. — М.: ДМК Пресс, 2011. — 231 с.
5. Боресков А.В. Параллельные вычисления на GPU: архитектура и программная модель CUDA: учеб. пособие для вузов. / А.В.Боресков, А.А.Харламов, Н.Д.Марковский и др. — М.: Изд-во МГУ, 2012. — 333 с.

# V.3. Программное обеспечение

|  |  |  |
| --- | --- | --- |
| *№ п/п* | *Наименование* | *Условия доступа* |
| 1 | Microsoft Windows 7 Professional RUS  Microsoft Windows 10  Microsoft Windows 8.1 Professional RUS | *Из внутренней сети университета (договор)* |
| 2 | Microsoft Office Professional Plus 2010 | *Из внутренней сети университета (договор)* |
| 3 | FASM | URL: <https://flatassembler.net/> (доступ свободный) |
| 4 | FasmW | URL: <https://flatassembler.net/fasmw17306.zip> (доступ свободный) |
| 5 | Fresh IDE | URL: <https://fresh.flatassembler.net/> (доступ свободный) |
| 6 | Fasm Editor 2.0 | URL: <http://asmworld.ru/instrumenty/fasm-editor-2-0/> (доступ свободный) |

**V.4. Профессиональные базы данных, информационные справочные системы, интернет-ресурсы (электронные образовательные ресурсы)**

|  |  |  |
| --- | --- | --- |
| *№ п/п* | *Наименование* | *Условия доступа* |
|  | ***Интернет-ресурсы (электронные образовательные ресурсы)*** | |
| 1 | Портал AsmWorld | URL: <http://asmworld.ru/> (доступ свободный) |
| 2 | NVIDIA: архитектура графических карт и технология CUDA | URL: <http://www.nvidia.ru>/ (доступ свободный) |

**V.5. Материально-техническое обеспечение дисциплины**

Учебные аудитории для лекционных занятий по дисциплине обеспечивают использование и демонстрацию тематических иллюстраций, соответствующих программе дисциплины в составе:

ПЭВМ с доступом в Интернет (операционная система, офисные программы, антивирусные программы);

мультимедийный проектор с дистанционным управлением.

Учебные аудитории для лабораторных и самостоятельных занятий по дисциплине оснащены программным обеспечением Microsoft Visual Studio 2015, компилятор FASM, средой разработки ассемблерных программ FasmW, Fasm Editor 2.0 ­­­­­­­­­­­­­­­­­­­­­­­­с возможностью подключения к сети Интернет и доступом к электронной информационно-образовательной среде НИУ ВШЭ.