

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

Департамент электронной инженерии

ОТЧЕТ

О ПРАКТИЧЕСКОЙ РАБОТЕ №2

по дисциплине «Системное программирование»

«Основы Ассемблера»

Вариант 3

Студент гр. БИБ201

Шадрунов Алексей

Дата выполнения: 4 декабря 2022 г.

Преподаватель:

Морозов В. И.

«___» _____ 2022 г.

Москва, 2022

Содержание

1	Задание на практическую работу	3
2	Ход работы	3
2.1	Структура программы	3
2.2	Работа программы	5
3	Выводы о проделанной работе	8
	Приложение А	9

1 Задание на практическую работу

Дан массив из 10 слов. Найти сумму остатков от деления каждого из них на 3. Результат поместить в отдельный элемент данных.

2 Ход работы

В ходе работы написана программа, решающая поставленную задачу на языке Assembler. В программе используется синтаксис Intel, а также реализован вывод в консоль без подключения библиотек.

2.1 Структура программы

В программе присутствуют два сегмента: сегмент данных и сегмент кода. В сегменте данных объявлены массив **array** из 10 элементов типа **DW** (definite word, 2 байта) и переменная **result** типа **DW**. Начальное значение **result** = 0, начальное значение массива также задано.

В сегменте кода происходит следующее:

- Объявляется цикл из 10 итераций. В каждой итерации текущий элемент массива, начиная с первого, делится на 3 с помощью инструкции **div**. При этом остаток от деления содержится в части регистра **ah**.
- После получения остатка в регистр **ebx** записывается предыдущее значение переменной **result**, затем командой **add** прибавляется значение остатка и сумма записывается в переменную **result**. Значение указателя сдвигается на 2 байта.
- Далее вызывается процедура вывода переменной **result** в консоль, затем выхода из программы.

Процедуры **print** и **exit** используют системные вызовы (4 — SYS_WRITE и 1 — SYS_EXIT) для вывода символов на экран и для выхода из программы.

Пример запуска программы в среде разработки SASM приведен на рисунке 1.

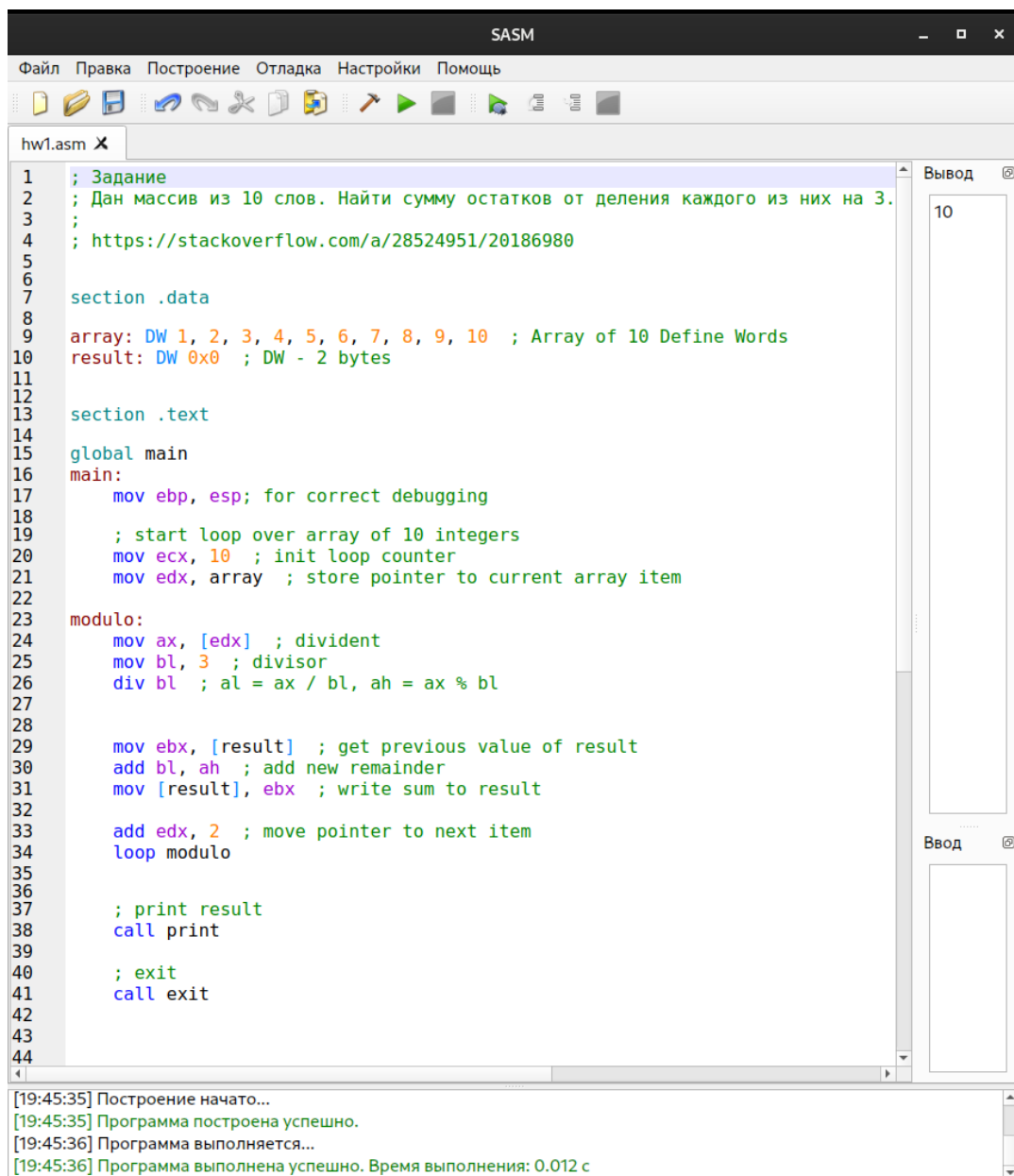
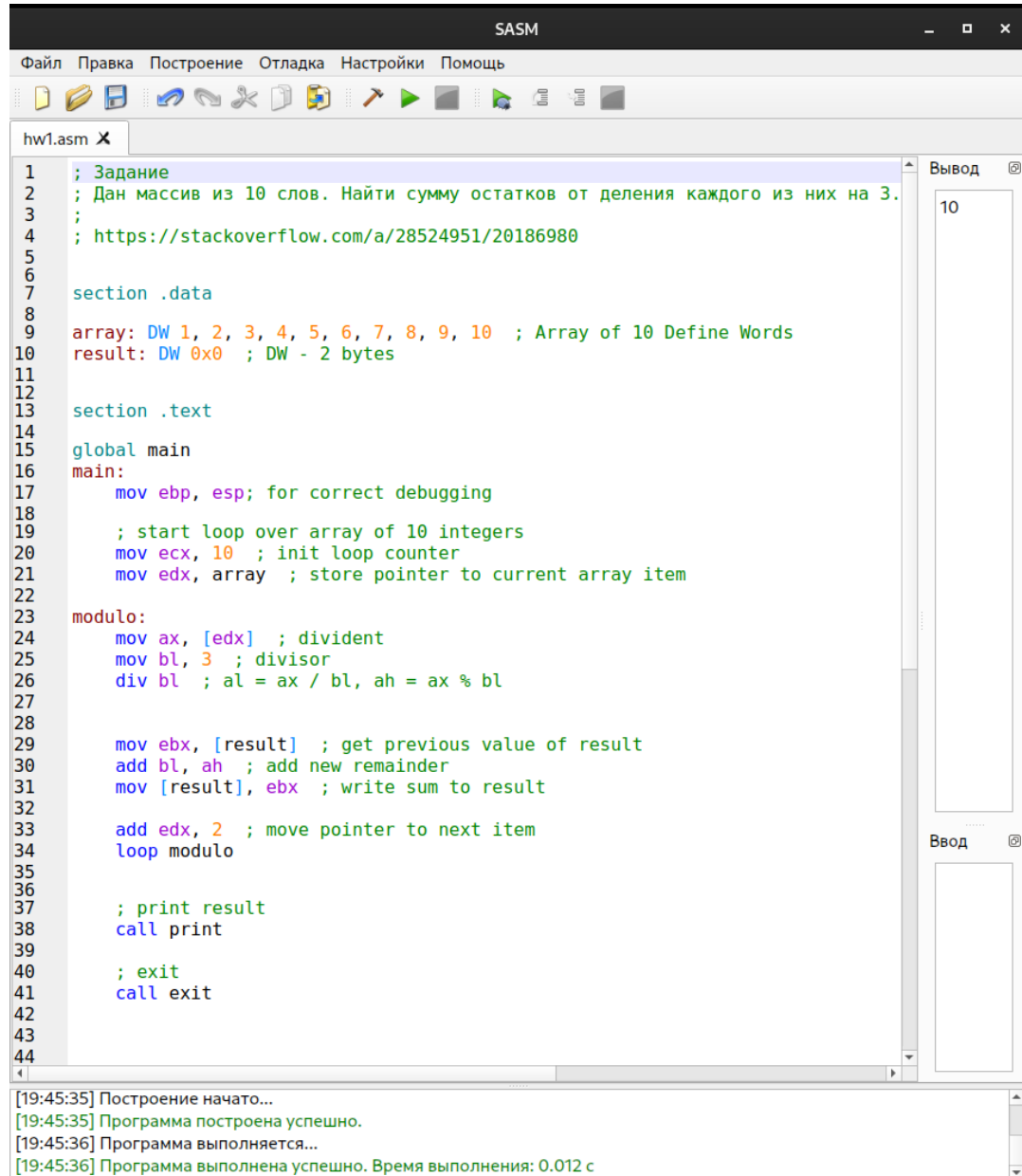


Рисунок 1 – Пример запуска программы

2.2 Работа программы

Продemonстрируем работу программы с различными входными данными. В качестве входных данных выступает только массив с числами. Будем инициализировать его различными числами. Результат на рисунках 2 — 4.



The screenshot shows the SASM (Simple Assembler) interface. The main window displays the assembly code for a file named `hw1.asm`. The code is as follows:

```
1 ; Задание
2 ; Дан массив из 10 слов. Найти сумму остатков от деления каждого из них на 3.
3 ;
4 ; https://stackoverflow.com/a/28524951/20186980
5
6
7 section .data
8
9 array: DW 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ; Array of 10 Define Words
10 result: DW 0x0 ; DW - 2 bytes
11
12
13 section .text
14
15 global main
16 main:
17     mov ebp, esp; for correct debugging
18
19     ; start loop over array of 10 integers
20     mov ecx, 10 ; init loop counter
21     mov edx, array ; store pointer to current array item
22
23 modulo:
24     mov ax, [edx] ; dividend
25     mov bl, 3 ; divisor
26     div bl ; al = ax / bl, ah = ax % bl
27
28
29     mov ebx, [result] ; get previous value of result
30     add bl, ah ; add new remainder
31     mov [result], ebx ; write sum to result
32
33     add edx, 2 ; move pointer to next item
34     loop modulo
35
36
37     ; print result
38     call print
39
40     ; exit
41     call exit
42
43
44
```

On the right side of the interface, there is a "Вывод" (Output) window showing the result of the program execution: `10`. Below the main window, there is a status bar with the following messages:

```
[19:45:35] Построение начато...
[19:45:35] Программа построена успешно.
[19:45:36] Программа выполняется...
[19:45:36] Программа выполнена успешно. Время выполнения: 0.012 с
```

Рисунок 2 – Массив чисел от 1 до 10. Результат — 10

The screenshot shows the SASM (Simple Assembler) interface. The main window displays the assembly code for a file named `hw1.asm`. The code is as follows:

```
1 ; Задание
2 ; Дан массив из 10 слов. Найти сумму остатков от деления каждого из них на 3.
3 ;
4 ; https://stackoverflow.com/a/28524951/20186980
5
6
7 section .data
8
9 array: DW 3, 3, 3, 3, 3, 3, 3, 3, 3, 3 ; Array of 10 Define Words
10 result: DW 0x0 ; DW - 2 bytes
11
12
13 section .text
14
15 global main
16 main:
17     mov ebp, esp; for correct debugging
18
19     ; start loop over array of 10 integers
20     mov ecx, 10 ; init loop counter
21     mov edx, array ; store pointer to current array item
22
23 modulo:
24     mov ax, [edx] ; dividend
25     mov bl, 3 ; divisor
26     div bl ; al = ax / bl, ah = ax % bl
27
28
29     mov ebx, [result] ; get previous value of result
30     add bl, ah ; add new remainder
31     mov [result], ebx ; write sum to result
32
33     add edx, 2 ; move pointer to next item
34     loop modulo
35
36
37     ; print result
38     call print
39
40     ; exit
41     call exit
42
43
44
```

On the right side of the interface, there is a "Вывод" (Output) window showing the result of the program execution, which is `0`. Below the main code window, there is a status bar with the following messages:

```
[20:08:50] Программа построена успешно.
[20:08:52] Программа выполняется...
[20:08:52] Программа выполнена успешно. Время выполнения: 0.011 с
```

Рисунок 3 – Массив из троек. Результат — 0

```
1 ; Задание
2 ; Дан массив из 10 слов. Найти сумму остатков от деления каждого из них на 3.
3 ;
4 ; https://stackoverflow.com/a/28524951/20186980
5
6
7 section .data
8
9 array: DW 30, 0, 0, 0, 1, 2, 33, 33, 33, 33 ; Array of 10 Define Words
10 result: DW 0x0 ; DW - 2 bytes
11
12
13 section .text
14
15 global main
16 main:
17     mov ebp, esp; for correct debugging
18
19     ; start loop over array of 10 integers
20     mov ecx, 10 ; init loop counter
21     mov edx, array ; store pointer to current array item
22
23 modulo:
24     mov ax, [edx] ; dividend
25     mov bl, 3 ; divisor
26     div bl ; al = ax / bl, ah = ax % bl
27
28
29     mov ebx, [result] ; get previous value of result
30     add bl, ah ; add new remainder
31     mov [result], ebx ; write sum to result
32
33     add edx, 2 ; move pointer to next item
34     loop modulo
35
36
37     ; print result
38     call print
39
40     ; exit
41     call exit
42
43
44
```

Вывод

3

Ввод

[20:10:01] Программа построена успешно.
[20:10:02] Программа выполняется...
[20:10:02] Программа выполнена успешно. Время выполнения: 0.022 с

Рисунок 4 – Случайный массив. Результат — 3

3 Выводы о проделанной работе

В ходе работы я реализовал программу, подсчитывающую сумму остатков от деления на 3 элементов массива длиной 10. Программа написана на ассемблере. В программе используется синтаксис Intel, а также реализован вывод в консоль без подключения библиотек.

Приложение А

```
1 ; Задание
2 ; Дан массив из 10 слов. Найти сумму остатков от деления каждого из них на 3.
   Результат поместить в отдельный элемент данных.
3 ;
4 ; https://stackoverflow.com/a/28524951/20186980
5
6
7 section .data
8
9 array: DW 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ; Array of 10 Define Words
10 result: DW 0x0 ; DW - 2 bytes
11
12
13 section .text
14
15 global main
16 main:
17     mov ebp, esp; for correct debugging
18
19     ; start loop over array of 10 integers
20     mov ecx, 10 ; init loop counter
21     mov edx, array ; store pointer to current array item
22
23 modulo:
24     mov ax, [edx] ; dividend
25     mov bl, 3 ; divisor
26     div bl ; al = ax / bl, ah = ax % bl
27
28
29     mov ebx, [result] ; get previous value of result
30     add bl, ah ; add new remainder
31     mov [result], ebx ; write sum to result
32
33     add edx, 2 ; move pointer to next item
34     loop modulo
35
36
37     ; print result
38     call print
```

```

39
40     ; exit
41     call exit
42
43
44
45 print:
46     ; Convert EAX to ASCII and store it onto the stack
47     xor eax, eax
48     mov ax, [result] ; store result to cleared EAX
49     sub esp, 16 ; reserve space on the stack
50     mov ecx, 10 ; divisor = 10
51     mov ebx, 16 ; stack shift pointer
52
53 extract_digit:
54     xor edx, edx ; Don't forget it!
55     div ecx ; Extract the last decimal digit
56     add dl, '0' ; Convert remainder to ASCII
57     sub ebx, 1
58     mov [esp+ebx], dl ; Store remainder on the stack (reverse order)
59     test eax, eax ; Until there is nothing left to divide
60     jnz extract_digit
61
62     mov eax, 4 ; SYS_WRITE
63     lea ecx, [esp+ebx] ; Pointer to the first ASCII digit,
https://stackoverflow.com/a/1665570/20186980
64     mov edx, 16
65     sub edx, ebx ; Count of digits
66     mov ebx, 1 ; STDOUT
67     int 0x80 ; Call 32-bit Linux
68
69     add esp, 16 ; Restore the stack
70
71
72 exit:
73     mov eax, 1 ; SYS_EXIT
74     xor ebx, ebx ; Return value
75     int 0x80 ; Call 32-bit Linux

```