Мерочкин Илья БПИ 218

Индивидуальное домашнее задание по архитектуре вычислительных систем №1.
Вариант 6

6. Сформировать массив B, состоящи из элементов массива A, значения которых кратны введенному числу X.

5 баллов

Код на C, лежит в файле asm.c:

```
#include <stdio.h>
#include <stdlib.h>

int a[100000], b[100000];

int get_size(int a_sz, int x) {
    int b_sz = 0, i = 0;
    for (i = 0; i < a_sz; ++i) {
        if (a[i] % x == 0) {
            ++b_sz;
        }
    }
    return b_sz;
}

int main(int argc, char* argv[]) {
    int a_sz = argc - 2, x = atoi(argv[argc - 1]), b_sz = 0, idx = 0, i;
    for (i = 0; i < a_sz; ++i) {
        a[i] = atoi(argv[i + 1]);
    }
    b_sz = get_size(a_sz, x);
    for (i = 0; i < a_sz; ++i) {
        if (a[i] % x == 0) {
```

```
            b[idx] = a[i];
            ++idx;
        }
    }
    for (i = 0; i < b_sz; ++i) {
        printf("%d ", b[i]);
    }
    printf("\n");
    return 0;
}
```

Ассемблерная программа, лежит в файле asm.s (некоторые комментарии не уместились в одну строку, поэтому расположены в двух):

```
        .file   "asm.c"
        .intel_syntax noprefix
        .text
        .globl  a
        .bss
        .align 32
        .type   a, @object
        .size   a, 400000
a:
        .zero   400000
        .globl  b
        .align 32
        .type   b, @object
        .size   b, 400000
b:
        .zero   400000
        .text
        .globl  get_size
        .type   get_size, @function
get_size:
        endbr64
        push    rbp                             # Кладем регистр rbp в стек
        mov     rbp, rsp
        mov     DWORD PTR -20[rbp], edi         # Записываем в аргумент a_sz
значение регистра edi
```

```
        mov     DWORD PTR -24[rbp], esi         # Записываем в аргумент x
значение регистра esi
        mov     DWORD PTR -4[rbp], 0            # Инициализируем локальную
переменную b_sz нулем
        mov     DWORD PTR -8[rbp], 0            # Инициализируем локальную
переменную i нулем
        mov     DWORD PTR -8[rbp], 0
        jmp     .L2
.L4:
        mov     eax, DWORD PTR -8[rbp]
        cdqe
        lea     rdx, 0[0+rax*4]
        lea     rax, a[rip]
        mov     eax, DWORD PTR [rdx+rax]
        cdq
        idiv    DWORD PTR -24[rbp]
        mov     eax, edx
        test    eax, eax
        jne     .L3
        add     DWORD PTR -4[rbp], 1            # увеличиваем значение локальной
переменной b_sz на 1
.L3:
        add     DWORD PTR -8[rbp], 1
.L2:
        mov     eax, DWORD PTR -8[rbp]          # Записываем в eax значение
локальной переменной i
        cmp     eax, DWORD PTR -20[rbp]         # Сравниваем регистр eax с
аргументом x
        jl      .L4
        mov     eax, DWORD PTR -4[rbp]          # Записываем в регистр eax
значение локальной переменной b_sz
        pop     rbp                             # Удаляем регистр rbp из стека
        ret                                     # Завершаем функцию
        .size   get_size, .-get_size
        .section        .rodata
.LC0:
        .string "%d "
        .text
        .globl  main
        .type   main, @function
```

```
main:
        endbr64
        push    rbp
        mov     rbp, rsp
        sub     rsp, 48
        mov     DWORD PTR -36[rbp], edi
        mov     QWORD PTR -48[rbp], rsi
        mov     eax, DWORD PTR -36[rbp]
        sub     eax, 2
        mov     DWORD PTR -12[rbp], eax      # Записываем в переменную a_sz
значение регистра eax (в котором лежит значение argc - 2)
        mov     eax, DWORD PTR -36[rbp]
        cdqe
        sal     rax, 3
        lea     rdx, -8[rax]
        mov     rax, QWORD PTR -48[rbp]
        add     rax, rdx
        mov     rax, QWORD PTR [rax]
        mov     rdi, rax
        call    atoi@PLT                     # Вызываем функцию atoi
        mov     DWORD PTR -16[rbp], eax  # Записываем в переменную x значение
регистра eax
        mov     DWORD PTR -20[rbp], 0     # Инициализируем переменную b_sz
нулем
        mov     DWORD PTR -4[rbp], 0      # Инициализируем переменную idx
нулем
        mov     DWORD PTR -8[rbp], 0      # Инициализируем переменную i нулем
        jmp     .L7
.L8:
        mov     eax, DWORD PTR -8[rbp]
        cdqe
        add     rax, 1
        lea     rdx, 0[0+rax*8]
        mov     rax, QWORD PTR -48[rbp]
        add     rax, rdx
        mov     rax, QWORD PTR [rax]
        mov     rdi, rax
        call    atoi@PLT
        mov     edx, DWORD PTR -8[rbp]
        movsx   rdx, edx
```

```
        lea     rcx, 0[0+rdx*4]
        lea     rdx, a[rip]
        mov     DWORD PTR [rcx+rdx], eax
        add     DWORD PTR -8[rbp], 1
.L7:
        mov     eax, DWORD PTR -8[rbp]
        cmp     eax, DWORD PTR -12[rbp]
        jl      .L8
        mov     esi, DWORD PTR -16[rbp]         # Записываем в регистр esi
значение переменной x
        mov     edi, DWORD PTR -12[rbp]         # Записываем в регистр edi
значение переменной a_sz
        call    get_size                        # Вызываем функцию get_size
        mov     DWORD PTR -20[rbp], eax         # Записываем в локальную
переменную значение регистра eax (a в нем лежит значение, которое вернула
функция get_size)
        mov     DWORD PTR -8[rbp], 0            # Приравниваем локальной
переменной i зачение 0
        jmp     .L9
.L11:
        mov     eax, DWORD PTR -8[rbp]
        cdqe
        lea     rdx, 0[0+rax*4]
        lea     rax, a[rip]
        mov     eax, DWORD PTR [rdx+rax]
        cdq
        idiv    DWORD PTR -16[rbp]
        mov     eax, edx
        test    eax, eax
        jne     .L10
        mov     eax, DWORD PTR -8[rbp]
        cdqe
        lea     rdx, 0[0+rax*4]
        lea     rax, a[rip]
        mov     eax, DWORD PTR [rdx+rax]
        mov     edx, DWORD PTR -4[rbp]
        movsx   rdx, edx
        lea     rcx, 0[0+rdx*4]
        lea     rdx, b[rip]
        mov     DWORD PTR [rcx+rdx], eax
```

```
        add     DWORD PTR -4[rbp], 1
.L10:
        add     DWORD PTR -8[rbp], 1
.L9:
        mov     eax, DWORD PTR -8[rbp]
        cmp     eax, DWORD PTR -12[rbp]
        jl      .L11
        mov     DWORD PTR -8[rbp], 0
        jmp     .L12
.L13:
        mov     eax, DWORD PTR -8[rbp]
        cdqe
        lea     rdx, 0[0+rax*4]
        lea     rax, b[rip]
        mov     eax, DWORD PTR [rdx+rax]
        mov     esi, eax
        lea     rax, .LC0[rip]
        mov     rdi, rax
        mov     eax, 0
        call    printf@PLT
        add     DWORD PTR -8[rbp], 1
.L12:
        mov     eax, DWORD PTR -8[rbp]
        cmp     eax, DWORD PTR -20[rbp]
        jl      .L13
        mov     edi, 10
        call    putchar@PLT
        mov     eax, 0
        leave
        ret
```

В реализованной программе используется функцию (get_size - она считает и возвращает количество элементов, которые будут в массиве b) с передачей данных через параметры, с локальными переменными и с возвращаемым значением. В комментариях ассемблированной программы описывается передача параметров и перенос возвращаемого значения. В нашем случае возвращаемое значение было перенесено из стека в регистр eax, откуда впоследствии было извлечено обратно в стек.