



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

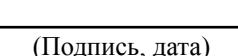
ИУ6-43Б

(Группа)


(Подпись, дата)

В.К. Залыгин
(И.О. Фамилия)

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 получим объектный файл и листинг (смотреть рисунок 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 получим исполняемый файл (смотреть рисунок 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. Проверим, что она корректно работает (смотреть рисунок 3)

```
● 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 (смотреть рисунок 4) расположены листинг секции кода (смотреть рисунок 5), представление в машинном коде (смотреть рисунок 6), содержание регистров (смотреть рисунок 7), содержание памяти (рисунок 8).

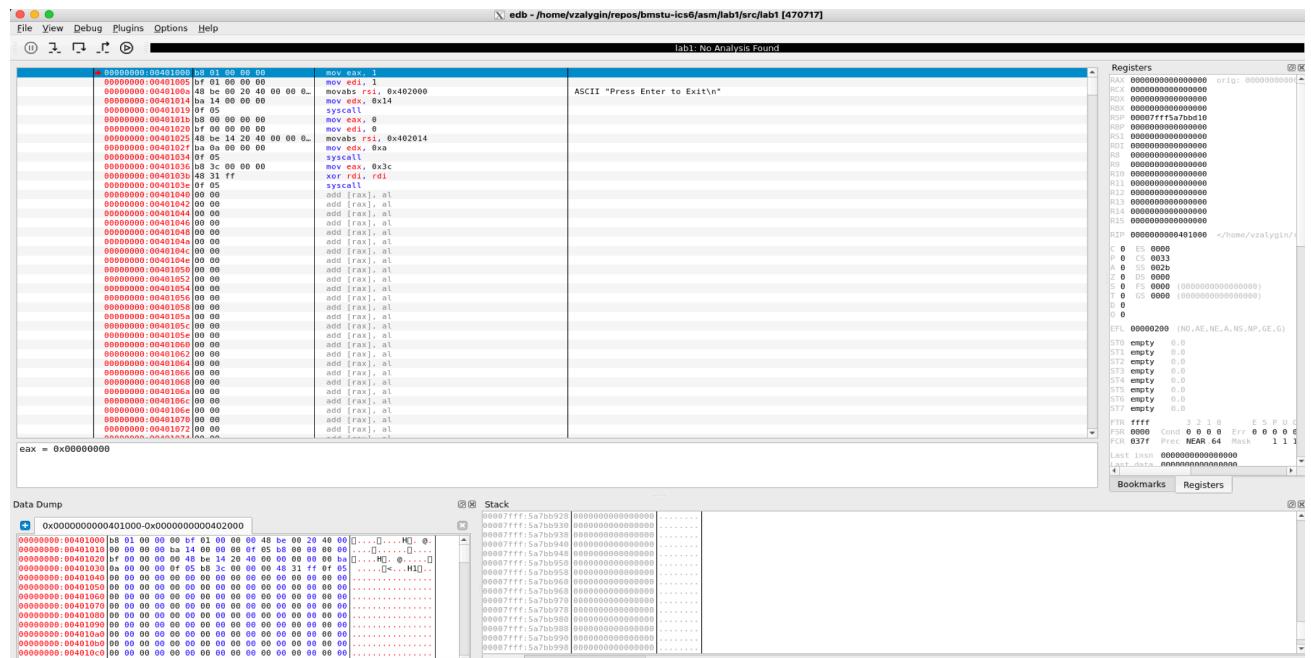


Рисунок 4 – окно отладчика

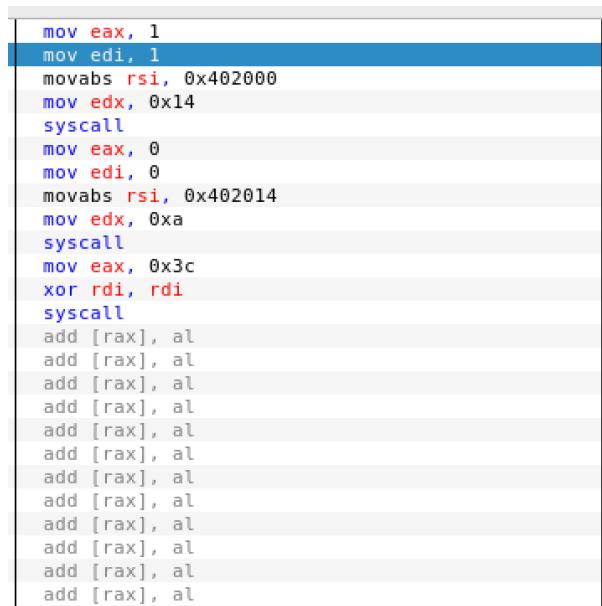


Рисунок 5 – секция кода

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 шестандацатеричном 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
cc1 00000000 r/m ac nc a nc m0 rc ri	

рисунок 7 – регистры

Рисунок 8 – память

Начнем построчно выполнять программу. Выполним первую строчку и убедимся, что изменения отразились в программе.

В первом строчке содержится команда `mov eax, 1`, которая записывает в регистр eax единичку. В результате в регистр действительно записалось значение 1 (смотреть рисунок 9).

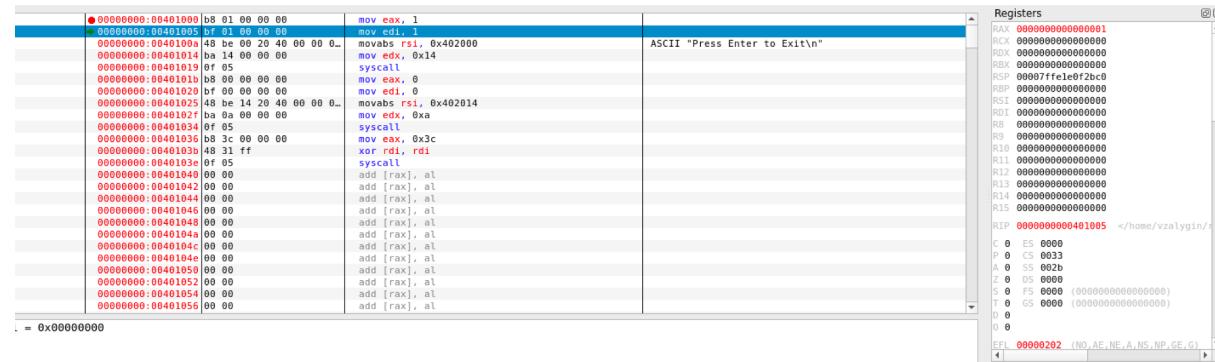


Рисунок 9 – проход через первую строчку

Вторая команда записывает в регистр edi значение 1 (смотреть рисунок 10).

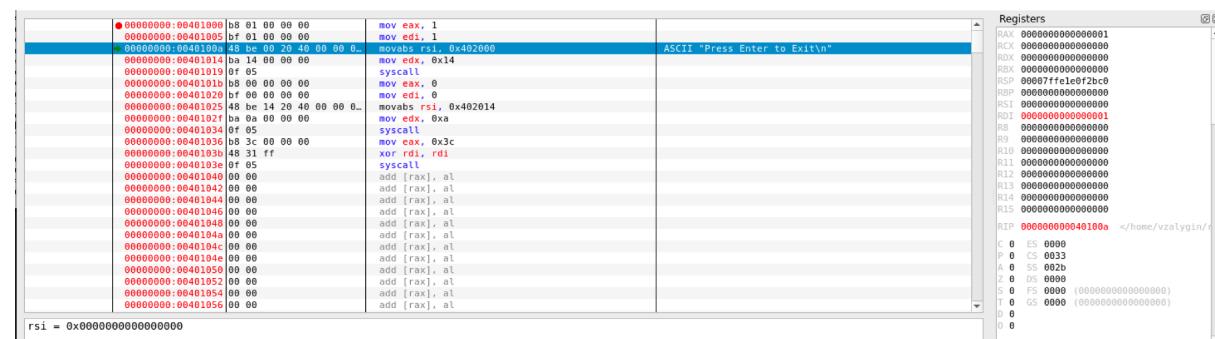


Рисунок 10 – проход через вторую строчку

Третья команда записывает в регистр rsi адрес 0x40200 – адрес, по которому записана ASCII-строка Press Enter to Exit (смотреть рисунок 11).

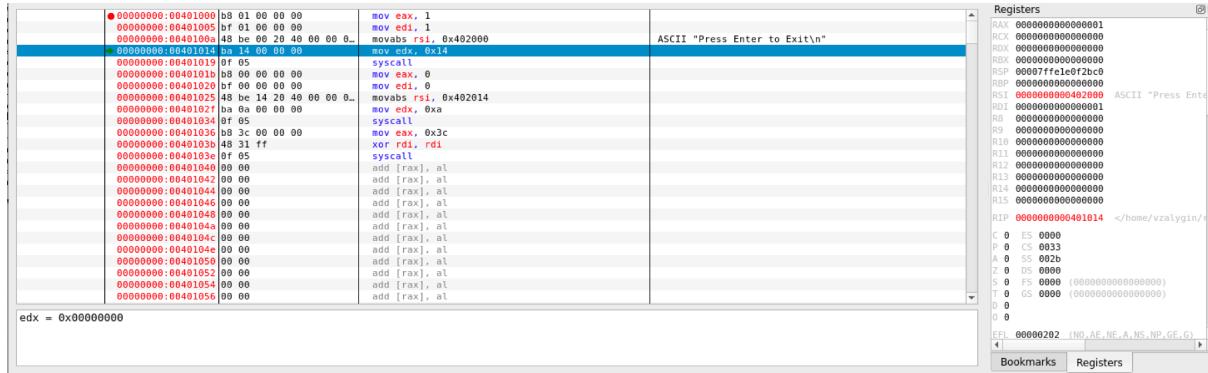


Рисунок 11 – проход через третью строчку

Наконец запишем в регистр edx значение 0x14 – длину строки, которая будет выводиться (смотреть рисунок 12).

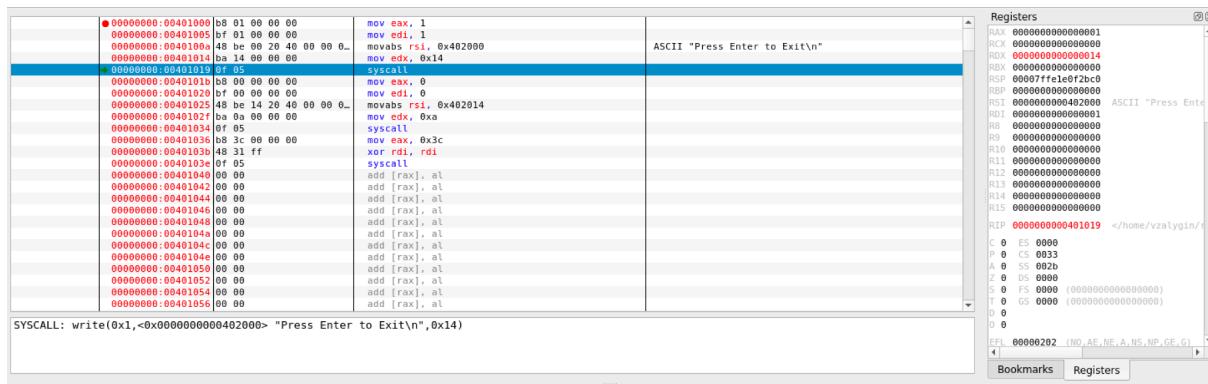


Рисунок 12 – проход через 4 строчку

Вызываем системную функцию (используем инструкцию syscall).

Тогда в окне консоли выводится текст (смотреть рисунок 13).

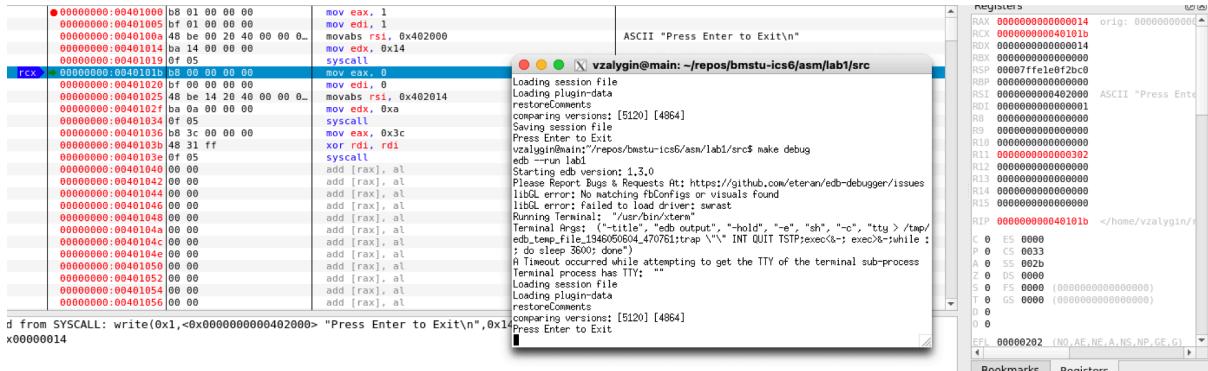


Рисунок 13 – проход через 5 строчку

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.

Перепишем программу, тогда получим следующий листинг (смотреть рисунок 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-rw-r-- 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.

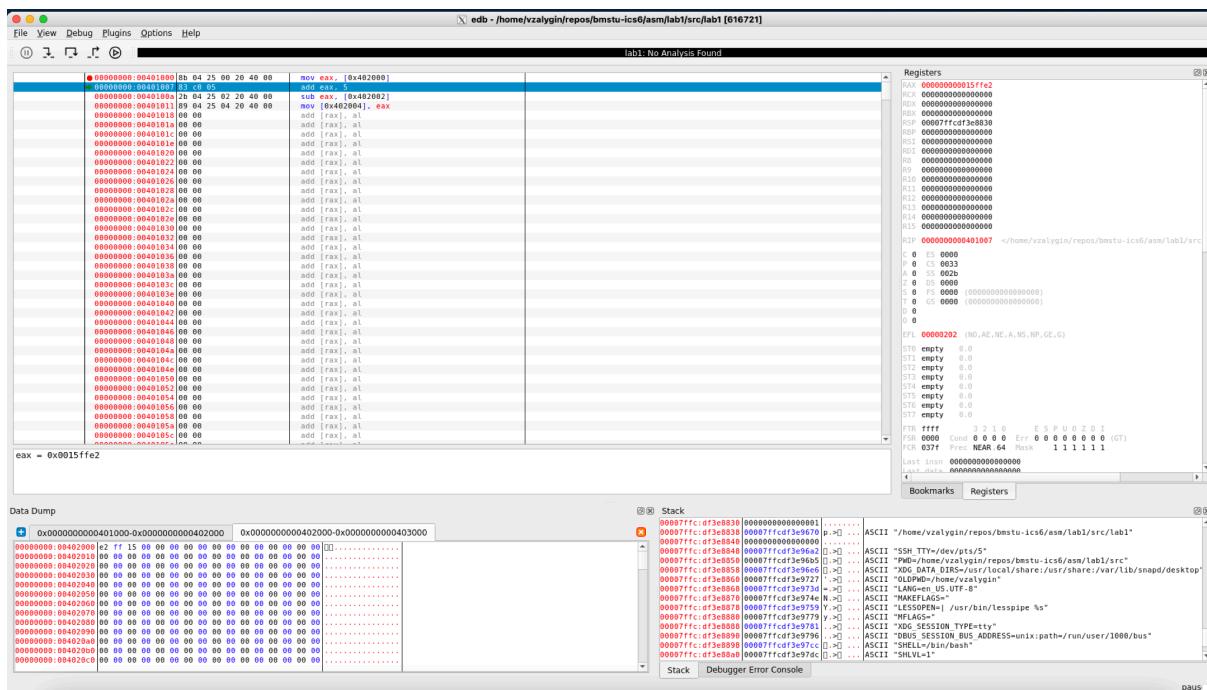


Рисунок 15 – отображение программы в отладчике

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

Registers

RAX	0000000000000000
RCX	0000000000000000
RSI	0000000000000000
RBX	0000000000000000
RSI	0000000000000000
RDI	0000000000000000
RBP	0000000000000000
RSP	0000000000000000
R8	0000000000000000
R9	0000000000000000
R10	0000000000000000
R11	0000000000000000
R12	0000000000000000
R13	0000000000000000
R14	0000000000000000
R15	0000000000000000
R16	0000000000000000
R17	0000000000000000
R18	0000000000000000
R19	0000000000000000
R20	0000000000000000
R21	0000000000000000
R22	0000000000000000
R23	0000000000000000
R24	0000000000000000
R25	0000000000000000
R26	0000000000000000
R27	0000000000000000
R28	0000000000000000
R29	0000000000000000
R30	0000000000000000
R31	0000000000000000

Registers

C	0	KS	0000
P	1	0333	
A	0	SS	0025
B	0	DS	0000
S	0	FS	0000 (0000000000000000)
T	0	GS	0000 (0000000000000000)
D	0		
O	0		

EFL 00000206 (NO,AE,NE,A,NO,P,GE,G)

ST0 empty 0.0

ST1 empty 0.0

ST2 empty 0.0

ST3 empty 0.0

ST4 empty 0.0

ST5 empty 0.0

ST6 empty 0.0

ST7 empty 0.0

FPR ffff

FSR 0000 Cond 0 0 0 Err 0 0 0 0 0 0 (GT)

FC 037 Prec NEAR 64 Mask 1 1 1 1 1 1

Last Inst 0000000000000000

Last Data 0000000000000000

1 Bookmarks Registers

Рисунок 16 – выполнение второй команды

Registers

RAX	0000000000000000
RCX	0000000000000000
RSI	0000000000000000
RBX	0000000000000000
RSI	0000000000000000
RDI	0000000000000000
RBP	0000000000000000
RSP	0000000000000000
R8	0000000000000000
R9	0000000000000000
R10	0000000000000000
R11	0000000000000000
R12	0000000000000000
R13	0000000000000000
R14	0000000000000000
R15	0000000000000000
R16	0000000000000000
R17	0000000000000000
R18	0000000000000000
R19	0000000000000000
R20	0000000000000000
R21	0000000000000000
R22	0000000000000000
R23	0000000000000000
R24	0000000000000000
R25	0000000000000000
R26	0000000000000000
R27	0000000000000000
R28	0000000000000000
R29	0000000000000000
R30	0000000000000000
R31	0000000000000000

Registers

C	0	KS	0000
P	1	0333	
A	0	SS	0025
B	0	DS	0000
S	0	FS	0000 (0000000000000000)
T	0	GS	0000 (0000000000000000)
D	0		
O	0		

EFL 00000206 (NO,AE,NE,A,NO,P,GE,G)

ST0 empty 0.0

ST1 empty 0.0

ST2 empty 0.0

ST3 empty 0.0

ST4 empty 0.0

ST5 empty 0.0

ST6 empty 0.0

ST7 empty 0.0

FPR ffff

FSR 0000 Cond 0 0 0 Err 0 0 0 0 0 0 (GT)

FC 037 Prec NEAR 64 Mask 1 1 1 1 1 1

Last Inst 0000000000000000

Last Data 0000000000000000

1 Bookmarks Registers

Рисунок 17 – выполнение третьей команды

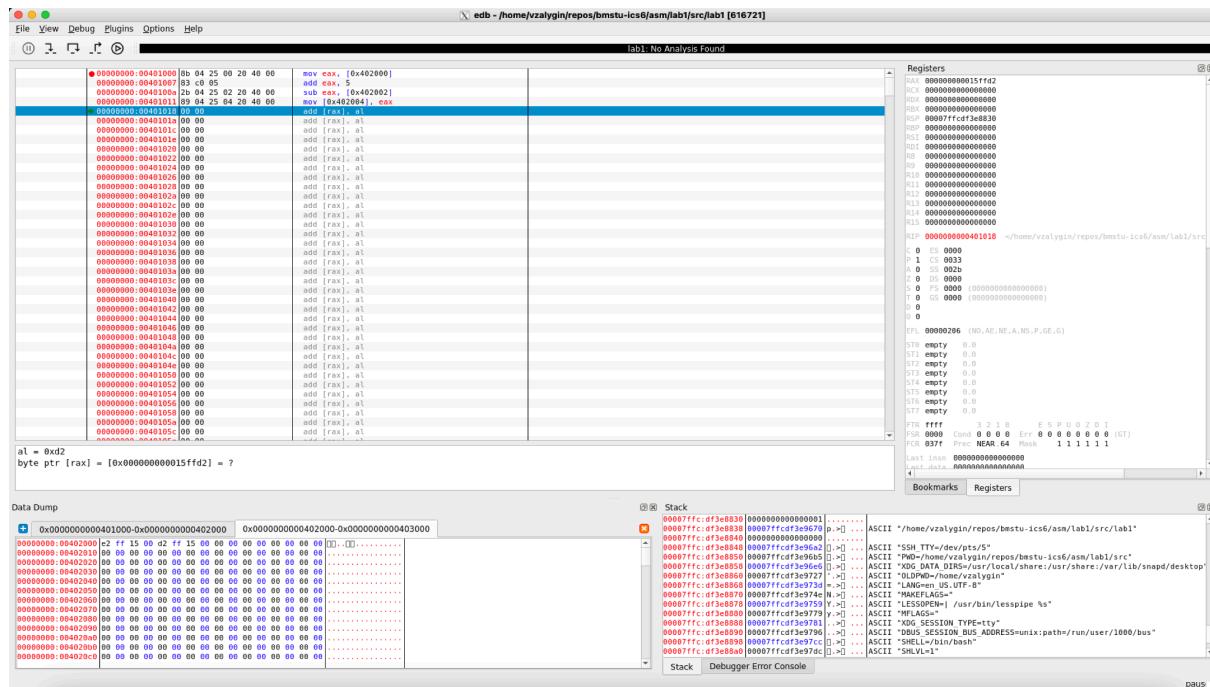


Рисунок 18 – выполнение пятой команды

Ведите следующие строки в разделы описания инициированных и неинициализированных данных и определите с помощью отладчика внутренние 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 представлена 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 – значение valar. Наконец для символов alu и f1 зарезервировано 10 слов (10*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 00 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;
- 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	.. Вячеслав 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 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 00 02
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 1001010b
4 |     ac dw 25h
5 |
6 |     ba dw 9472
7 |     bb dw 100101000000000b
8 |     bc dw 2500h
```

Рисунок 22 – секция инициализированных данных

+	0x000000000000401000-0x000000000000402000	0x00000000
00000000 : 00402000	25 00 25 00 25 00 00 25 00 25 00 25 00 1	
00000000 : 00402010	00 00 00 00 00 00 00 00 00 00 00 00 00 00	

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

0.

Dword ptr [0x402002] = [0x0000000000402000] = 0x0000ffff

Data Dump

Stack

Registers

Рисунок 24 – изменение флагов

cax = 0x00000000

Data Dump

Stack

Registers

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.
- 4) Для выполнения команды отладчика используют следующие комбинации клавиш:
F7 – выполнить шаг с заходом в тело процедуры;
F8 – выполнить шаг, не заходя в тело процедуры.
- 5) 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.
- 6) Выражения программируются при помощи использования регистра аккумулятора EAX (RAX, AX и т.д.).
`mov AL,[A]`
`add AL,[B]`
`mov [C],AL`