



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

ОТЧЕТ

по лабораторной работе № 1

Дисциплина: М3ЯиОК

Студент

ИУ6-43Б
(Группа)

B.K. Залыгин
(Подпись, дата)

Преподаватель

(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Цель

Изучение процессов создания, запуска и отладки программ на ассемблере Nasm под управлением операционной системы Linux, а также особенностей описания и внутреннего представления данных.

Выполнение

Выполнить трансляцию программы. В результате получен объектный файл и листинг. Список файлов приведен на рисунке 1.

```
● vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ nasm -f elf64 lab1.asm -l lab1.lst
● vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ ll
total 24
drwxrwxr-x 2 vzalygin vzalygin 4096 Mar 12 13:21 .
drwxrwxr-x 3 vzalygin vzalygin 4096 Mar 12 13:20 ..
-rw-rw-r-- 1 vzalygin vzalygin 1052 Mar 11 22:26 lab1.asm
-rw-rw-r-- 1 vzalygin vzalygin 2197 Mar 12 13:21 lab1.lst
-rw-rw-r-- 1 vzalygin vzalygin 1072 Mar 12 13:21 lab1.o
-rw-rw-r-- 1 vzalygin vzalygin 338 Mar 11 22:28 Makefile
○ vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ █
```

Рисунок 1 – Результат трансляции

Выполнить линковку программы. В результате получен исполняемый файл. Список файлов приведен на рисунке 2.

```
● vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ ld -o lab1 lab1.o
● vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ ll
total 36
drwxrwxr-x 2 vzalygin vzalygin 4096 Mar 12 13:22 .
drwxrwxr-x 3 vzalygin vzalygin 4096 Mar 12 13:20 ..
-rwxrwxr-x 1 vzalygin vzalygin 9016 Mar 12 13:22 lab1*
-rw-rw-r-- 1 vzalygin vzalygin 1053 Mar 12 13:22 lab1.asm
-rw-rw-r-- 1 vzalygin vzalygin 2197 Mar 12 13:21 lab1.lst
-rw-rw-r-- 1 vzalygin vzalygin 1072 Mar 12 13:21 lab1.o
-rw-rw-r-- 1 vzalygin vzalygin 338 Mar 11 22:28 Makefile
○ vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ █
```

Рисунок 2 – Результат компоновки

Выполнить запуск программы. Рисунок 3 иллюстрирует работу программы.

```
● vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ ./lab1
Press Enter to Exit
123
○ vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ █
```

Рисунок 3 – Выполнение программы

Открыть edb, а в нем открыть программы lab1. В окне, представленном на рисунке 4, расположены листинг секции кода, показан на рисунке 5, представление в машинном коде, изображено на рисунке 6, содержание регистров, представлено на рисунке 7, содержание памяти, представлено на рисунке 8.

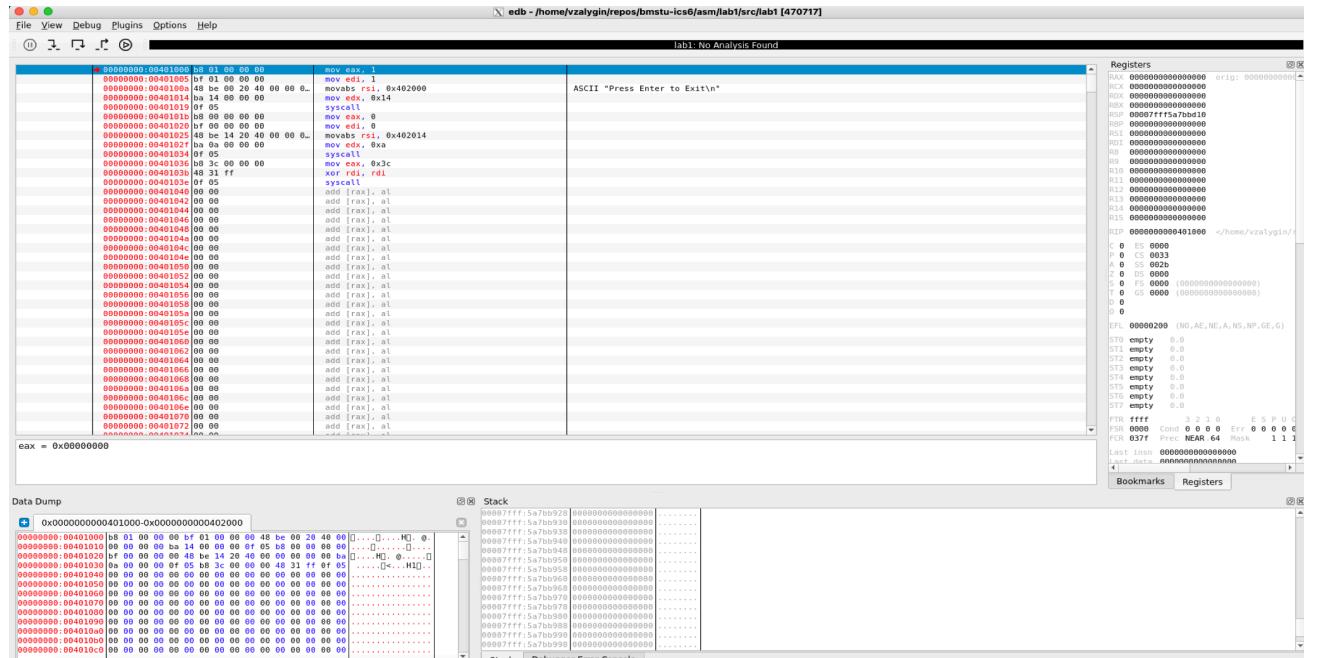


Рисунок 4 – Окно отладчика

```

    mov eax, 1
    mov edi, 1
    movabs rsi, 0x402000
    mov edx, 0x14
    syscall
    mov eax, 0
    mov edi, 0
    movabs rsi, 0x402014
    mov edx, 0xa
    syscall
    mov eax, 0x3c
    xor rdi, rdi
    syscall
    add [rax], al
    add [rax], al

```

Рисунок 5 – Секция кода

00000000:00401000	b8 01 00 00 00
00000000:00401005	bf 01 00 00 00
00000000:0040100a	48 be 00 20 40 00 00 0...
00000000:00401014	ba 14 00 00 00
00000000:00401019	0f 05
00000000:0040101b	b8 00 00 00 00
00000000:00401020	bf 00 00 00 00
00000000:00401025	48 be 14 20 40 00 00 0...
00000000:0040102f	ba 0a 00 00 00
00000000:00401034	0f 05
00000000:00401036	b8 3c 00 00 00
00000000:0040103b	48 31 ff
00000000:0040103e	0f 05
00000000:00401040	00 00
00000000:00401042	00 00
00000000:00401044	00 00
00000000:00401046	00 00
00000000:00401048	00 00
00000000:0040104a	00 00
00000000:0040104c	00 00
00000000:0040104e	00 00
00000000:00401050	00 00
00000000:00401052	00 00
00000000:00401054	00 00
00000000:00401056	00 00

Рисунок 6 – Представление кодов в шестнадцатеричном виде

Registers	
RAX	0000000000000000 orig: 0000000000
RCX	0000000000000000
RDX	0000000000000000
RBX	0000000000000000
RSP	00007ffe1e0f2bc0
RBP	0000000000000000
RSI	0000000000000000
RDI	0000000000000000
R8	0000000000000000
R9	0000000000000000
R10	0000000000000000
R11	0000000000000000
R12	0000000000000000
R13	0000000000000000
R14	0000000000000000
R15	0000000000000000
RIP	0000000000401000 </home/vzalygin/r
C	0 ES 0000
P	0 CS 0033
A	0 SS 002b
Z	0 DS 0000
S	0 FS 0000 (0000000000000000)
T	0 GS 0000 (0000000000000000)
D	0
O	0
CC: 00000000 7M AC NC A MC MD FC RI	

Рисунок 7 – Состояние регистров

0000000000:00402000	50 72 65 73 73 20 45 6e 74 65 72 20 74 6f 20 45	Press Enter to Exit
0000000000:00402010	78 69 74 0a 0a 00 00 00 00 00 00 00 06 00 00 00*
0000000000:00402020	00 00 00 00 2a 00 00 00 00 00 00 00 02 00 00 00
0000000000:00402030	00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00
0000000000:00402040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:00402050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:00402060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:00402070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:00402080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:00402090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:004020a0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:004020b0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000:004020c0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Рисунок 8 – Состояние памяти

Далее приведено выполнение программы построчно. На 9 рисунке и изображен результат применения первой строчки. В первом строчке содержится команда mov eax, 1, которая записывает в регистр eax единичку. В результате в регистр записалось значение 1.

```

0000000000:00401000 b8 01 00 00 00 mov eax, 1
0000000000:00401005 bf 01 00 00 00 mov edi, 1
0000000000:00401010 48 be 00 20 40 00 00 00 movabs rsi, 0x402000
0000000000:00401014 ba 14 00 00 00 mov edx, 0x14
0000000000:00401019 0f 05 syscall
0000000000:0040101b b8 00 00 00 00 mov eax, 0
0000000000:0040101d 48 8b 00 20 40 00 00 00 movabs rsi, 0x402014
0000000000:00401025 48 be 14 20 40 00 00 00 mov edx, 0xa
0000000000:00401027 ba 0a 00 00 00 mov edi, 0xa
0000000000:0040102f 0f 05 syscall
0000000000:00401034 0f 05
0000000000:00401036 b8 3c 00 00 00 mov eax, 0x3c
0000000000:00401038 48 31 ff xor rdi, rdi
0000000000:0040103a 0f 05 syscall
0000000000:00401042 0f 05
0000000000:00401044 0f 05
0000000000:00401045 0f 05
0000000000:00401048 0f 05
0000000000:0040104a 0f 05
0000000000:0040104c 0f 05
0000000000:0040104e 0f 05
0000000000:00401050 0f 05
0000000000:00401052 0f 05
0000000000:00401054 0f 05
0000000000:00401056 0f 05

```

Рисунок 9 – Проход через первую строчку

Вторая команда записывает в регистр edi значение 1, изображено на рисунке 10.

```

0000000000:00401000 b8 01 00 00 00 mov eax, 1
0000000000:00401005 bf 01 00 00 00 mov edi, 1
0000000000:0040100a 48 be 00 20 40 00 00 00 movabs rsi, 0x402000
0000000000:00401014 ba 14 00 00 00 mov edx, 0x14
0000000000:00401019 0f 05 syscall
0000000000:0040101b b8 00 00 00 00 mov eax, 0
0000000000:00401025 48 be 14 20 40 00 00 00 movabs rsi, 0x402014
0000000000:00401027 ba 0a 00 00 00 mov edi, 0xa
0000000000:0040102f 0f 05 syscall
0000000000:00401034 0f 05
0000000000:00401036 b8 3c 00 00 00 mov eax, 0x3c
0000000000:00401038 48 31 ff xor rdi, rdi
0000000000:0040103a 0f 05
0000000000:00401040 0f 05
0000000000:00401042 0f 05
0000000000:00401044 0f 05
0000000000:00401046 0f 05
0000000000:00401048 0f 05
0000000000:0040104a 0f 05
0000000000:0040104c 0f 05
0000000000:0040104e 0f 05
0000000000:00401050 0f 05
0000000000:00401052 0f 05
0000000000:00401054 0f 05
0000000000:00401056 0f 05

```

Рисунок 10 – Проход через вторую строчку

Третья команда записывает в регистр rsi адрес 0x40200 – адрес, по которому записана ASCII-строка Press Enter to Exit. Результат применения команды изображен на рисунке 11.

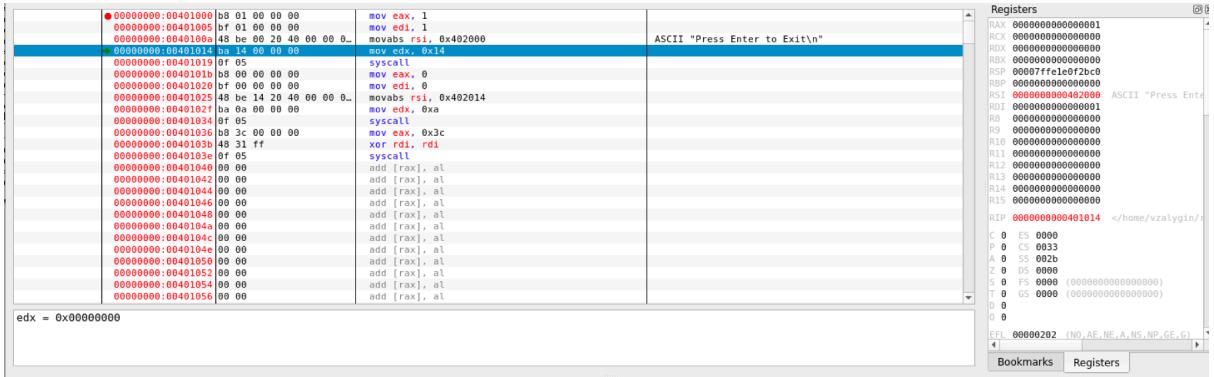


Рисунок 11 – Проход через третью строчку

В регистр edx записывается значение 0x14 – длина строки, которая будет выводиться. Состояние после операции показано на рисунке 12.

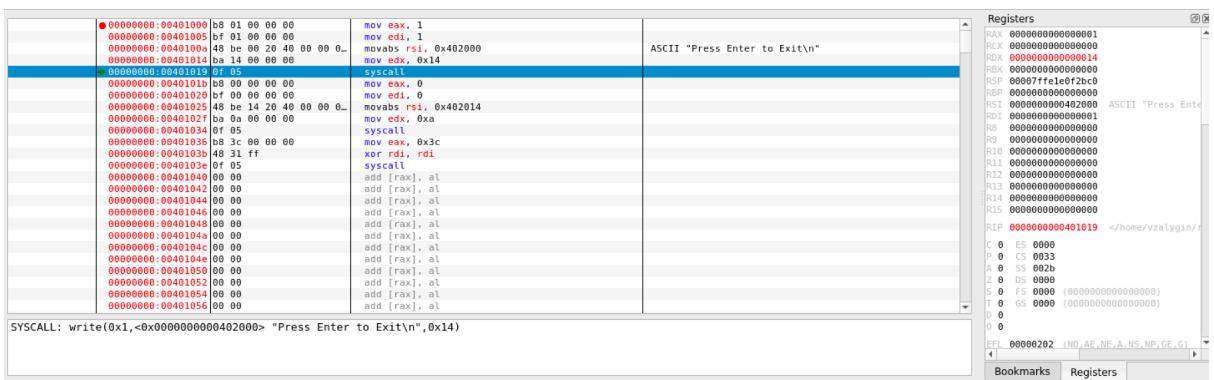


Рисунок 12 – Проход через 4 строчку

Вызывается системная функция (используется инструкция syscall). Тогда в окне консоли выводится текст, показано на рисунке 13.

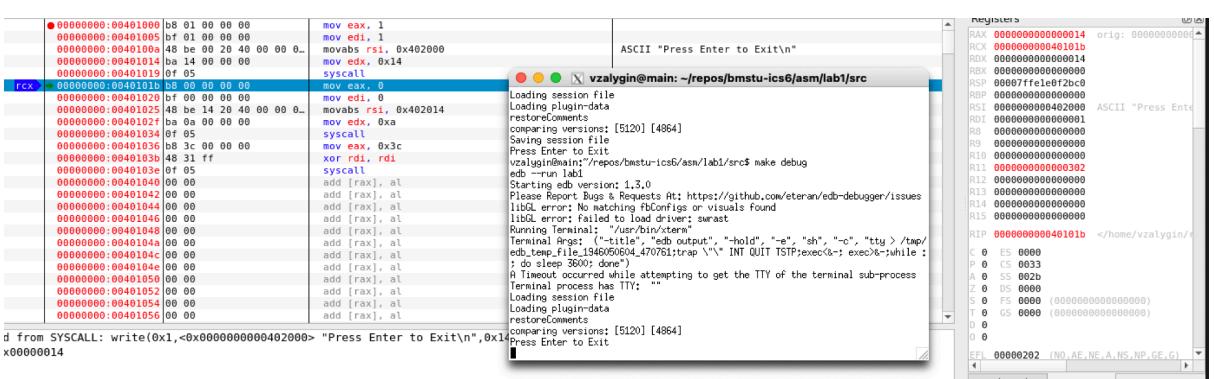


Рисунок 13 – Проход через 5 строчку

Изменить программу для вычисления результата выражения $X = A + 5 - B$. Сохраните программу с тем же именем, затем выполните ее трансляцию, компоновку и загрузку в отладчик. Зафиксируйте изменение программы в отчете.

Переписанная программа представлена на рисунке 13.

```
1      section .data
2      A dw -30
3      B dw 21
4      section .bss
5      X resd 1
6      section .text
7      global _start
8      _start: ; X = A + 5 - B
9          mov EAX, [A]
10         add EAX,5
11         sub EAX, [B]
12         mov [X],EAX
13
```

Рисунок 13 – Код программы

Сохраните программу с тем же именем, затем выполните ее трансляцию, компоновку и загрузку в отладчик.

Процесс выполнения трансляции и компоновки показан на рисунке 14.

```
vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ nasm -f elf64 lab1.asm -l lab1.lst && ld -o lab1 lab1.o
vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$ ll
total 36
drwxrwxr-x 2 vzalygin vzalygin 4096 Mar 31 20:30 .
drwxrwxr-x 3 vzalygin vzalygin 4096 Mar 31 20:29 ../
-rw-rwxr-x 1 vzalygin vzalygin 8952 Mar 31 20:30 lab1*
-rw-rw-r-- 1 vzalygin vzalygin 179 Mar 31 20:28 lab1.asm
-rw-rw-r-- 1 vzalygin vzalygin 659 Mar 31 20:30 lab1.lst
-rw-rw-r-- 1 vzalygin vzalygin 1024 Mar 31 20:30 lab1.o
-rw-rw-r-- 1 vzalygin vzalygin 338 Mar 11 22:28 Makefile
vzalygin@main:~/repos/bmstu-ics6/asm/lab1/src$
```

Рисунок 14 – Трансляция

Найдите в отладчике внутреннее представление исходных данных, отразите его в отчете и поясните. Проследите в отладчике выполнение программы и зафиксируйте в отчете результаты выполнения каждой добавленной команды (изменение регистров, флагов и полей данных).

На рисунке 15 показано, как программа отображается в отладчике. В окне дампа памяти видны значения, записанные в секцию инициализированных данных. Это двухбайтовые представления чисел -30 и 21 в дополнительном коде. В секцию bss записывается четырехбайтовое (double word) значение из регистра EAX – результат вычисления.

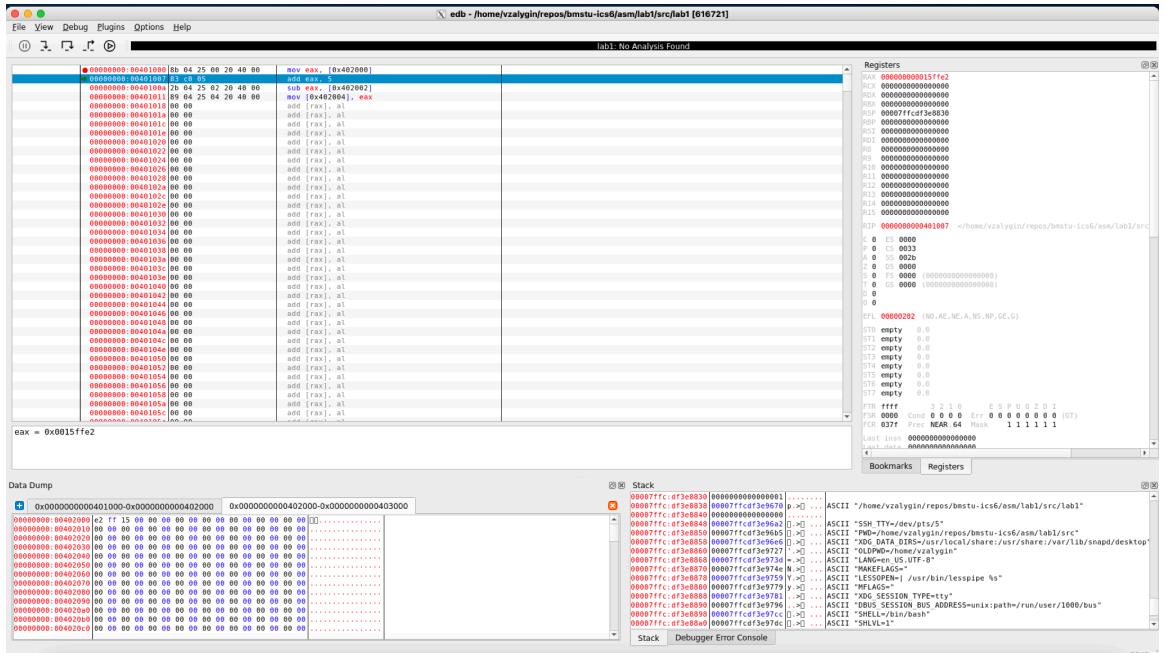


Рисунок 15 – Отображение программы в отладчике

Построчное выполнение команд представлено на рисунках 16, 17, 18.

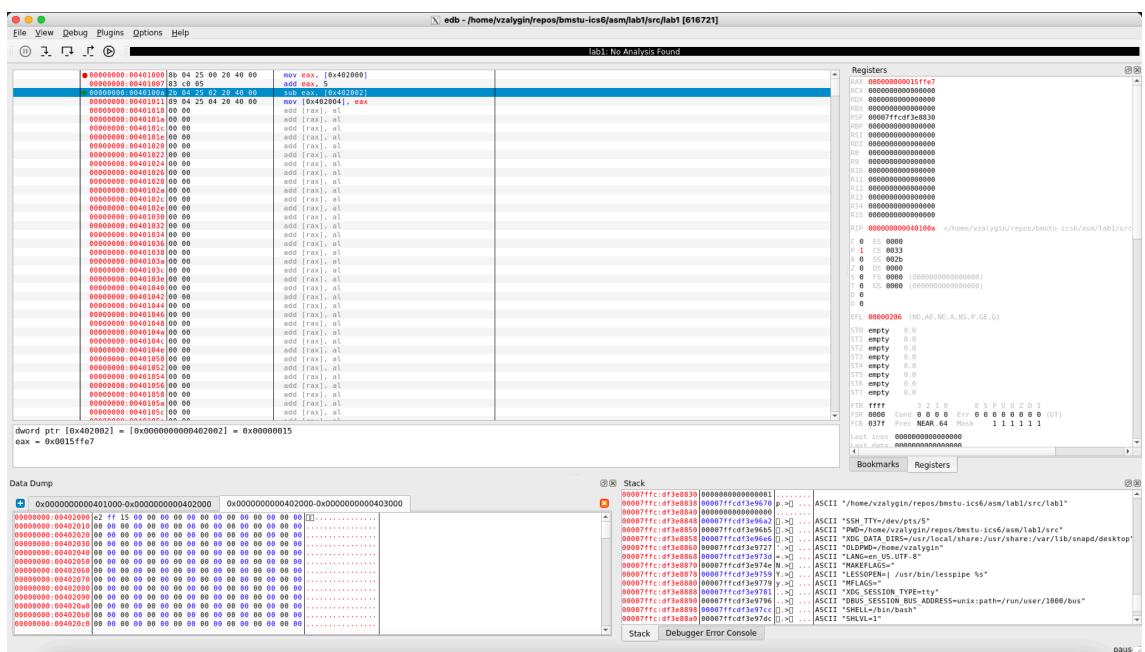


Рисунок 16 – Выполнение второй команды

Рисунок 17 – Выполнение третьей команды

Рисунок 18 – Выполнение пятой команды

Ведите следующие строки в разделы описания инициированных и неинициализированных данных и определите с помощью отладчика внутренне представление этих данных в памяти. Результаты проанализируйте и занесите в отчет.

```

val1 db 255
chart dw 256
lue3 dw -128
v5 db 10h
db 100101B
beta db 23,23h,0ch
sdk db "Hello",10
min dw -32767
ar dd 12345678h
valar times 5 db 8
alu resw 10
f1 resb 5

```

Результат введения данных строк представлен на рисунке 19. ff – значение val1, 00 01 – значение chart, 80 ff – значение lue3, 10 – значение v5, 25 – безымянное значение (100101b), 17 23 0c – значение beta, 48 65 6c 6c 6f 0a – значение sdk (hello,10), 01 80 – значение min, 78 56 34 12 – значение ar, 08 08 08 08 08 – значение valar. Наконец для символов alu и f1 зарезервировано 10 слов ($10 \times 2 = 20$ байтов) и 5 байтов соответственно.

	0x000000000000401000-0x000000000000402000	0x000000000000402000-0x000000000000403000
00000000:00402000	ff 00 01 80 ff 10 25 17 23 0c 48 65 6c 6c 6f 0a	...%.# Hello
00000000:00402010	01 80 78 56 34 12 08 08 08 08 08 00 00 00 00 00	...xV4.....
00000000:00402020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:004020a0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:004020b0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:004020c0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Рисунок 19 – Состояние памяти

Определите в памяти следующие данные:

- целое число 25 размером 2 байта со знаком;
- двойное слово, содержащее число -35;

в) символьную строку, содержащую ваше имя (русскими буквами и латинскими буквами).

Зафиксируйте в отчете описание и внутреннее представление этих данных и дайте пояснение.

Определение данных изображено на рисунке 20. Представление данных в памяти изображено на рисунке 21. 19 00 – значение a, dd ff ff ff – значение b, далее до символа 0a идет представление строки с. В данной строке 1 байт представляет 1 символ.

```
C > ASM lab1.asm
1 |     section .data
2 a dw 25
3 b dd -35
4 c db "Вячеслав Vyacheslav",10
5 |     section .bss
```

Рисунок 20 – Секция инициализированных данных

0x000000000000401000-0x000000000000402000		0x000000000000402000-0x000000000000403000	
00000000:00402000	19 00 dd ff ff c2 ff f7 e5 f1 eb e0 e2 20 56	..	V
00000000:00402010	79 61 63 68 65 73 6c 61 76 0a 00 00 00 00 00	yacheslav
00000000:00402020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402030	00 00 00 00 00 00 00 01 00 00 00 04 00 f1 ff
00000000:00402040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000:00402050	1f 00 00 00 00 00 02 00 00 20 40 00 00 00 00 00	@.....
00000000:00402060	00 00 00 00 00 00 00 00 0a 00 00 00 00 00 02 00
00000000:00402070	02 20 40 00 00 00 00 00 00 00 00 00 00 00 00 00	@.....
00000000:00402080	0c 00 00 00 00 00 02 00 06 20 40 00 00 00 00 00	@.....
00000000:00402090	00 00 00 00 00 00 00 00 13 00 00 00 10 00 01 00
00000000:004020a0	00 10 40 00 00 00 00 00 00 00 00 00 00 00 00 00	@.....
00000000:004020b0	0e 00 00 00 10 00 02 00 1a 20 40 00 00 00 00 00	@.....
00000000:004020c0	00 00 00 00 00 00 00 00 1a 00 00 00 10 00 02 00

Рисунок 21 – Представление данных

Определите несколькими способами в программе числа, которые во внутреннем представлении (в отладчике) будут выглядеть как 25 00 и 00 25. Проверьте правильность ваших предположений, введя соответствующие строки в программу.

Определение данных представлено на рисунке 22, их представление изображено на рисунке 23.

```
1 |      section .data
2 aa dw 37
3 ab dw 100101b
4 ac dw 25h
5
6 ba dw 9472
7 bb dw 100101000000000b
8 bc dw 2500h
```

Рисунок 22 – Секция инициализированных данных

0x00000000000401000-0x00000000000402000 0x00000000000000000000000000000000
00000000:00402000 25 00 25 00 25 00 00 25 00 25 00 25 00 25 00 25 00 |
00000000:00402010 AA |

Рисунок 23 – Представление данных

Добавьте в программу переменную F1=65535 размером слово и переменную F2= 65535 размером двойное слово. Вставьте в программу команды сложения этих чисел с 1:

```
add [F1],1
add [F2],1
```

Результаты выполнения операций представлены на рисунках 24 и 25. При выполнении первой операции произошло переполнение, тогда значение 2 байтов памяти F1 обнулилось, а в флаги переноса CF и флаг переполнения OF (на рисунке Z) выставились единички. При второй операции переполнения разрядной сетки не происходит, поэтому значение в памяти не обнуляется (оно

становится равным 00 00 01 00), а соответствующие флаги устанавливаются в

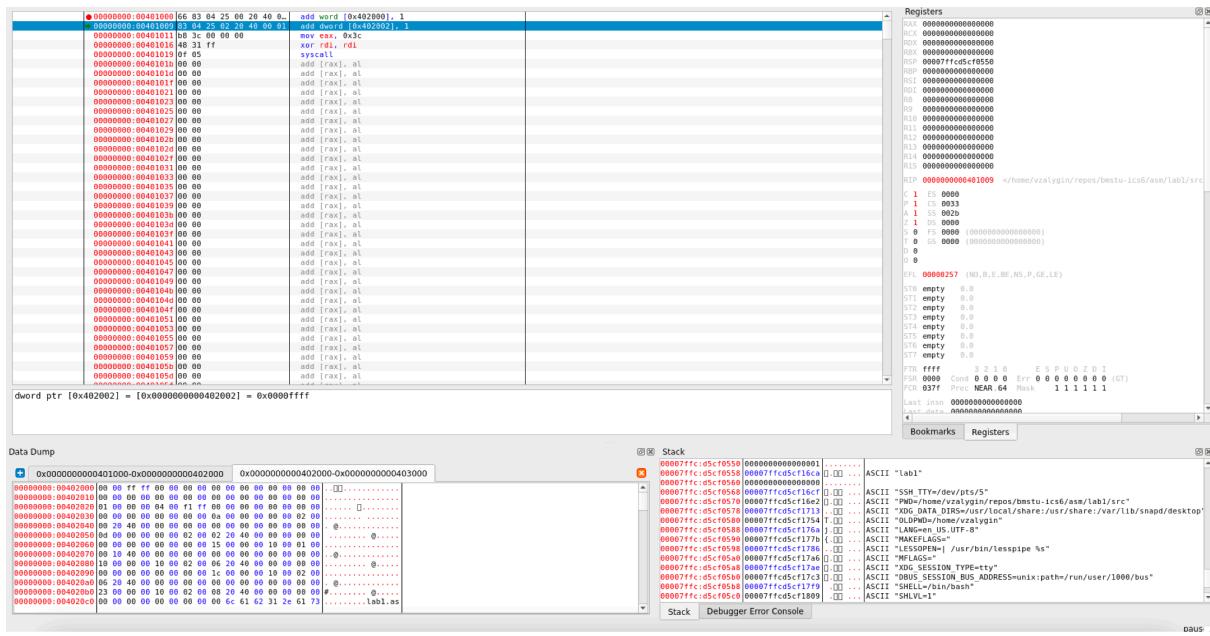


Рисунок 24 – Изменение флагов

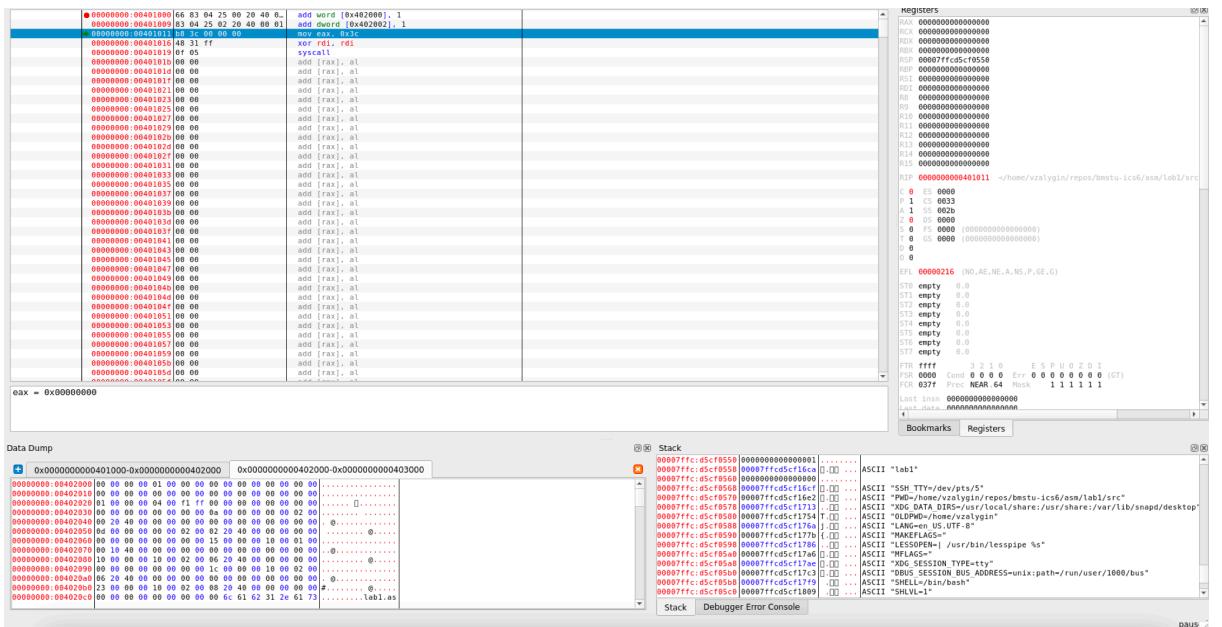


Рисунок 25 – Изменение флагов

Ответы на контрольные вопросы

1) Ассемблер – семейство языков программирования низкого уровня,

команды которых соответствуют одной или нескольким командам процессора.

2) Программа на ассемблере состоит из 3 секций: секция инициализированных данных, секция неинициализированных данных, секция кода.

3) Для запуска программы необходимо сначала транслировать ее в объектный файл, а потом скомпоновать в исполняемый файл. На этапе трансляции код превращается в последовательность машинных команд, зависящий от архитектуры процессора, задаются таблицы символов. На этапе компоновки происходит связывание частей программы в единый исполняемый файл.

Для выполнения команды отладчика используют следующие комбинации клавиш:

F7 – выполнить шаг с заходом в тело процедуры;

F8 – выполнить шаг, не заходя в тело процедуры.

Отладчик представляет числа в дополнительном коде в шестнадцатеричном представлении. Числа A dw 5,-5 будут представлены как 05 00, FB FF. После загрузки в регистр: 00 05, FF FB.

Выражения программируются при помощи использования регистра аккумулятора EAX (RAX, AX и т.д.).

mov AL,[A]

add AL,[B]

mov [C],AL