Общая схема проведения экзамена.

В аудитории одновременно находятся не более 15 студентов. Новые студенты заходят по мере освобождения мест.

Оценка рассчитывается по следующей формуле:

$$G = [P + T_1 + 0{,}5A_1 + T_2 + 0{,}5A_2] - \mathrm{C},$$

где $P \in [0;\ 2]$ – оценка за практическое задание,
$T_i \in [0;\ 1]$ – оценки за теоретические вопросы,
$A_i \in [0; 1]$ – оценки за дополнительные вопросы,
C – сумма штрафных баллов (см. ниже).

Экзамен – устно-письменный. Студенты пишут практическое задание на листке, затем отдают его преподавателю на проверку, после проверки – отвечают теоретические вопросы.

Практическое задание должно быть написано на языке ассемблера. Все локальные переменные должны располагаться на стеке или в регистрах.

Время на подготовку: 1 час с момента получения билета (40 минут, если практическое задание зачтено за выполнение л/р). В ходе подготовки к ответу студенты могут пользоваться *рукописным* конспектом лекций объемом не более 1 листа А4 (можно использовать обе стороны листа).

Непосредственно в ходе ответа использовать что-либо, кроме ручки и листка запрещается. Допустимо, помимо решения практической задачи, записать на листок основные моменты по теоретическим вопросам (полный ответ переписывать не надо: в этом случае преподаватель имеет право забрать листок).

В случае обнаружения недочетов в решении практического задания преподаватель возвращает листок на доработку решения без понижения оценки не более 2 раз при наличии несущественных недочетов, и не более 1 раза – при наличии значительных ошибок. За последующие возвраты назначается по 0,5 штрафных балла.

Для каждого теоретического вопроса преподаватель задает 1 дополнительный вопрос по теме вопроса либо по смежной теме. Первый теоретический вопрос выбирается из модулей 1 и 2, второй вопрос – из модулей 3 и 4.

Билеты распределяются случайным образом. Студент имеет право 1 раз заменить билет на другой случайный билет без понижения оценки. Вторая замена билета ограничивает максимальную оценку 3 баллами

Попытка списывания (неважно, с телефона, шпаргалки или у другого студента) наказывается удалением студента из аудитории. Ответ «просто хотел спросить» - не принимается. Если надо – поднимите руку и спросите у преподавателя.

Модуль 1. Язык ассемблера x86-64. Соглашения о вызовах.

1. Регистры процессора архитектуры x86-64. Части регистров.
2. Инструкции перемещения данных. Методы адресации. Инструкция lea. Стек и инструкции работы с ним.
3. Арифметические операции. Регистр флагов. Кодировка отрицательных чисел. Представление чисел в памяти.
4. Инструкции сравнения. Регистр флагов. Инструкции условного и безусловного перехода.
5. Стек и инструкции работы с ним. Вызов функций.
6. Вычисления с плавающей запятой. Сопроцессор x87*.
7. Вычисления с плавающей запятой. Инструкции SSE.
8. Векторные инструкции. Отличия наборов SSE и AVX.
9. Кадр стека и регистр RBP. Соглашения о вызовах. Cdecl, System V**.
10. Кадр стека и регистр RBP. Соглашения о вызовах. Cdecl, stdcall, Microsoft x64**.

* точные инструкции запоминать не обязательно, главное помнить его возможности
** правила возврата структур – возможный доп. вопрос.

Модуль 2. Компиляция и компоновка. Уязвимости.

1. Сборка программ. Компиляция. Устройство компиляторов. Компоновка. Понятие символа.
2. Компоновка. Статическая компоновка. Таблицы символов.
3. Компоновка. Динамическая компоновка. Global Offset Table. Релокация.
4. Компоновка. Динамическая компоновка. Global Offset Table. Position Independent Code.
5. Компоновка. Динамическая компоновка. Атака подмены библиотеки и меры защиты.
6. Переполнение буфера на стеке. Потенциальные цели переполнения. Меры защиты.
7. Структура кучи. Переполнение буфера в куче*.
8. Запрет исполнения кода. Return Oriented Programming.
9. ASLR. RIP-relative адресация.

*без особых подробностей и примера с перезаписью GOT.

Модуль 3. Микроархитектура.

1. Выполнение инструкций процессором. Конвейер.
2. Суперскалярная архитектура. Микрооперации.
3. Микрооперации. Атомарные операции. Префикс LOCK.
4. Внеочередное выполнение. Спекулятивное выполнение*.
5. Выравнивание данных. Кэш-память.

***** типы зависимостей и разрыв зависимостей - возможные доп. вопросы.

Модуль 4. Архитектура x86-64.

1. Взаимодействие с внешними устройствами. Прерывания. Таблица векторов прерываний.
2. Сегментная модель памяти. Сегментные регистры. Таблицы дескрипторов.
3. Страничная модель памяти. Буфер трансляции адресов. Файл подкачки.
4. Режимы работы процессора.
5. Кольца защиты. Пространство ядра и пространство пользователя. Системные вызовы.
6. Отладка. Отладочные регистры.
7. Отладка. Программная отладка.
8. Эмуляция. Виртуализация. Типы гипервизоров.
9. CISC, RISC и VLIW.
10. Основные особенности архитектуры ARM.

Примеры практических задач:

1. Напишите функцию bool foo(), которая проверяет, что для введенных пользователем чисел двойной точности a и b верно a > 0 && b <=0. Целевая платформа – Windows x86-64..
2. Напишите функцию float bar(), читает из потока ввода вещественные числа одинарной точности, пока пользователь не введет отрицательное число, и возвращает это число. Целевая платформа – Linux x86-64.
3. Напишите функцию void bar(unsigned int x, double y), которая выводит y в степени x. Целевая платформа – Linux x86-64.
4. Напишите функцию void foo(), которая читает из потока ввода вещественные числа до тех пор, пока пользователь не введет -44. Целевая платформа – Windows x86-32.
5. Напишите функцию void foo(const char* str), которая выводит в консоль, сколько раз в строке встретилась буква, с которой начинается ваша фамилия. Целевая платформа – Linux x86-64.

Примечание 1: фраза «Целевая платформа - X» неявно указывает на соглашение о вызовах, которое вы должны использовать (cdecl/Microsoft x64/System V).

Примечание 2: фраза «Читает из протока ввода …» подразумевает использование gets() или scanf() для чтения соответствующих данных. Слово «выводит …» означает, что данные распечатываются printf() или puts().

Примечание 3: фраза «используя соответствующие функции стандартной библиотеки языка C для вычисления…» означает, что вместо прямых вычислений следует вызывать функции sin()/cos()/tan()/sqrt()/exp(), передавая нужное значение в виде аргумента.

Примечание 4: Размещение переменных в .data/.bss считается недочетом. Константы можно (и даже нужно) помещать в .rodata.