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????
movsx eax, word [x+1]
shr   eax, 4
shl   eax, 4
; EAX = 0x23a0

y dd 0x????????
mov eax, dword [y]
rol eax, 56
xor ah, 0xfe
; EAX = 0xbeca00ba

z db 0x??, 0x??, 0x??, 0x??
mov  eax, dword [z]
lea  eax, [8*eax]
add  eax, eax
; EAX = 0x1caffca0
```

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

Поле `payload` хранит количество фиников в данной вершине дерева, а `left` и `right` являются указателями на левого и правого потомков соответственно (если потомка нет, соответствующий указатель равен NULL). На выходе функция должна вернуть одно беззнаковое 32-битное число - высоту дерева. Финики 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

Модельный 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 перечеркнута.

После выполнения шести обращений к менеджеру динамической памяти

```
p1 = malloc(7);
p2 = malloc(48);
free(p1);
p3 = malloc(4);
p1 = malloc(32);
free(p3);
```

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:

```
short mcopy(char *command, short count) {
    char buffer[15] = "> ";
    short result = 0;
    memcpy(buffer + 2, command, count);
    result += count / 5;
    return result;
}
```

Для этой функции компилятор построил следующий код:

```
mcopy:
        endbr32
        push    ebp
        mov     ebp, esp
        sub     esp, 56
        mov     edx, dword[ebp+12]
        mov     eax, dword[ebp+8]
        mov     dword[ebp-44], eax
        mov     eax, edx
        mov     word[ebp-48], ax
        mov     eax, dword[gs:20]
        mov     dword[ebp-12], eax
        xor     eax, eax
        mov     dword[ebp-27], 8228
        mov     dword[ebp-23], 0
        mov     dword[ebp-20], 0
        mov     dword[ebp-16], 0
        mov     word[ebp-30], 0
        movsx   edx, word[ebp-48]
        lea     eax, [ebp-27]
        add     eax, 2
        sub     esp, 4
        push    edx
        push    dword[ebp-44]
        push    eax
        call    memcpy
        add     esp, 16
        movzx   eax, word[ebp-48]
        movsx   edx, ax
        imul    edx, edx, 26215
        shr     edx, 16
        sar     dx
        sar     ax, 15
        mov     ecx, eax
        mov     eax, edx
        sub     eax, ecx
        mov     edx, eax
        movzx   eax, word[ebp-30]
        add     eax, edx
        mov     word[ebp-30], ax
        movzx   eax, word[ebp-30]
        mov     edx, dword[ebp-12]
        sub     edx, dword[gs:20]
        je      .L3
        call    __stack_chk_fail
.L3:
        leave
        ret     8
```

I. В **первой строке** ответа укажите использованное при вызове данной функции соглашение вызова. Для этого выпишите **одну букву**, соотвествующую верному варианту:
   A. cdecl
   B. fastcall
   C. stdcall
   D. системный вызов

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**, соотвествующих верным вариантам:
   A. CET
   B. BVP
   C. DEP
   D. канарейка
   E. ASLR

III. В **третьей строке** ответа необходимо выписать состояние фрейма функции в момент времени непосредственно перед вызовом функции `memcpy`. Требуется выписать значения ячеек памяти, начиная с адреса, по которому расположены аргументы функции `mcopy` 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 20. Выравнивающие байты в начале ответа (если они есть) можно не выписывать.

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

Пример форматирования ответа:

```
A
A B C D E
19 18 17 16
```

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 наименьшее число, представимое в целочисленном формате, меньше чем наименьшее число, представимое в вещественном формате?
C. Сколько положительных чисел, представимых в целочисленном формате, представимы также и 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. Элемент статической памяти предполагает замыкание выходов на свои же входы
B. Динамическая память дороже и быстрей статической
C. Логика работы алгоритмов, выполняющихся на FPGA, задается единожды, при изготовлении микросхемы на производстве
D. Закон Белла утверждает, что примерно каждое десятилетие появляется новый класс компьютеров и открывается новая область, где этот класс компьютеров начинает применяться

Какие высказывания верны в отношении шин данных?

A. Обычно регистры PCI-устройства отображаются в адресное пространство, перекрывая доступ к 4КБ памяти
B. Контроллер шины USB в свою очередь является PCI-устройством
C. На любой шине обмениваться данными может только пара подключенных к ней устройств
D. В архитектуре x86 синхронизация ядер при обращении к шине памяти невозможна

Какие высказывания верны в отношении постоянной памяти?

A. Плавающий затвор позволяет сохранять состояние ячейки памяти даже после отключения питающего напряжения
B. Длительность доступа к блоку данных на жестком диске зависит только от скорости вращения пластины
C. Запись данных на жесткий диск возможна только на очищенную дорожку
D. У твердотельных накопителей время записи примерно на порядок дольше времени чтения

Какие высказывания верны в отношении оперативной памяти?

A. Расслоение оперативной памяти позволяет кратно повысить ее пропускную способность, но в ограниченных приделах
B. Буфер строки используется для извлечения значения суперячейки после передачи строба столбца
C. Латентность доступа к памяти обратно пропорциональна ее пропускной способности
D. Временная локальность – это когда процессор опрашивает отдельный модуль оперативной памяти, сколько тактов требуется на передачу стробов строки и столбца

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 – 64 байта. Транслированные адреса сохраняются в TLB, организованный как полностью ассоциативный кэш. Обращение к физической памяти предваряется проверкой кэша данных, имеющего следующее устройство: 2-канальный множественно ассоциативный, 8 байт в строке, 4 набора. Даны: состояние TLB, фрагмент таблицы страниц, кэш данных. Бит p в TLB и таблице страниц показывает присутствие страницы.

Фрагмент таблицы страниц

VPN	PPN	p
3	7	1
4	5	0
6	3	0
2	4	1

Состояние TLB

tag	v	PPN	p
3	1	7	1
5	1	6	1
7	1	3	1
2	1	4	1

Кэш данных

Набор	tag	v
0	9	0
	C	1
1	A	0
	F	0
2	4	1
	8	1
3	8	1
	3	1

Исходя из того, как именно будет происходить чтение байта по виртуальному линейному адресу `0x98`, выпишите в ответе следующие значения, расположив их на отдельных строках в заданном порядке:

- Номер виртуальной страницы VPN (число в шестнадцатеричной кодировке)
- Смещение внутри страницы (число в шестнадцатеричной кодировке)
- Попадание в TLB (yes/no)
- Страница доступна (yes/no)
- Номер физической страницы PPN (число в шестнадцатеричной кодировке)
- Номер запрашиваемого набора в кэше 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
A
no
yes
D
1
yes
```

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>

int end;

int series_mult(int, int (*)(int));

int init = 3;

static int mul2(int a) {
    return a * 2;
}

int main(void) {
    end = 15;
    int result = series_mult(3, &mul2);
    printf("%d\n", result);
    return 0;
}
```

m2.c

```
int end;

int changer = 3;

int init;

static int nothing(int a) {
  return a;
}

int series_mult(int start, int (*func)(int)) {
  int ans = init;
  if (!func)
    func = &nothing;
  for (int i = start; i < end; i++) {
    ans *= func(i);
  }
  return ans + changer;
}
```

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 определён в модуле, отличном от `m1.o` и `m2.o`, в столбцах "Модуль, в котором символ определён" и "Секция, в которой символ определён" ставьте прочерк (−).

Исходный файл; Объектный файл; Имя функции/переменной; Присутствует ли в `.symtab` объектного файла; Тип связывания символа; Модуль, в котором символ определён ; Секция, в которой символ определён

```
m2.c; m2.o; end; −; −; −; −
m2.c; m2.o; changer; −; −; −; −
m1.c; m1.o; mul2; −; −; −; −
m1.c; m1.o; result; −; −; −; −
m2.c; m2.o; init; −; −; −; −
```

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 приведёнными ниже возможными значениями.

Название столбца	**Возможные значения столбца**
Присутствует ли в `.symtab` объектного файла	`+/−`
Тип связывания символа	`local/global/−`
Модуль, в котором символ определён	`m1.o/m2.o/m1.o, m2.o/−`
Секция, в которой символ определён	`.text/.data/.bss/COMMON/−`

Задача 9-Reloc: Преобразование ссылок

Си-программа состоит из двух модулей: `1.c` и `2.c`, 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 *holiday;
    char *date;
};

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,
    .holiday = "World Bicycle Day",
    .date = "June 3"
};

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
  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
  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 04 00 00 00       mov    ecx,DWORD PTR ds:0x4
            d: R_386_32    day
  11:    8b 15 08 00 00 00       mov    edx,DWORD PTR ds:0x8
            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`):

  ```
  a9 80 c3 08 0b 96 c4 08 1d 96 c4 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 57 6f 72 6c 64  |%s is %s...World|
  20 42 69 63 79 63 6c 65  20 44 61 79 00 4a 75 6e  | Bicycle Day.Jun|
  65 20 33 00                                       |e 3.|
  ```

- Функция `print_holiday` была размещена по адресу `0x08c380d4`.
- Первая из ссылок в `print_holiday` получила значение `1c b0 c4 09`.

Найти

1. Значение ссылки типа `R_386_32` в функции `is`;
2. Значение ссылки типа `R_386_PC32` в функции `main`;
3. 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. Базовый адрес сегмента памяти может быть 0
B. Привилегированный режим работы в x86 был разработан для того, чтобы управлять энергопотреблением компьютера
C. Механизмы сегментной защиты памяти и виртуальной памяти кэшируют результаты своей работы непосредственно на кристалле процессора
D. Таймер позволяет регулярно передавать управление операционной системе

Какие высказывания верны в отношении архитектуры RISC-V?

A. Базовый набор команд поддерживает операции над тензорами
B. Описание архитектуры процессора свободно распространяется
C. Команда условной передачи управления считывает флаги регистра состояния FLAGS
D. К памяти можно обратиться только специальными командами загрузки и выгрузки (LOAD и STORE)

Какие высказывания верны в отношении микроархитектуры процессора?

A. Увеличение размеров кэша сокращает энергопотребление процессора
B. Конвейер позволяет увеличить число выполнившихся команд за единицу времени
C. Из-за расширений ISA MMX и SSE архитектуру x86 можно отнести к SIMD, согласно таксономии Флинна
D. Функциональное устройство отвечает за кэширование кодов команд в специализированной L1 кэш-памяти

Пример правильно форматированного ответа:

```
ABCD
AB
C
```