|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«*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\_\_\_\_\_\_\_\_

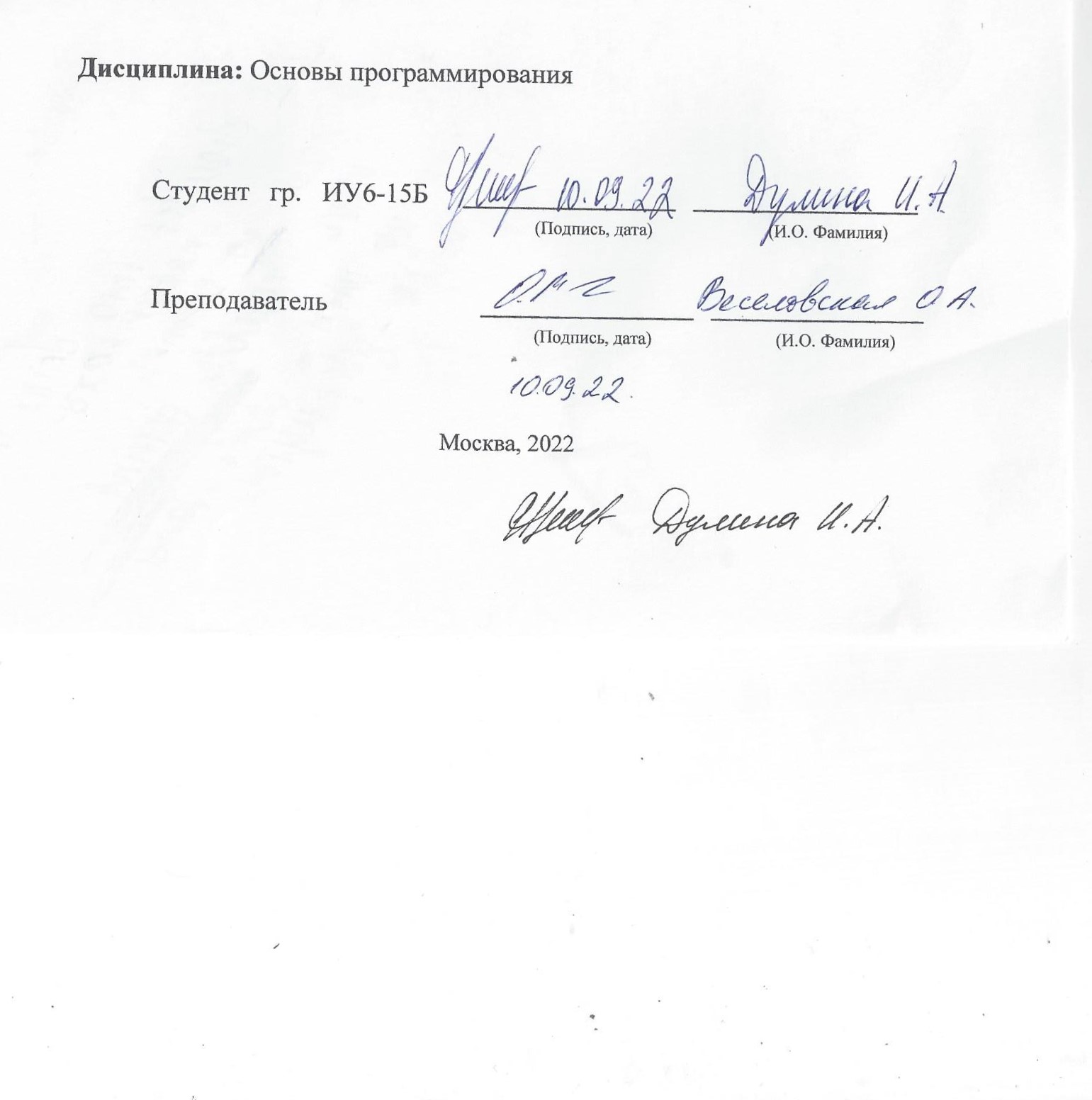
КАФЕДРА \_\_\_\_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 1**

**Дисциплина:** Машинно-зависимые языки и основы компиляции

**Название лабораторной работы:** Изучение среды и отладчика ассемблера



Студент гр. ИУ6-45Б\_\_\_\_\_\_19.03.2024**\_\_\_ \_\_\_**И.А.Дулина**\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_** С.С.Данилюк**\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2024

**Цель работы:** изучение процессов создания, запуска и отладки программ на ассемблере Nasm под управлением операционной системы Linux, а также особенностей описания и внутреннего представления данных.

**Задание:**

Для изучения возможностей отладчика добавьте в заготовку несколько команд для вычисления результата следующего выражения: X=A+5-B. 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.

На листинге 1 показан код программы.

Листинг 1 – Код программы для первого задания

|  |
| --- |
| section .data ;сегмент инициализированных переменных  ExitMsg db "Press Enter to Exit", 10 ;выводимое сообщение  lenExit equ $-ExitMsg  A DW -30  B DW 21  section .bss ;сегмент неинициализированных переменных  InBuf resb 10 ; буфер для вводимой строки  lenIn equ $-InBuf  X resd 1 ;зарезервировать место  section .text ;сегмент кода  global \_start  \_start:  ;  mov eax, [A] ;загрузим число в регистр eax  add eax, 5 ;сложим eax и 5, результат в eax  sub eax, [B] ;вычесть число B, результат в eax  mov [X], eax ; сохраним результат в памяти  ;write  mov rax,1 ;системная функция 1(write)  mov rdi, 1;дескрипторо файла stdout=1  mov rsi, ExitMsg;адрес выводимой строки  mov rdx, lenExit;длина строки  syscall;вызов системной функции  ;read  mov rax, 0;системная функция 0 (read)  mov rdi, 0;дескриптор файла stdin=0  mov rsi, InBuf;адрес вводимой строки  mov rdx, lenIn; длина строки  syscall; вызов системной функции  ;exit  mov rax, 60;системная функция 60 (exit)  xor rdi, rdi;return code 0  syscall;вызов системной функции |

На рисунке 1 показаны регистры после выполнения программы.

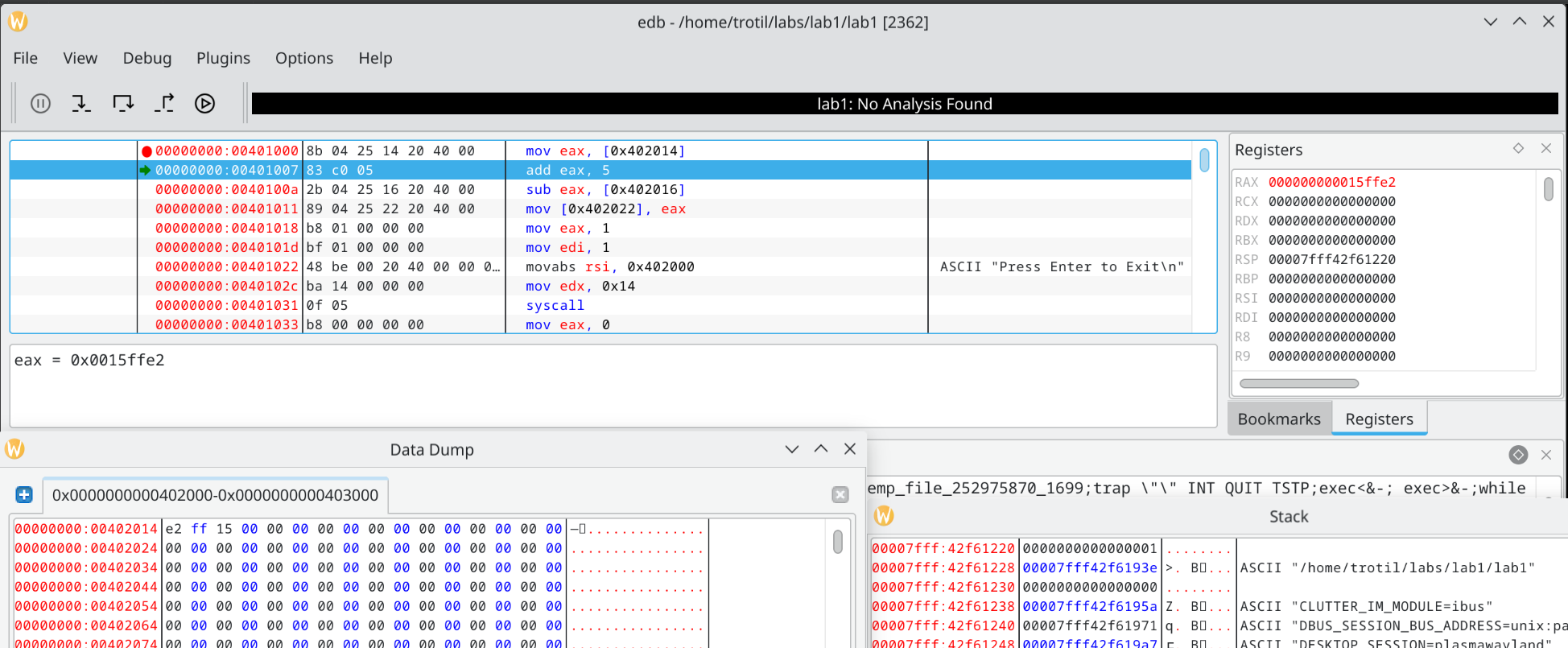




Рисунок 1 – Регистры и поле данных после выполнения mov eax, [A]

На рисунке 2 показано изменение регистра EAX после выполнения mov EAX, 5.

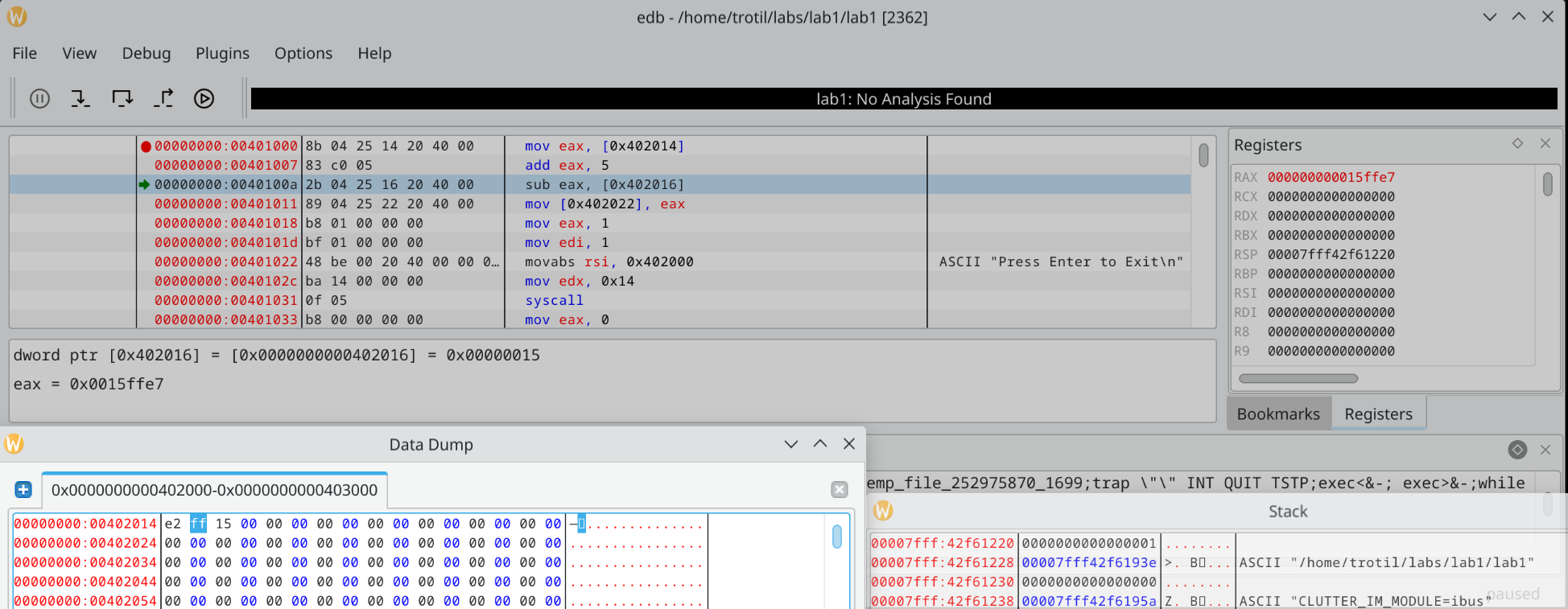




Рисунок 2 – Изменение регистра после выполнения mov eax, 5

На рисунке 3 показано изменение регистра EAX после выполнения команды SUB EAX, [B].

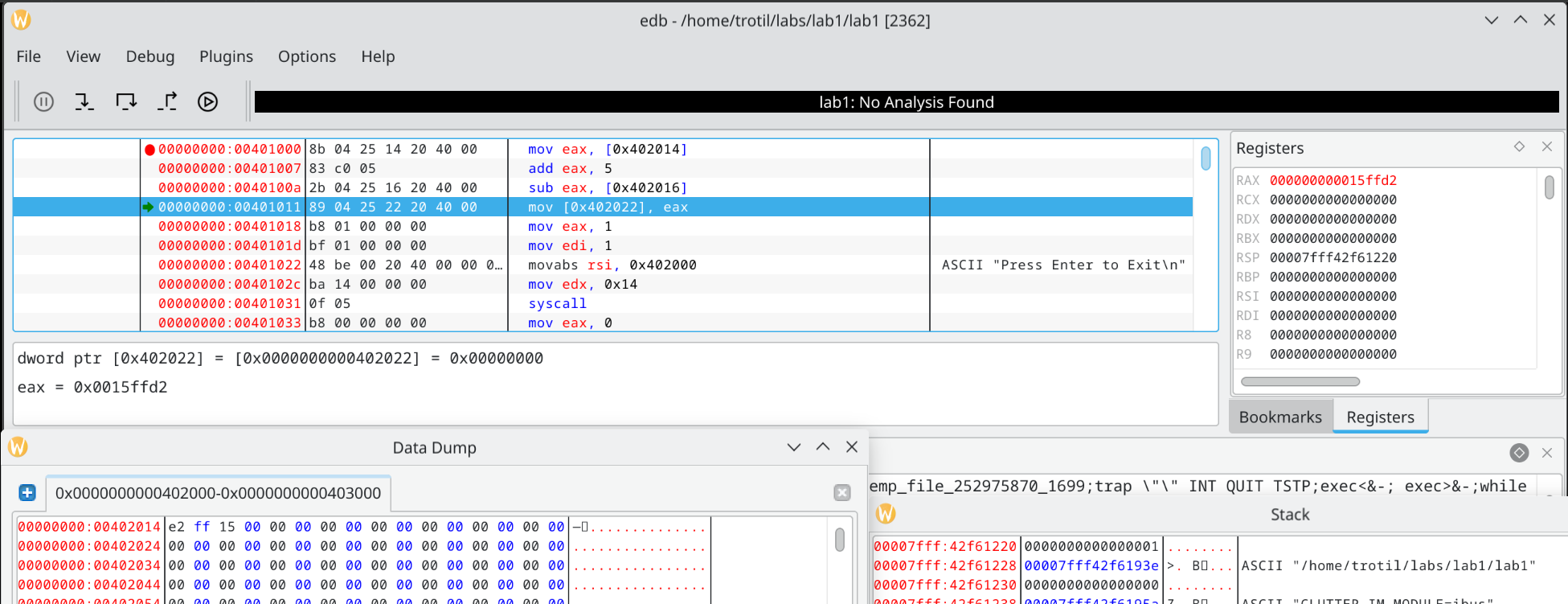




Рисунок 3 – Изменение регистра после выполнения sub eax, [B]

На рисунке 4 показано изменение поля данных после выполнения команды mov [X], EAX, где FF E2 – число -30, 00 15 – число 15, FF D2 – число -46.

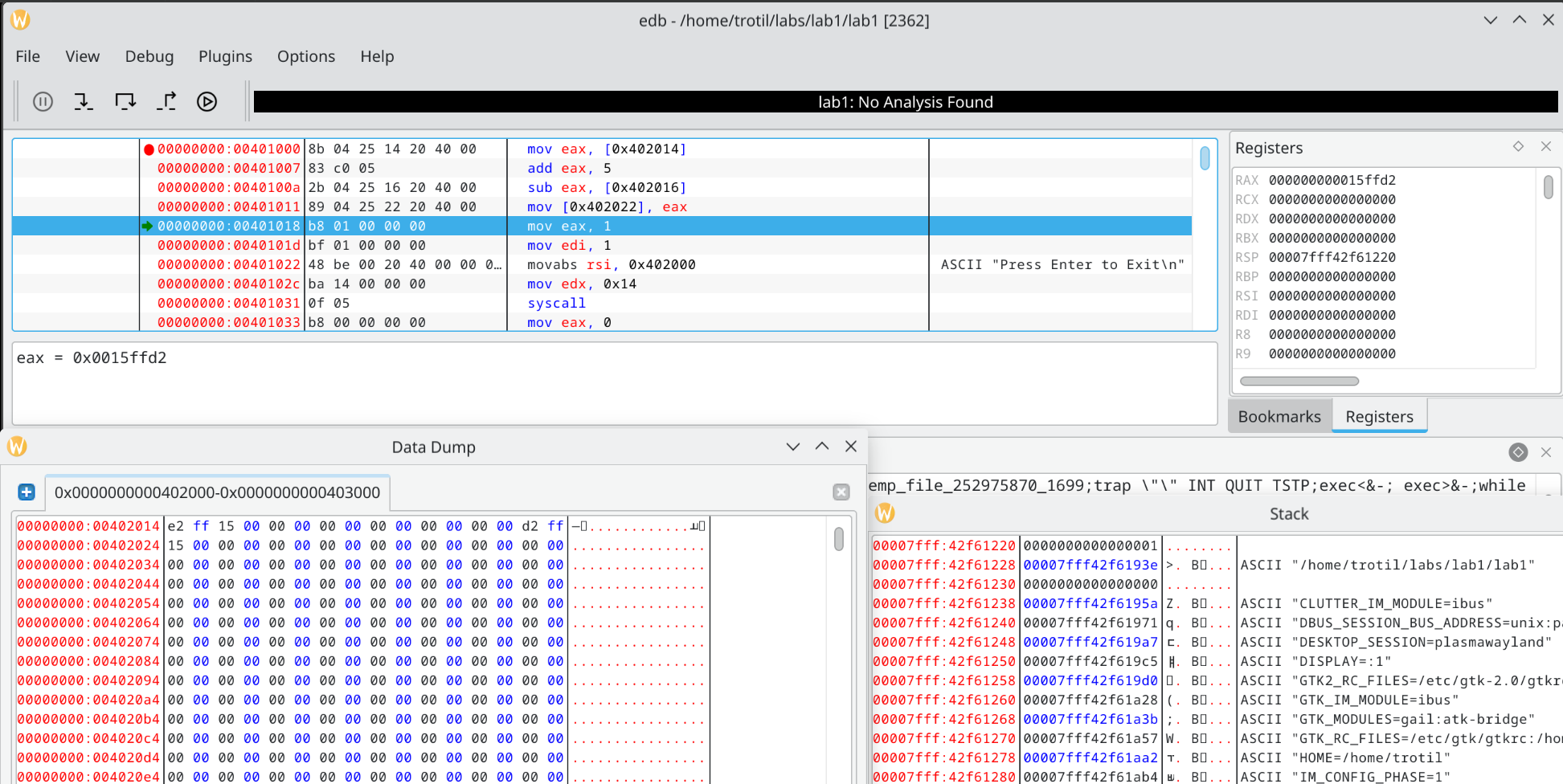




Рисунок 4 – Изменение поля данных после выполнения mov [X], eax

**Задание**

Введите 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.

В листинге 2 показан код программы.

Листинг 2 – Код программы для второго задания

|  |
| --- |
| section .data ;сегмент инициализированных переменных  vall 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  section .bss ;сегмент неинициализированных переменных  alu resw 10  fl resb 5  section .text ;сегмент кода  global \_start  \_start:  mov eax, [vall]  mov eax, [chart]  mov eax, [lue3]  mov eax, [v5]  mov eax, [beta]  mov eax, [sdk]  mov eax, [min]  mov eax, [ar]  mov eax, [valar]  mov eax, [alu]  mov eax, [fl]  syscall |

На рисунке 5 показано представление введенных данных в памяти, где:

* ff – число 255, которое занимает один байт;
* 0100 – число 256;
* ff80 – число -128;;
* 10 – однобайтовая переменная v5;
* 25 – однобайтовое число 100101b;
* 17 – однобайтовая число 23;
* 23 – однобайтовое число 23h;
* 0c – однобайтовое число 0ch;
* 48 65 6c 6c 6f 0a – строка Hello с маркером конца строки 10;
* 8001 – двухбайтовое число -32767;
* 12345678 – число 12345678h;
* 0808080808 – переменная valar;
* затем 20 байтов зарезервировано под переменную alu и 5 байтов под переменную fl.

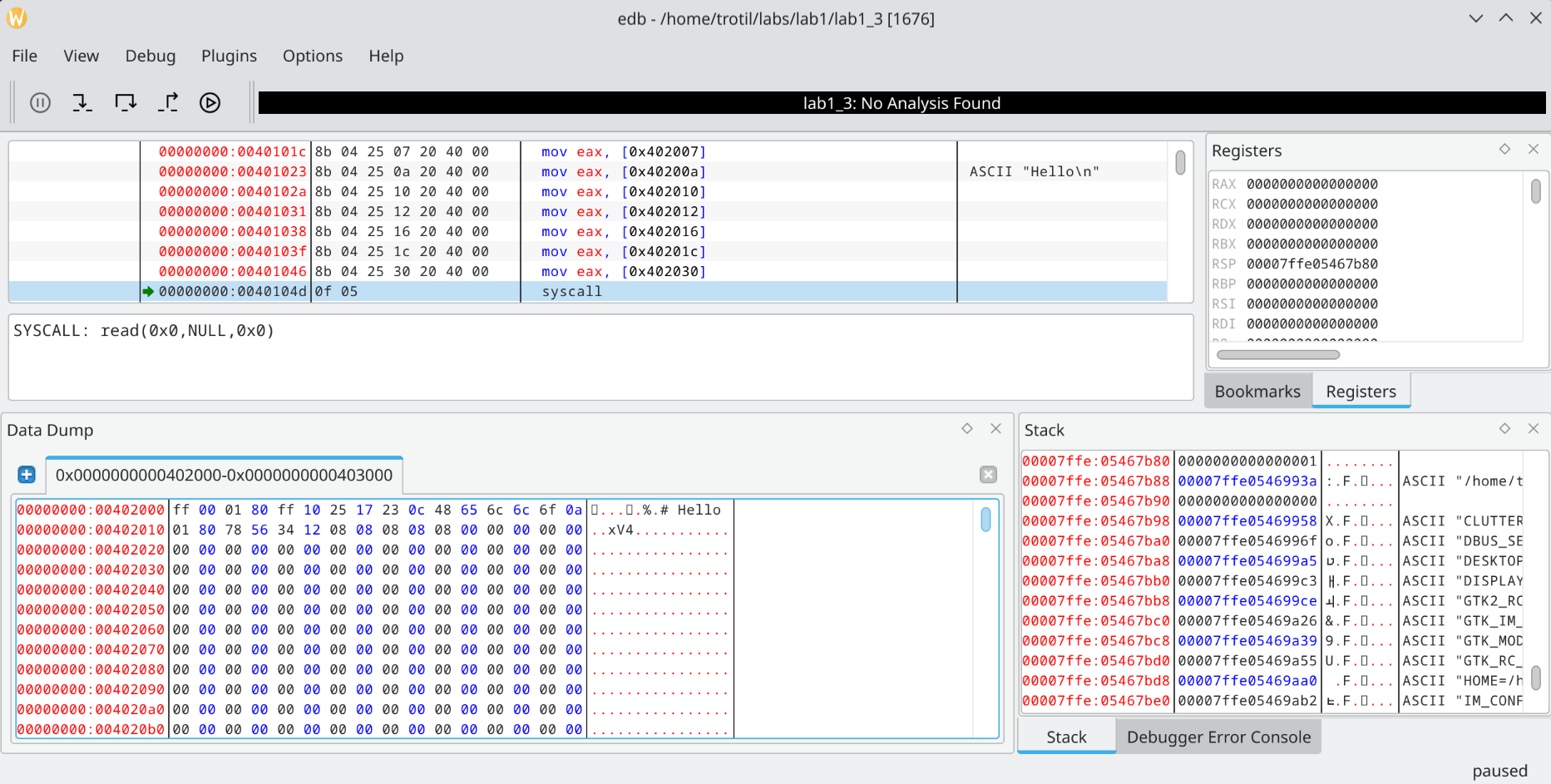




Рисунок 5 – Представление данных в памяти

**Задание:**

Определите в памяти следующие данные:

а) целое число 25 размером 2 байта со знаком;

б) 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.

В листинге 3 показан код программы.

Листинг 3 – Код программы для 3 задания

|  |
| --- |
| section .data ;сегмент инициализированных переменных  right dw 25  slovo dd -35  name db "Irina", 10  imya db "Ирина", 10  section .bss ;сегмент неинициализированных переменных  section .text ;сегмент кода  global \_start  \_start:  mov eax, right;  mov eax, slovo;  mov eax, name;  mov eax, imya;  syscall |

На рисунке 6 показано представление введенных данных в памяти, где:

* 0019 – двухбайтовое число 25 (right)
* ffffffdd – четырёхбайтовое число -35 (slovo)
* 49 72 69 6e 61 0a – моё имя латинскими буквами, каждая из которых занимает в памяти 1 байт, заканчивающееся маркером конца строки 10
* d098 d180 d0b8 d0bd d0b0 0a – моё имя русскими буквами, каждая из которых занимает в памяти 2 байта (из-за кодировки UTF-8), заканчивающееся маркером конца строки 10.

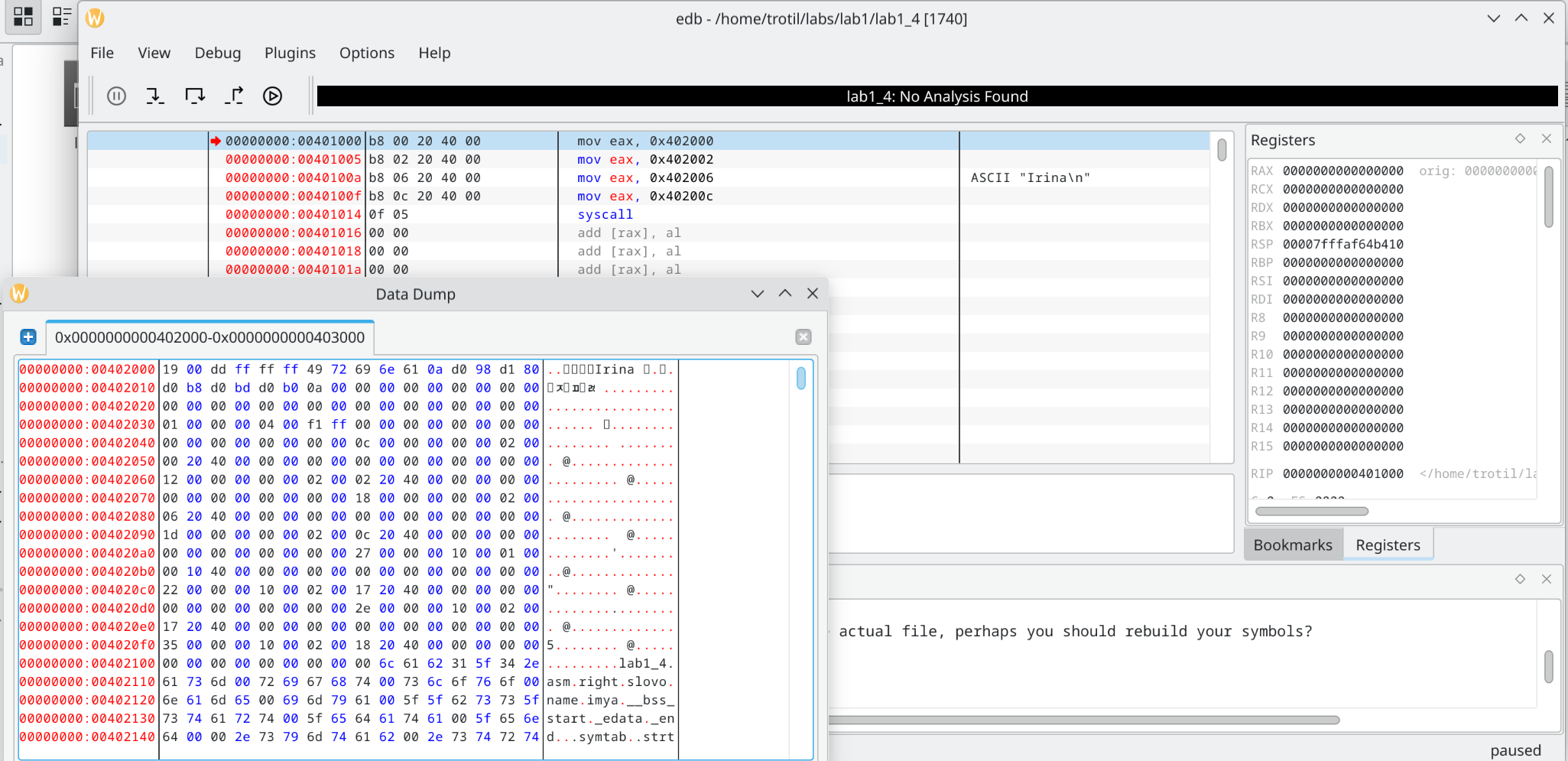




Рисунок 6 – Представление данных в памяти

**Задание**:

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. Зафиксируйте результаты в отчете.

Чтобы в отладчике увидеть 25 00 необходимо ввести 0025h или 37.

В листинге 4 показан код программы первой части.

Листинг 4 – Код программы для первой части четвертого задания

|  |
| --- |
| section .data ;сегмент инициализированных переменных  right dw 37  dw 25h  section .bss ;сегмент неинициализированных переменных  section .text ;сегмент кода  global \_start  \_start:  mov eax, right;  syscall |

На рисунке 7 показано представление введенных данных в памяти.





Рисунок 7 – Представление данных в памяти

Чтобы в отладчике увидеть 00 25 необходимо ввести 9472 или 2500h.

В листинге 5 показан код программы второй части.

Листинг 5 – Код программы второй части четвертого задания.

|  |
| --- |
| section .data ;сегмент инициализированных переменных  right dw 9472  dw 2500h  section .bss ;сегмент неинициализированных переменных  section .text ;сегмент кода  global \_start  \_start:  mov eax, right;  syscall |

На рисунке 8 показано представление введенных данных.

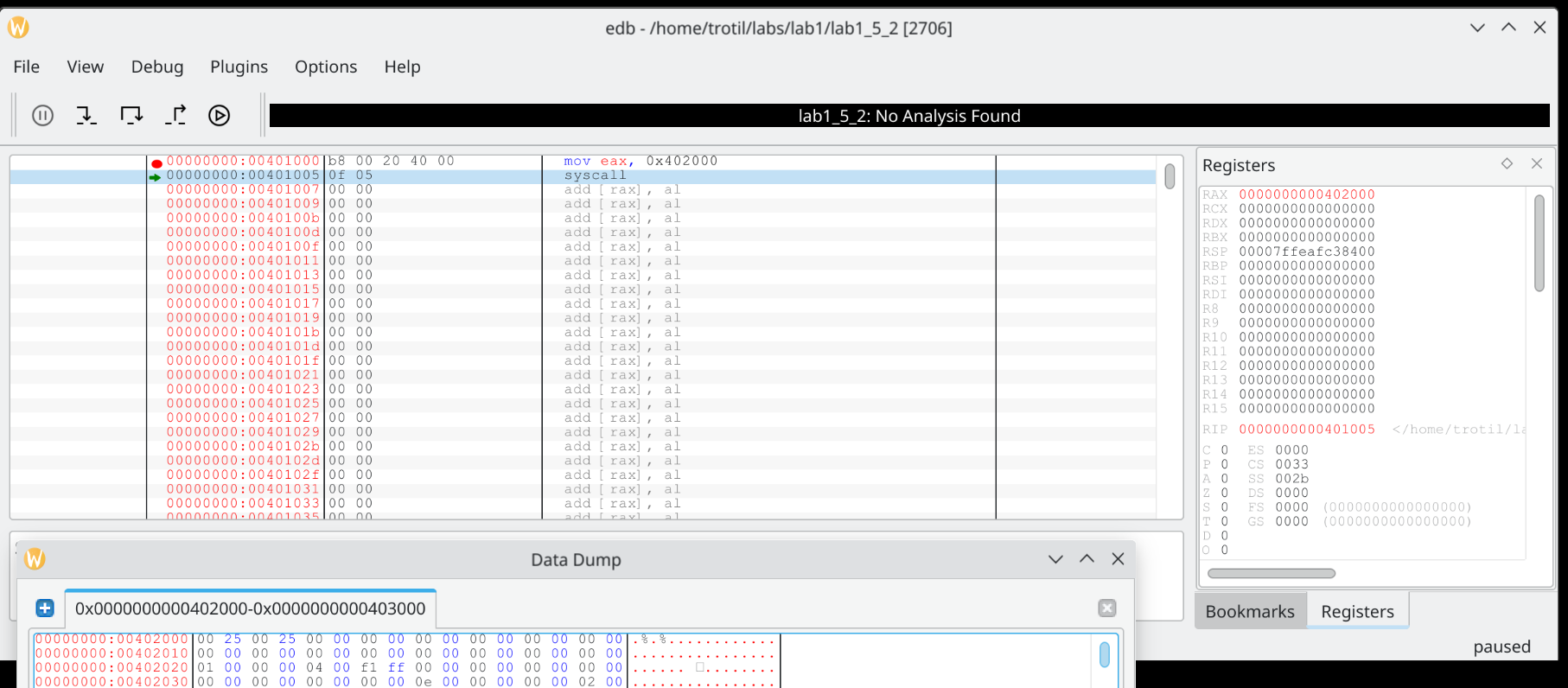
****



Рисунок 8 – Представление данных в памяти

**Задание**:

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

Проанализируйте и прокомментируйте в отчете полученный результат (обратите внимание на флаги).

В листинге 6 показан код программы.

Листинг 6 – Код программы для пятого задания

|  |
| --- |
| section .data ;сегмент инициализироанных переменных  F1 dw 65535  F2 dd 65535  section .bss ;сегмент неинициализированных переменных  section .text ;сегмент кода  global \_start  \_start:  add word[F1], 1  add dword[F2], 1  syscall |

На рисунке 9 показано представление данных в памяти и регистров до выполнения арифметических операций.

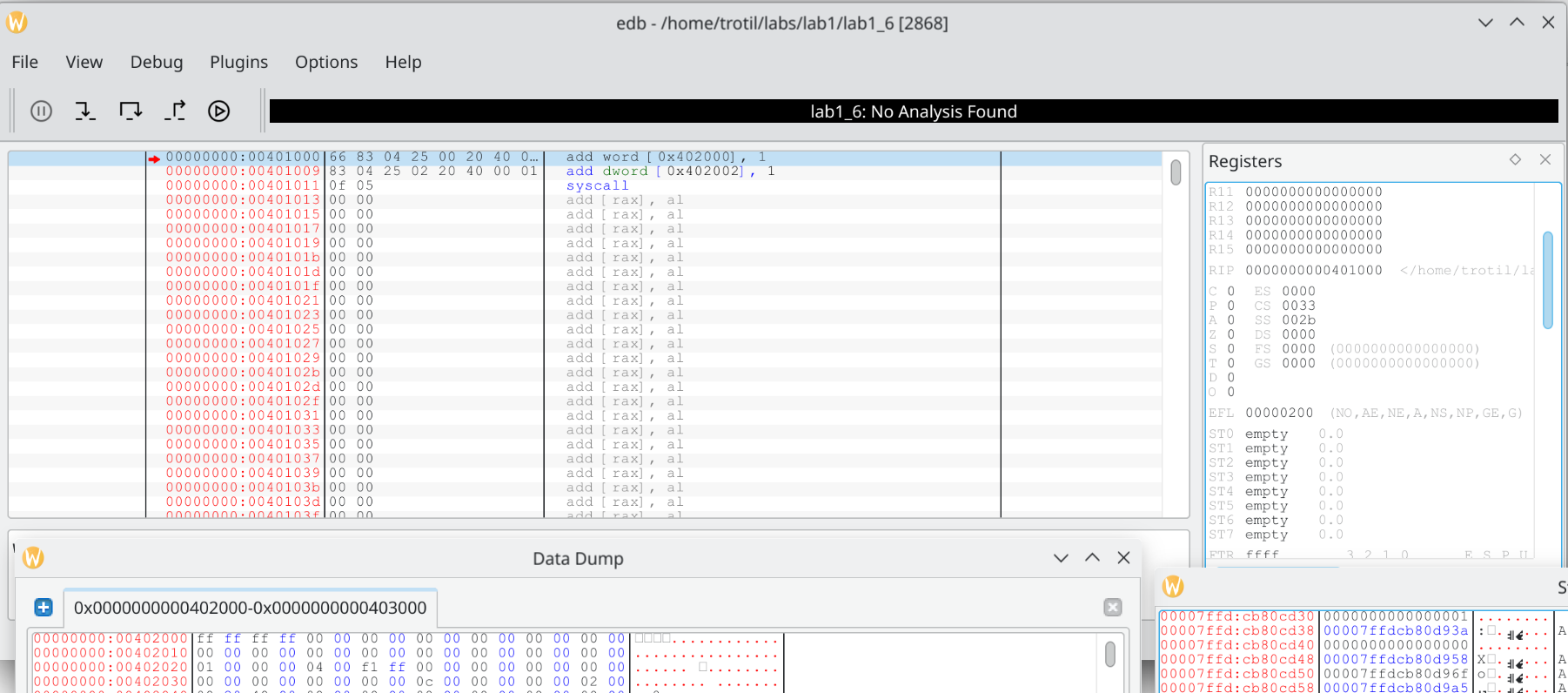
****



Рисунок 9 – Представление данных в памяти и вид регистров

На рисунке 10 показание изменения регистров после выполнения команды add [f1], 1: был установлен Carry Flag (флаг переноса), Auxiliary Carry Flag (флаг вспомогательного переноса), так как произошло переполнение в результате арифметической операции.

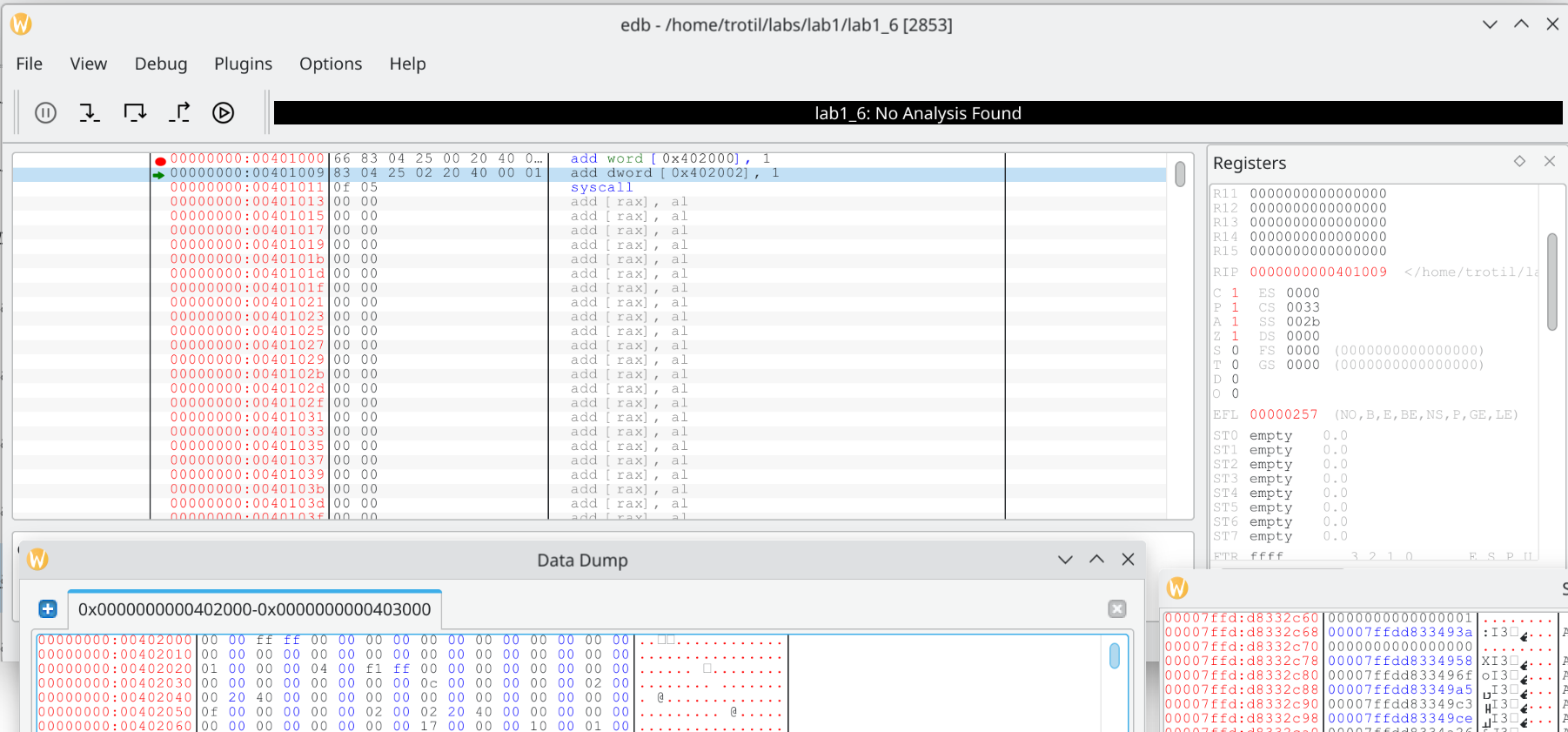
****



Рисунок 10 – Изменение регистров после выполнения add [f1], 1

На рисунке 11 показано изменение регистров после выполнения команды add [f2], 1: число 65535 для типа данных DWORD не является максимальным, поэтому при прибавлении 1 значение F2 увеличивается и устанавливается Auxiliary Carry Flag (флаг вспомогательного переноса).

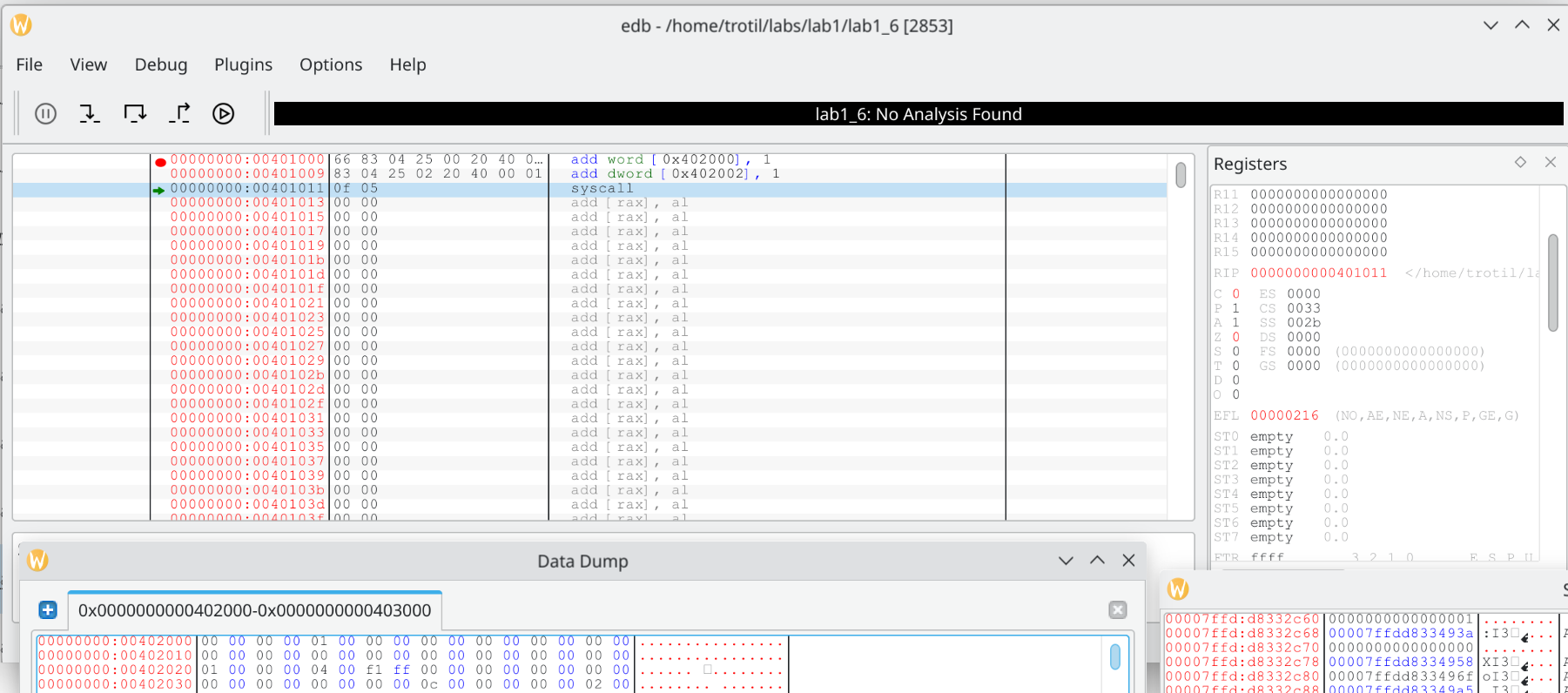
****



Рисунок 11 – Изменение регистров после выполнения add [f2], 1

**Вывод**: мы изучили процессы создания, запуска и отладки программ на ассемблере Nasm, внутреннее представление данных в памяти и изменение регистров, а также написали первые программы на этом языке программирования.

**Контрольные вопросы**:

1. Дайте определение ассемблеру. К какой группе языков он относится?

Ассемблер — язык низкого уровня с командами, обычно соответствующими командам процессора (так называемым машинным командам). Это соответствие позволяет отнести язык к группе машинно-зависимых, к которой относятся также машинные языки.

1. Из каких частей состоит заготовка программы на ассемблере?

И 32-х и 64-х разрядные программы состоят из трех секций (сегментов):

.text – сегмент кода;

.data – сегмент инициализированных данных;

.bss – сегмент неинициализированных данных.

1. Как запустить программу на ассемблере на выполнение? Что происходит с программой на каждом этапе обработки?
2. Создание трансляции Создать/Assemble. Программа преобразуется в её двоичный эквивалентный код.

nasm -f <Формат> <Имя\_исходного\_модуля>

[-o <Имя\_объектного\_модуля>]

[-l <Имя\_файла\_листинга>]

1. Осуществление компоновки Создать/Link. К ранее созданному двоичному коду добавляются коды подпрограмм.

ld [-m elf\_i386] [-o <Имя\_исполняемого\_модуля>]

<Имя\_объектного\_модуля>

1. запуск на выполнение Создать/Run. Запуск программы на выполнение (программа в текущем каталоге) осуществляется по ее имени:

./<Имя\_исполняемого\_модуля>

Программа при запуске загружается в оперативную память и начинает выполняться.

1. Назовите основные режимы работы отладчика. Как осуществить пошаговое выполнение программы и просмотреть результаты выполнения машинных команд.

- F7 — выполнить шаг с заходом в подпрограмму;

- F8 — выполнить шаг без захода в подпрограмму;

- Ctrl + F2 — начать отладку сначала;

- Ctrl + F9 — выполнить подпрограмму до возврата из нее.

1. В каком виде отладчик показывает положительные и отрицательные целые числа? Как будут представлены в памяти числа: A dw 5,-5 ? Как те же числа будут выглядеть после загрузки в регистр AX?

Для положительных чисел EDB отображает их в шестнадцатеричном виде без каких-либо альтернативных обозначений.

Отрицательные числа в EDB отображаются в дополнительном коде.

В памяти число 5 будет представлено в виде 0005, а -5 – fffb (0000 0000 0000 0101 – 1111 1111 1111 1010 – 1111 1111 1111 1011 - fffb).

После загрузки 5 в AX: 0000 0005.

После загрузки -5 в AX: 0000 fffb.

1. Каким образом в ассемблере программируются выражения? Составьте фрагмент программы для вычисления С=A+B, где A, В и С – целые числа формата BYTE.

Для программирования выражений в ассемблере нужно указывать операнды и операторы, которые определяют операцию, которую необходимо выполнить, например, add, sub, div, imul и т.п.

В листинге 7 показан код программы.

Листинг 7 – Код программы для шестого контрольного вопроса

|  |
| --- |
| section .data ;сегмент инициализированных переменных  A dw 5  B dw 10  section .bss ;сегмент неинициализированных переменных  C resb 1  section .text ;сегмент кода  global \_start  \_start:  Mov ax, [A]  add ax, [B]  mov [C], ax  syscall;вызов системной функции |

На рисунке 12 показано содержание регистра AX и представление данных в памяти.

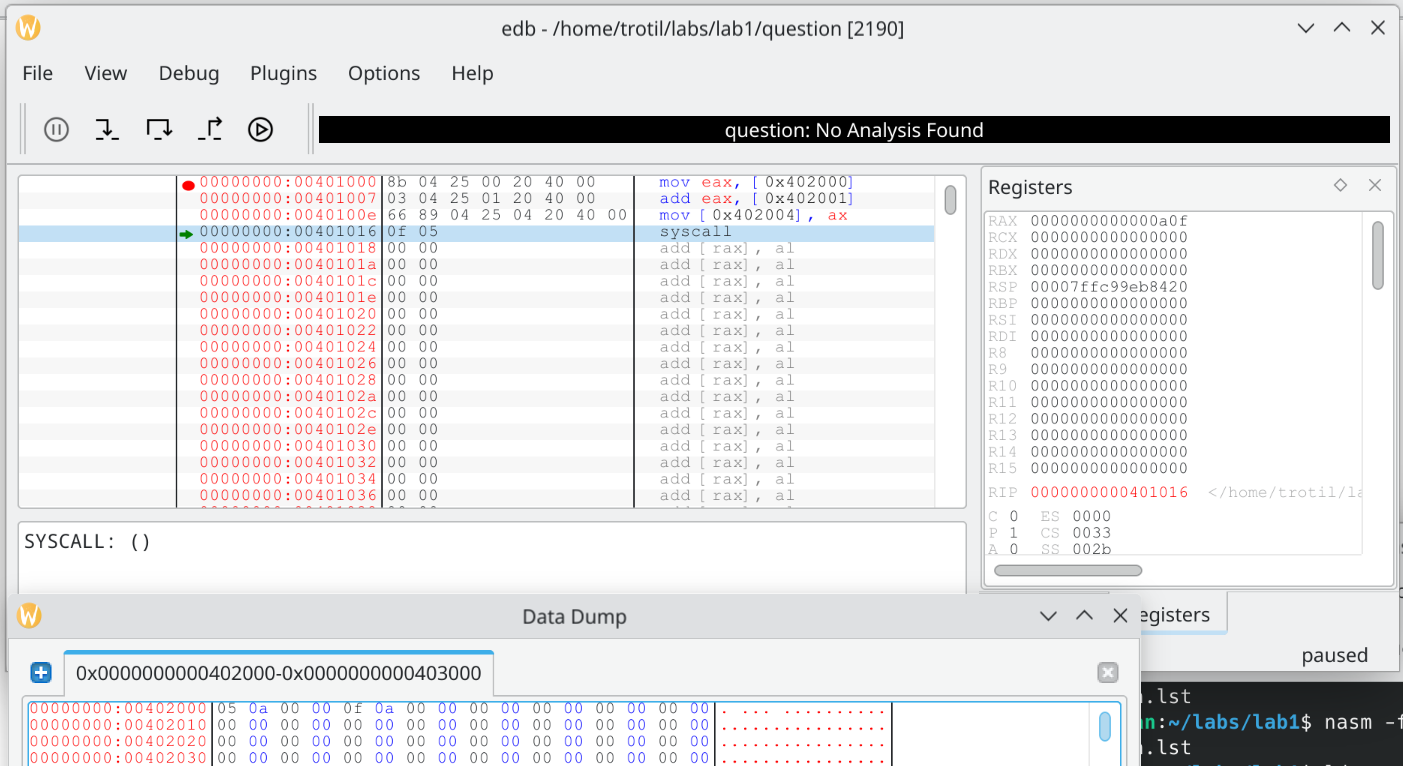




Рисунок 12 – Выполнение программы, содержание регистра AX, представление данных в памяти