Problem 1-Fill: Заполнение пропусков

Восстановите все пропуски ("?") в приведённом коде, чтобы он завершался успешно (метка .success):

```
section .data
    j db 0x18, 0x77, ?, 0xe, ?, 0xf, 0x5d, 0xe, 0xd1, 0xf5
    f db 0xe, 0xe8, 0x9c, 0xac, 0x1f, ?, 0x26, 0x0, 0x80, 0x8
    o db 0x0, 0x6e, 0x8e, 0xea

section .text
global main
main:
    mov eax, 2

.l1:
    movsx ebx, byte[j + eax]
    movsx ecx, byte[j + eax + 1]
    cmp ebx, ecx
    jae .l2
    add eax, 2
    jmp .l1

.l2:
    mov cx, word[j + ecx]
    mov edx, dword[f + 7]

    rol edx, cl
    cmp edx, 0x20000002
    je .success

    mov eax, 2
    jmp .l1

.success:
    xor eax, eax
    ret
```

Восстановленные значения вводить по одному на строке в том же порядке, что и в задаче (слева направо, сверху вниз). **Один "?" соответствует одному значению.** Если вариантов значений несколько - писать минимальный (0x92 > 0x11).

"Да пребудет с вами отладчик"

Пример ответа:

```
ca
c
a0
```

Problem 2-Switch: Кто сказал таблица?

Дана функция на языке ассемблера:

```
section .rodata
    choice.t dd choice.0, choice.1, choice.2, choice.3

section .text
global choice
choice:
    push    ebp
    mov     ebp, esp
    mov     eax, dword [ebp + 8]
    mov     edx, dword [ebp + 12]
    sub     edx, 3
    cmp     edx, 3
    ja      choice.4
    jmp     [choice.t + 4 * edx]
choice.0:
    and     eax, dword [ebp + 12]
    leave
    ret
choice.1:
    shr     eax, 2
    leave
    ret
choice.2:
    lea     eax, [eax + 4 * eax]
    leave
    ret
choice.3:
    add     eax, dword [ebp + 12]
    leave
    ret
choice.4:
    xor     eax, eax
    leave
    ret
```

Необходимо написать эквивалентную функцию на языке Си. Функция должна принимать два аргумента типа **int**.

При написании функции запрещается использовать оператор **if** и тернарную операцию!

Оформление решения

В этой задаче от вас требуется написать только одну функцию, а не всю программу. Файл-посылка должен содержать только искомую функцию **без** подключений библиотек. Выводить на экран ничего не нужно, проверяться будет возвращаемое значение функции.

Examples

Input

```
7 3
```

Output

```
3
```

Problem 3-Librephobia: Либрофобия

У мальчика Пети редкое заболевание: либрофобия — боязнь вызова библиотечных функций. Чтобы Петя смог адаптироваться к жизни во взрослом обществе, доктор прописал ему вызов одной библиотечной функции в день. Но Петя не хочет лечиться, поэтому за кругленькую сумму он нанял вас, чтобы вы вызвали все функции вместо него. Вы, конечно, добропорядочный студент, but 300 bucks is 300 bucks...

Напишите на языке ассемблера функцию `callall`, принимающую указатель на массив четырехбайтовых значений, описывающих библиотечные функции, вызывающую их, и возвращающую указатель на массив возвращаемых ими значений.

Входной массив устроен следующим образом. В нулевом элементе массива записано число вызываемых функций $M \leq 10000$, а следом записана информация об M вызываемых функциях подряд по порядку с первой. Инофрмация о каждой функции состоит из $N+2$ последовательных значений: число аргументов функции $N \leq 100$, указатель на функцию, аргументы функции по порядку с первого.

Функция `callall` должна удовлетворять соглашению **cdecl**. Перед вызовом каждой библиотечной функции стек должен быть выровнен по 16 байт.

Пример массива входных данных:

```
2, 3, scanf, "%d %d", &a, &b, 2, printf, "%d", a + b
```

Пример файла-посылки:

```
  section .text

  global callall
  callall:
    ...
```

Problem 4-StackFrame: Stack Frame

Stack Frame

Дана функция на языке C:

```
int hello(char *name, int *x, char q) {
    char gr[20] = "Hello, ";
    int r = 0x2023;
    strcat(gr, name);
    int *a = q ? &r : x;
    *a = 0x622;
    return r;
}
```

Для этой функции компилятор построил следующий код:

```
hello:
        push   ebp
        mov    ebp, esp
        push   ebx
        push   edi
        sub    esp, 0x28
        mov    edi, edx

        mov    ebx, dword[ebp + 0x8]
        mov    eax, dword[gs:0x14]
        mov    dword[ebp-0xc], eax
        xor    eax, eax

        mov    dword[ebp-0x20], 0x6c6c6548
        mov    dword[ebp-0x1c], 0x202c6f
        mov    dword[ebp-0x18], 0x0
        mov    dword[ebp-0x14], 0x0
        mov    dword[ebp-0x10], 0x0

        mov    dword[ebp-0x24], 0x2023

        push   ecx
        lea    eax, [ebp-0x20]
        push   eax
        call   strcat
        add    esp, 0x10

        lea    eax, [ebp-0x24]
        mov    edx, ebx
        test   dl, dl
        cmovne edi, eax
        mov    dword[edi], 0x622
        mov    eax, dword[ebp-0x24]

        mov    edx, dword[ebp-0xc]
        xor    edx, dword[gs:0x14]
        je     .L1
        call   __stack_chk_fail

.L1:
        lea    esp, [ebp-0x8]
        pop    edi
        pop    ebx
        pop    ebp
        ret    0x4
```

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

II. Во **второй строке** ответа напишите цифру 1, если использовался указатель фрейма, и цифру 0 в противном случае.

III. В **третьей строке** ответа необходимо выписать состояние фрейма функции в момент времени непосредственно перед вызовом функции `strcat`. Требуется выписать значения ячеек памяти, начиная с адреса, по которому расположены аргументы функции `hello` 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 17. Выравнивающие байты в начале ответа (если они есть) можно не выписывать.

1 переменная `r` (4)	6 колибри (4)	11 сохранённый `EAX (4)`	16 сохранённый `ESI (4)`	21 сохранённый `EDI (4)`
2 параметр `x` (4)	7 канарейка (4)	12 параметр `q` (1)	17 выравнивающие байты (*)	22 адрес переменной `r` (4)
3 адрес возврата (4)	8 вьюрок (4)	13 сохранённый `ECX` (4)	18 сохранённый `EBX` (4)	
4 адрес массива `gr` (4)	9 параметр `name` (4)	14 сохранённый `EDX` (4)	19 сохранённый `ESP` (4)	
5 массив `gr` (20)	10 сохранённый `EBP (4)`	15 сохранённый `EFX` (4)	20 сохранённый `EIP` (4)	

Пример форматирования ответа:

```
A
0
16 6 1 11
```

Problem 5-FP: IEEE 754

Мальчик Вася изучает программирование. Так как ему всего 8 лет, он использует только 8-битные типы данных. Он уже изучил работу с целыми числами и битовые операции, но в силу своего возраста еще не проходил в школе дроби и не знает, что такое вещественные числа. Помогите ему понять разницу между восьмибитным знаковым целочисленным типом и восьмибитным IEEE-754 вещественным типом с плавающей точкой (3 бита под мантиссу), ответив на следующие вопросы:

A. Сколько существует различных целых чисел, представимых в обоих форматах?
B. Чему равна разница между наибольшим целым числом, представимым в целочисленном типе, и наименьшим целым числом, представимым в вещественном типе?
C. Сколько существует различных целых чисел, представимых в обоих форматах, для которых побитовая конъюнкция вещественного представления с вещественной единицей и побитовая конъюнкция целочисленного представления с целочисленной единицей будут равны одному и тому же числу (хоть и в разных форматах)?

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. Архитектура компьютеров

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.

Логический вентиль:

A. Часть цифровой электрической схемы, выполняющий элементарную логическую операцию
B. Компонент процессора, отключающий питающее напряжение
C. Реализуется транзисторами и резисторами с использованием параллельных и последовательных соединений
D. Выдает на выходную линию гармонически колеблющееся напряжение

Увеличить объем данных, получаемый из оперативной памяти, позволяет:

A. Понижение тактовой чистоты
B. Учащение циклов регенерации
C. Повторное использование адреса строки
D. Расслоение (распределение данных по нескольким чипам динамической памяти)

Шина PCI-Express

A. Настраивается командами IN и OUT
B. Имеет топологию «общая шина»
C. Позволяет варьировать ширину (число) соединений
D. Имеет общую с оперативной памятью адресацию устройств

Жесткий диск

A. Общее время доступа к блоку данных зависит от скорости вращения диска
B. Объем хранимой информации определяется только плотностью записи (линейной плотностью) и трековой плотностью
C. Содержит контроллер (специализированный процессор), оперативную память, микросхему постоянной памяти
D. Время записи на порядок дольше времени чтения

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 2-канальный множественно 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
F	3	0
1	B	1
7	9	1
3	C	1

Состояние TLB

Набор	tag	v	PPN	p
0	7	1	9	1
	3	0	2	1
1	2	1	D	1
	1	1	C	1

Кэш данных

Набор	tag	v
0	6	0
	E	1
1	C	0
	6	1
2	7	1
	C	1
3	B	1
	F	0

Исходя из того, как именно будет происходить чтение байта по виртуальному линейному адресу `0x71`, 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)
- Попадание в TLB (yes/no)
- Страница доступна (yes/no)
- Номер запрашиваемого набора в кэш данных (число)
- Поле тег в адресе при обращении в кэш данных (число)
- 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
```

Problem 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 reduce(int, const int *, int (*)(int, int));

int init = 42;

int add(int a, int b) {
    return a + b;
}

static int arr[10] = {0};

int main(void) {
    for (int i = 0; i < 10; ++i)
        arr[i] = i;
    int result = reduce(10, arr, &add);
    printf("%d\n", result);
    return 0;
}
```

m2.c

```
int init;

int reduce(int n, const int *arr, int (*func)(int, int)) {
    int ans = init;
    for (int i = 0; i < n; ++i) {
        ans = func(ans, arr[i]);
    }
    return ans;
}
```

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

`m1.c; m1.o; arr; −; −; −; −`
`m1.c; m1.o; reduce; −; −; −; −`
`m2.c; m2.o; ans; −; −; −; −`
`m2.c; m2.o; init; −; −; −; −`
`m1.c; m1.o; printf; −; −; −; −`

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/−`

Problem 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 descriptor {
    char *name;
    char *description;
    void (*summary)(char *, char*);
};

extern void was_an(char *, char*);
extern void describe_moore(void);
```

```
/* 1.c: */

#include "header.h"

void was_an(char *name, char *description)
{
    printf("%s was an %s.\n", name, description);
}

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

```
/* 2.c: */

#include "header.h"

struct descriptor moore = {
    .name = "Gordon Moore",
    .description = "American businessman, engineer, and the co-founder and emeritus chairman of Intel Corporation",
    .summary = was_an
};

void describe_moore()
{
    moore.summary(moore.name, moore.description);
}
```

```
1.o:     file format elf32-i386


Disassembly of section .text:

00000000 <was_an>:
   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 <was_an+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    describe_moore
  26:    31 c0                    xor    eax,eax
  28:    c9                       leave
  29:    c3                       ret
```

```
2.o:     file format elf32-i386


Disassembly of section .text:

00000000 <describe_moore>:
   0:    55                      push   ebp
   1:    89 e5                   mov    ebp,esp
   3:    83 ec 08                sub    esp,0x8
   6:    a1 08 00 00 00          mov    eax,ds:0x8
            7: R_386_32    moore
   b:    8b 0d 04 00 00 00       mov    ecx,DWORD PTR ds:0x4
            d: R_386_32    moore
  11:    8b 15 00 00 00 00       mov    edx,DWORD PTR ds:0x0
            13: R_386_32    moore
  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 `moore` (из секции `.data` файла `out`):

  ```
  10 e0 b0 08 20 e0 b0 08 b9 c1 04 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 77 61 73 20 61  6e 20 25 73 2e 0a 00 00  |%s was an %s....|
  47 6f 72 64 6f 6e 20 4d  6f 6f 72 65 00 00 00 00  |Gordon Moore....|
  41 6d 65 72 69 63 61 6e  20 62 75 73 69 6e 65 73  |American busines|
  73 6d 61 6e 2c 20 65 6e  67 69 6e 65 65 72 2c 20  |sman, engineer, |
  61 6e 64 20 74 68 65 20  63 6f 2d 66 6f 75 6e 64  |and the co-found|
  65 72 20 61 6e 64 20 65  6d 65 72 69 74 75 73 20  |er and emeritus |
  63 68 61 69 72 6d 61 6e  20 6f 66 20 49 6e 74 65  |chairman of Inte|
  6c 20 43 6f 72 70 6f 72  61 74 69 6f 6e 00        |l Corporation.|
  ```

- Функция `describe_moore` была размещена по адресу `0x08050000`.
- Первая из ссылок в `describe_moore` получила значение `a4 fe bf 08`.

Найти

1. Значение ссылки типа `R_386_32` в функции `was_an`;
2. Значение ссылки типа `R_386_PC32` в функции `main`;
3. Значение третьей ссылки в функции `describe_moore`.

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
```

Problem 10-Quiz-arch-1: 10.1. Аппаратура ЭВМ (20 баллов)

Укажите верные высказывания

A. В x86 сегментная защита памяти и страничная память работают независимо друг от друга
B. Кольца защиты – механизм, препятствующий вскрытию корпуса компьютера
C. Без таймера невозможно гарантировать вытеснение пользовательской программы с процессора
D. Системные прерывания обеспечивают временную остановку процессора на командах NOP и PAUSE

В ответе выпишите через пробел все буквы, соответствующие верным вариантам ответа.

Пример форматирования ответа:

```
A C
```

Problem 10-Quiz-arch-2: 10.2. Аппаратура ЭВМ (20 баллов)

Какие высказывания верны в отношении архитектуры RISC-V?

A. Описание архитектуры процессора свободно распространяется
B. Поместить в регистр 32 разрядную константу одой командой можно только в том случае, когда константа – 0
C. Команды условной передачи управления используют регистр флагов
D. Длина команды варьируется от одного до четырех байт

В ответе выпишите через пробел все буквы, соответствующие верным вариантам ответа.

Пример форматирования ответа:

```
A C
```

Problem 10-Quiz-arch-3: 10.3. Аппаратура ЭВМ (20 баллов)

Укажите верные высказывания.

A. Микроархитектура процессора — множество механизмов, обеспечивающих реализацию архитектуры набора команд (ISA) в процессоре.
B. Конвейер позволяет ускорить выполнение команды
C. Упреждающая выборка памяти в некоторых случаях способна полностью скрыть нехватку объема кэшей
D. Предсказание переходов использует генератор случайных чисел

В ответе выпишите через пробел все буквы, соответствующие верным вариантам ответа.

Пример форматирования ответа:

```
A C
```