

### Задача 1-Fill: Заполнение пропусков

После **независимого** выполнения приведенных ниже фрагментов кода на языке ассемблера, значения регистра `eax` были сохранены и записаны. Восстановите объявления переменных в шестнадцатеричной записи, оставляя ? на месте тех шестнадцатеричных цифр, однозначно восстановить которые невозможно.

```
x dw 0x????, 0x????
movzx eax, word [x+1]
shl    eax, 4
; EAX = 0xdead0

y dd 0x????????
mov eax, dword [y]
rol    eax, 16
and    al, 0xfe
; EAX = 0xbabecafe

z db 0x??, 0x??, 0x??, 0x??
mov    eax, dword [z]
lea    eax, [8*eax]
add    eax, eax
; EAX = 0xcfaacab0
```

Выводите искомые шестнадцатеричные числа со всеми ведущими нулями без префикса `0x`, разделяя их произвольным количеством пробельных символов.

**Пример правильного форматирования ответа:**

```
??ab ?cd?
0?01234?
01 02 03 04
```

### Задача 2: Откуда дровишки

В волшебном лесу приключилась беда: в нём завелись лесорубы. Лесник Петрович решил на всякий случай пересчитать высоту вековой сосны, чтобы проверить, что её ещё не подпилили, но сосна большая, а памяти в его стареньком компьютере очень мало. Петрович просит нашей помощи в написании рекурсивной функции, решающей данную задачу с использованием соглашения `fastcall`, и мы не в силах отказать старику. Однако соглашение `fastcall` мы изрядно подзабыли, поэтому просим Вас написать такую функцию.

На вход в качестве единственного аргумента Вашей функции подается указатель на корень дерева. Он, как и прочие вершины, представляет собой волшебную структуру из трёх элементов:

```
struct Node {
    char payload;
    struct Node *right;
    struct Node *left;
}
```

Поле `payload` хранит количество яблок в данной вершине дерева, а `right` и `left` являются указателями на правого и левого потомков соответственно (если потомка нет, соответствующий указатель равен `NULL`). На выходе функция должна вернуть одно беззнаковое 32-битное число - высоту дерева. Яблоки пересчитывать не требуется: как-никак лесорубы завелись, а не полевые воры.

**Указание.**

Реализация должна использовать рекурсивный вызов. Также код должен отражать особенности компиляции с ключами `-fomit-frame-pointer` и `-ffixed-ebp` (запрещено использовать указатель фрейма и регистр `ebp` в целом).

#### Оформление решения

В этой задаче от вас требуется написать только одну функцию с именем `tree_height_rec`, а не всю программу. Файл-посылка должен содержать искомую функцию, объявление переменных, функций стандартной библиотеки, вспомогательные функции (если есть). Выводить на экран ничего не нужно, проверяться будет возвращаемое значение функции. В случае несоблюдения соглашения о вызовах решение будет отклоняться автоматически со статусом "Ошибка выполнения".

Пример файла-посылки:

```
section .text
global tree_height_rec
tree_height_rec:
    // реализация функции
```

### Задача 3 Динамическая память

Моделный менеджер памяти управляет кучей из 32 четырехбайтных машинных слов. Для отслеживания свободных блоков используется неявный список. Начальный и последний блок – служебные, для пользователя они недоступны. Байтовый размер блока хранится в заголовке и граничном теге. В выделенных блоках, за исключением начального блока, граничный тег не используется. Предоставляемая пользователю память выравнивается по 8-ми байтной границе (границы обозначены засечками). Поиск свободного блока начинается с текущей позиции в списке, выбирается первый подходящий. При расщеплении используется первая часть блока, вторая часть становится текущей позицией. Слияние блоков проводится незамедлительно. Начальное состояние кучи приведено на рисунке. В машинных словах, занятых заголовком и граничным тегом, показан размер блока и признак занятости (0 – свободен, 1 – занят). Свободный блок белого цвета, занятые блоки заштрихованы, неиспользуемая из-за выравнивания память подчеркнута.



После выполнения шести обращений к менеджеру динамической памяти

```
p1 = malloc(24);
p2 = malloc(17);
p3 = malloc(4);
free(p2);
p2 = malloc(48);
free(p1);
```

А) Рассчитайте и запишите в первой строке ответа пиковое использование памяти U

Б) Опишите в следующих строках ответа получившееся состояние кучи

Формат записи пикового использования памяти - несократимая дробь  $U=N/M$ , где N и M натуральные числа.

Формат описания состояния кучи следующий. Неиспользуемое слово в начале кучи и служебные блоки не указываются. На отдельной строке описывается каждый блок. Через запятую описываются состояния четырехбайтных машинных слов. Заголовок и граничный тег блока обозначаются  $L/status$ , где L – размер блока, а status – состояние блока (0 – свободен, 1 – занят). Слова, предоставленные для размещения пользовательских данных, включая неиспользуемое пространство для выравнивания, обозначаются символом '\*'. Слова свободного блока обозначаются символом '@'.

**Пример форматирования ответа:**

U=7/16

32/0,@,@,@,@,@,32/0

48/1,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*,\*

# Задача 4-StackFrame: Stack Frame

## Stack Frame

Дана функция на языке C:

```
int copy(char *command, int add, short *mul) {
    char buffer[10] = "> ";
    int result = 2024;
    strcpy(buffer + 2, command);
    result += add * *mul;
    return result;
}
```

Для этой функции компилятор построил следующий код:

```
copy:
    endbr32
    push    ebp
    mov     ebp, esp
    sub     esp, 56
    mov     eax, dword[ebp+8]
    mov     dword[ebp-44], eax
    mov     eax, dword[ebp+16]
    mov     dword[ebp-48], eax
    mov     eax, dword[gs:20]
    mov     dword[ebp-12], eax
    xor     eax, eax
    mov     dword[ebp-22], 8254
    mov     dword[ebp-18], 0
    mov     word[ebp-14], 0
    mov     dword[ebp-28], 2024
    lea     eax, [ebp-22]
    add     eax, 2
    sub     esp, 8
    push    dword[ebp-44]
    push    eax
    call    strcpy
    add     esp, 16
    mov     eax, dword[ebp-48]
    movzx   eax, word[eax]
    cwde
    imul    eax, dword[ebp+12]
    add     dword[ebp-28], eax
    mov     eax, dword[ebp-28]
    mov     edx, dword[ebp-12]
    sub     edx, dword[gs:20]
    je      .L3
    call    __stack_chk_fail
.L3:
    leave
    ret
```

I. В **первой строке** ответа укажите использованное при вызове данной функции соглашение вызова. Для этого выпишите **одну букву**, соответствующую верному варианту:

- A. fastcall
- B. stdcall
- C. системный вызов
- D. cdecl

II. Во **второй строке** ответа укажите механизмы защиты от эксплуатации уязвимости переполнения буфера, использование которых можно установить из приведённого кода. Для этого выпишите **одну или несколько букв через пробел**, соответствующих верным вариантам:

- A. BVP
- B. канарейка
- C. DEP
- D. ASLR
- E. CET

III. В **третьей строке** ответа необходимо выписать состояние фрейма функции в момент времени непосредственно перед вызовом функции strcpy. Требуется выписать значения ячеек памяти, начиная с адреса, по которому расположены аргументы функции сору и заканчивая ячейкой, на которую указывает регистр ESP. Для формирования ответа выберите верные значения из списка ниже и выпишите их номера в правильном порядке в одну строку через пробел. Начинайте выписывать со значений, соответствующих **старшим** адресам ячеек памяти, и продолжайте в направлении младших адресов (т.е. в направлении роста стека). Значения могут повторяться. В скобках указан размер значений в байтах. Для последовательности выравнивающих байт размер не уточняется, т.е. любое ненулевое количество подряд идущих выравнивающих байт может быть описано единственным числом 21. Выравнивающие байты в начале ответа (если они есть) можно не выписывать.

1 переменная result (4)	6 параметр mul (4)	11 сохранённый EAX (4)	16 сохранённый ESI (4)	21 выравнивающие байты (*)
2 массив buffer (10)	7 выюрок (4)	12 сохранённый EBP (4)	17 сохранённый EDI (4)	22 значение *mul (2)
3 адрес buffer + 2 (4)	8 канарейка (4)	13 сохранённый EDX (4)	18 сохранённый EBX (4)	
4 параметр command (4)	9 колибри (4)	14 сохранённый ECX (4)	19 сохранённый ESP (4)	
5 параметр add (4)	10 адрес возврата (4)	15 сохранённый EFX (4)	20 сохранённый EIP (4)	

Пример форматирования ответа:

A  
A B C D E  
20 15 10 5

### Задача 5-FP: IEEE 754

Мальчик Вася продолжает изучать программирование. С прошлого экзамена прошел год, так что ему уже целых 9 лет! Теперь он использует еще и 9-битные типы данных, и так как его любимый язык программирования не поддерживает типы такого размера, ему приходится программировать на бумажке. Он уже изучил работу с целыми числами, но в силу своего возраста еще не проходил в школе дроби и не знает, что такое вещественные числа. Помогите ему понять разницу между девятибитным знаковым целочисленным типом и девятибитным IEEE-754 вещественным типом с плавающей точкой (4 бита под мантиссу), ответив на следующие вопросы:

- A. Сколько различных чисел, представимых в этом вещественном формате, меньше чем -9?
- B. На сколько наибольшее число, представимое в целочисленном формате, больше чем наибольшее число, представимое в вещественном формате?
- C. Сколько отрицательных чисел, представимых в целочисленном формате, не представимы в вещественном формате?

*Подсказка: бесконечность — не число!*

#### Формат ответа

Ответы задаются по одному на строке, порядок их следования фиксирован. Названия переменных (A, B, C) отделяются от значений знаком равенства. Все пробельные символы будут проигнорированы.

Пример ответа, удовлетворяющего формату:

A = 1  
B = 2  
C = 3

### Задача 6. Аппаратура компьютера

Выберите из приведенных утверждений истинные.

Для каждой группы утверждений выпишите буквы без пробелов в алфавитном порядке на отдельной строке. Если в группе утверждений нет ни одного верного, оставьте строку пустой.

Какие высказывания верны в отношении элементной базы компьютера?

- A. Статическая память сохраняет состояние даже после отключения питающего напряжения
- B. Ячейка динамической памяти хранит информацию в виде заряда конденсатора
- C. Современная аппаратура компьютера разрабатывается на специализированных языках программирования
- D. Закон Гроша утверждает, что производительность компьютера увеличивается пропорционально его стоимости

Какие высказывания верны в отношении шин данных?

- A. Шина PCI в начальный период работы настраивается командами IN и OUT
- B. Топология связей в шине определяется ее разрядностью
- C. Шина PCI позволяет устройству взаимодействовать с оперативной памятью (читать и писать) напрямую, без участия процессора
- D. Арбитраж шины – процесс взыскания лицензионный отчислений с производителя устройств, способных подключаться к данной шине

Какие высказывания верны в отношении постоянной памяти?

- A. Длительность считывания данных с поверхности диска определяется только трековой плотностью и скоростью вращения диска
- B. Ячейка флэш-памяти с ростом числа хранимых состояний вырабатывается на порядки быстрее
- C. Ограничение скорости передачи данных по внешнему диаметру диска было полностью преодолено кэш-памятью для запрашиваемых блоков
- D. У жестких дисков время записи на порядок дольше времени чтения

Какие высказывания верны в отношении оперативной памяти?

- A. Циклы регенерации нужны для оповещения процессора о подключении или отключении модуля оперативной памяти
- B. Латентность доступа к памяти сокращается при понижении питающего напряжения
- C. Повторное использование адреса строки позволяет ускорить доступ к данным
- D. Принцип локальности применим как к данным, так и к коду

**Пример правильно форматированного ответа:**

ABCD  
AB  
  
B

Задача 7 Виртуальная память

Память модельного компьютера состоит из 512 адресуемых ячеек размером 1 байт. Выполняется страничная трансляция линейных адресов при обращении к физической памяти. Размер страницы – 32 байта. Транслированные адреса сохраняются в TLB, организованный как кэш прямого отображения. Обращение к физической памяти предваряется проверкой кэша данных, имеющего следующее устройство: 2-канальный множественно ассоциативный, 16 байт в строке, 2 набора. Даны: состояние TLB, фрагмент таблицы страниц, кэш данных. Бит р в TLB и таблице страниц показывает присутствие страницы.

Фрагмент таблицы страниц			Состояние TLB					Кэш данных		
VPN	PPN	р	Набор	tag	v	PPN	р	Набор	tag	v
9	3	1	0	3	1	F	1	0	6	1
3	5	1	1	1	1	A	1		A	1
F	9	0	2	1	1	3	1	1	5	0
A	C	0	3	0	1	5	1		8	1

Исходя из того, как именно будет происходить чтение байта по виртуальному линейному адресу 0x123, выпишите в ответе следующие значения, расположив их на отдельных строках в заданном порядке:

- Номер виртуальной страницы VPN (число в шестнадцатеричной кодировке)
- Номер запрашиваемого набора в TLB (число в шестнадцатеричной кодировке)
- Попадание в TLB (yes/no)
- Страница доступна (yes/no)
- Номер запрашиваемого набора в кэше данных (число в шестнадцатеричной кодировке)
- Поле тег в адресе при обращении в кэш данных (число в шестнадцатеричной кодировке)
- Попадание в кэш данных (yes/no)

Если на вопрос ответить невозможно, например, страница недоступна и дальнейшее извлечение данных из памяти не выполняется, вследствие чего невозможно указать номер набора в кэше памяти, тег и т.п., в таких случаях пишите символ '-'.

Пример правильно форматированного ответа:

B  
0  
yes  
yes  
7  
0  
no

**Задача 8-Linking: Размещение данных, связывание символов**

Си-программа состоит из двух модулей: m1. c и m2. c, содержимое которых приведено ниже.

Программа компилируется с опцией -fcommon.

m1.c

#include <stdio.h>  
  
extern int series\_mult(const int \*, int, int (\*)(int));  
  
int multiplier = 2;  
  
int mul2(int a) {  
 return 2 \* a;  
}  
  
static int arr[4] = {0};  
  
int main(void) {  
 for (int i = 0; i < 4; ++i)  
 arr[i] = mul2(i);  
 int result = series\_mult(arr, 4, &mul2);  
 printf("%d\n", result);  
 return 0;  
}

m2.c

int multiplier;  
  
static int square(int x) {  
 return x \* x;  
}  
  
int series\_mult(int \*arr, int end, int (\*func)(int)) {  
 int ans = 0;  
 if (!func)  
 func = &square;  
 for (int i = 0; i < end; i++) {  
 ans += func(arr[i]);  
 }  
 return ans \* multiplier;  
}

Заполните таблицу, приведенную ниже. Ячейки таблицы разделены точкой с запятой. Для каждого заданного в таблице имени переменной или функции укажите (+/–), содержится ли соответствующая запись в таблице символов .symtab объектного файла. Если да, укажите тип связывания символа (local/global), в каком модуле (m1.o/m2.o/если символ в обоих модулях является COMMON-символом – указывайте оба модуля через запятую: m1.o, m2.o) и в какой именно секции этого модуля (.text/.bss/.data/если символ в обоих модулях является COMMON-символом – укажите в этом поле COMMON) символ определен. Если ответ дать невозможно – ставьте прочерк (–). Если символ определен в модуле, отличном от m1.o и m2.o, в столбцах "Модуль, в котором символ определен" и "Секция, в которой символ определен" ставьте прочерк (–).

Исходный файл; Объектный файл; Имя функции/переменной; Присутствует ли в .symtab объектного файла; Тип связывания символа; Модуль, в котором символ определен; Секция, в которой символ определен	
m1.c; m1.o; arr; –; –; –	
m1.c; m1.o; multiplier; –; –; –	
m1.c; m1.o; result; –; –; –	
m2.c; m2.o; square; –; –; –	
m1.c; m1.o; series_mult; –; –; –	

Скопируйте 5 строк таблицы (кроме заголовка) в поле ответа и заполните прочерки там, где это необходимо. Пробельные символы при проверке не учитываются.

Столбцы "Исходный файл", "Объектный файл", "Имя функции/переменной" следует оставить неизменными. Остальные столбцы должны быть заполнены в соответствии с приведенными ниже возможными значениями.

Название столбца	Возможные значения столбца
Присутствует ли в .symtab объектного файла	+/–
Тип связывания символа	local/global/–
Модуль, в котором символ определен	m1.o/m2.o/m1.o, m2.o/–
Секция, в которой символ определен	.text/.data/.bss/Common/–

### Задача 9-Reloc: Преобразование ссылок

Си-программа состоит из двух модулей: 1.с и 2.с, использующих общий заголовочный файл header.h.

Объектные модули 1.o и 2.o были получены в результате компиляции соответствующих модулей исходного кода с опцией -fno-PIC. После этого в результате компоновки gcc 1.o 2.o -o out был получен исполняемый файл out.

#### Дано

```
/* header.h: */

#include <stdio.h>

struct holiday {
    void (*print)(char *, char*);
    char *date;
    char *holiday;
};

extern void is(char *, char*);
extern void print_holiday(void);
```

```
/* 1.c: */

#include "header.h"

void is(char *date, char *holiday)
{
    printf("%s is %s.\n", date, holiday);
}

int main(void)
{
    print_holiday();
    return 0;
}
```

```
/* 2.c: */

#include "header.h"

struct holiday day = {
    .print = is,
    .date = "June 3",
    .holiday = "World Bicycle Day"
};

void print_holiday()
{
    day.print(day.date, day.holiday);
}
```

1.o: file format elf32-i386

Disassembly of section .text:

```
00000000 <is>:
 0: 55                push    ebp
 1: 89 e5             mov     ebp,esp
 3: 83 ec 0c          sub     esp,0xc
 6: ff 75 0c          push    DWORD PTR [ebp+0xc]
 9: ff 75 08          push    DWORD PTR [ebp+0x8]
 c: 68 00 00 00 00    push    0x0
                   d: R_386_32      .rodata.str1.1
11: e8 fc ff ff ff    call    12 <is+0x12>
                   12: R_386_PC32      printf
16: 83 c4 10          add     esp,0x10
19: c9                leave   esp
1a: c3                ret

0000001b <main>:
1b: 55                push    ebp
1c: 89 e5             mov     ebp,esp
1e: 83 e4 f0          and     esp,0xfffffff0
21: e8 fc ff ff ff    call    22 <main+0x7>
                   22: R_386_PC32      print_holiday
26: 31 c0             xor     eax,eax
28: c9                leave   esp
29: c3                ret
```

2.o: file format elf32-i386

Disassembly of section .text:

```
00000000 <print_holiday>:
0: 55                push    ebp
1: 89 e5             mov     ebp,esp
3: 83 ec 08          sub     esp,0x8
6: a1 00 00 00 00    mov     eax,ds:0x0
7: R_386_32         day
b: 8b 0d 08 00 00 00 mov     ecx,DWORD PTR ds:0x8
d: R_386_32         day
11: 8b 15 04 00 00 00 mov     edx,DWORD PTR ds:0x4
13: R_386_32         day
17: 83 ec 08          sub     esp,0x8
1a: 51                push    ecx
1b: 52                push    edx
1c: ff d0             call    eax
1e: 83 c4 10          add     esp,0x10
21: 90                nop
22: c9                leave
23: c3                ret
```

Известно:

- Содержимое переменной day (из секции .data файла out):

f9 ae d3 08 0b 22 e5 08 12 22 e5 08

- Содержимое секции .rodata файла out, полученное с помощью hexdump -C (специальные символы, в частности, нуль-терминатор, в правой колонке отображаются в виде точки):

```
25 73 20 69 73 20 25 73 2e 0a 00 4a 75 6e 65 20 |%s is %s...June |
33 00 57 6f 72 6c 64 20 42 69 63 79 63 6c 65 20 |3.World Bicycle |
44 61 79 00                                     |Day. |
```

- Функция print\_holiday была размещена по адресу 0x08d40000.
- Первая из ссылок в print\_holiday получила значение 2c 51 e6 08.

Найти

- Значение ссылки типа R\_386\_32 в функции is;
- Значение ссылки типа R\_386\_PC32 в функции main;
- Значение третьей ссылки в функции print\_holiday.

Формат ответа

Для каждого из заданий выше необходимо выписать байты в порядке их следования в бинарном файле. Каждый байт кодируется двумя шестнадцатеричными цифрами. Соседние байты могут быть отделены пробельными символами.

Ответы задаются по одному на строке, порядок их следования фиксирован. Номера заданий отделяются от значений знаком равенства. Пример ответа, удовлетворяющий формату, приведён ниже:

```
1 = 00 0c d1 30
2 = f0 ee db ba
3 = fc ff ff ff
```

## Задача 10. Микроархитектура процессора

Выберите из приведенных утверждений истинные.

Для каждой группы утверждений выпишите буквы без пробелов в алфавитном порядке на отдельной строке. Если в группе утверждений нет ни одного верного, оставьте строку пустой.

Какие высказывания верны в отношении аппаратных средств архитектуры x86, обеспечивающих многозадачную работу?

- A. Размер сегмента памяти может быть 4КБ
- B. Кольца защиты – механизм, применяемый для изоляции ресурсов привилегированного режима работы процессора от непривилегированного.
- C. Для работы таймера необходим механизм прерываний
- D. Виртуальная память предназначена для скрытия от программиста кэшей процессора

Какие высказывания верны в отношении архитектуры RISC-V?

- A. Описание архитектуры процессора можно преобразовать в битовый поток для FPGA
- B. Базовый набор команд содержит специализированные команды для ускорения работы ИИ
- C. Использовать архитектуру RISC-V можно только заплатив лицензионные отчисления
- D. Команда RISC-V декодируется существенно проще, чем у любой CISC архитектуры

Какие высказывания верны в отношении микроархитектуры процессора?

- A. Кэш памяти позволяет сохранять данные из регистров устройств, отображенных в память
- B. Конвейер позволяет сократить длительность выполнения отдельной команды
- C. Таксономия Флинна независимо рассматривает параллелизм в потоках команд и данных
- D. Механизм предсказания переходов позволяет конвейеру работать более эффективно

Пример правильно сформатированного ответа:

ABCD  
AB  
C