|  |  |
| --- | --- |
| 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 № 2**

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

**Название лабораторной работы: Программирование целочисленных вычислений**

Вариант 2.21

Студент гр. ИУ6-42Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.С. Твердюк**

(Подпись, дата) (И.О. Фамилия)

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

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

***Цель работы:*** изучение форматов машинных команд, команд целочисленной арифметики ассемблера и программирование целочисленных вычислений.

***Ход работы:***

***Задание.*** Разработать программу на языке ассемблера, вычисляющую выражение:

**Программа LR2.asm**

section .data

prsM db "Enter m:", 10

lenPrsM equ $-prsM

prsA db "Enter a:", 10

lenPrsA equ $-prsA

ExitMsg db "Result:", 10

lenExit equ $-ExitMsg

section .bss

InBuf resb 10

lenIn equ $-InBuf

OutBuf resb 10

a resw 1

m resw 1

b resw 1

x resw 1

section .text

global \_start

\_start:

; Enter m

mov rax, 1 ; системная функция 1 (write)

mov rdi, 1 ; дескриптор файла stdout=1

mov rsi, prsM ; адрес выводимой строки

mov rdx, lenPrsM ; длина выводимой строки

syscall ; вызов системной функции

mov rax, 0 ; системная функция 0 (read)

mov rdi, 0 ; дескриптор файла stdin=0

mov rsi, InBuf ; адрес вводимой строки

mov rdx, lenIn ; длина вводимой строки

syscall ; вызов системной функции

mov rsi, InBuf ; адрес введённой строки

call StrToInt64

cmp ebx, 0 ; проверка кода ошибки

jne StrToInt64.Error ; при преобразовании обнаружена ошибка

mov [m], ax ; запись числа в память

; Enter a

mov rax, 1 ; системная функция 1 (write)

mov rdi, 1 ; дескриптор файла stdout=1

mov rsi, prsA ; адрес выводимой строки

mov rdx, lenPrsA ; длина выводимой строки

syscall ; вызов системной функции

mov rax, 0 ; системная функция 0 (read)

mov rdi, 0 ; дескриптор файла stdin=0

mov rsi, InBuf ; адрес вводимой строки

mov rdx, lenIn ; длина вводимой строки

syscall ; вызов системной функции

mov rsi, InBuf ; адрес введённой строки

call StrToInt64

cmp ebx, 0 ; проверка кода ошибки

mov [a], ax ; запись числа в память

mov rax, 0

mov rcx, 0

mov rdx, 0

mov dx, [m] ; загрузка числа в регистр

sub dx, 5 ; m - 5

mov ax, [m] ; загрузка числа в регистр

add ax, 2 ; m + 2

imul dx, ax ; (m - 5) \* (m + 2)

add dx, [m] ; (m - 5) \* (m + 2) + m

mov cx, dx

mov rax, 0

mov rdx, 0

mov ax, [a] ; загрузка числа в регистр

mov bx, 2 ; загрузка числа в регистр

cwd

idiv bx ; a/2

add cx, ax ; (m - 5) \* (m + 2) + m + a/2

mov [x], cx ; загрузка результата в память

mov rax, 1

mov rdi, 1

mov rsi, ExitMsg

mov rdx, lenExit

syscall

mov rsi, OutBuf

mov ax, [x]

cwde

call IntToStr64

mov rdx, rax

mov rax, 1

mov rdi, 1

mov rsi, OutBuf

syscall

; exit

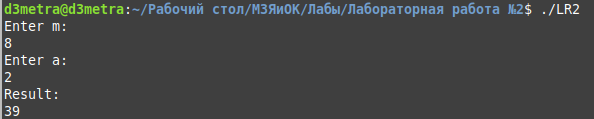
mov rax, 60 ; системная функция 60 (exit)

xor rdi, rdi ; return code 0

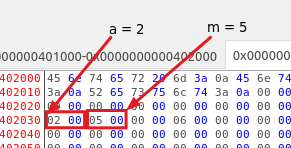
syscall ; вызов системной функции

%include "../lib64.asm"

Запустим программу. Результат выполнения программы представлен на рисунке 1.

Рисунок 1 - Выполнение программы

Проведем поэтапную отладку программы в edb в таблице 1. Используем в качестве тестовых данных: m = 5, a = 2. Их представление отображено на рисунке 2.

Рисунок 2 - Представление переменных a, m после ввода

*Таблица 1 — Трассировка программы*

|  |  |  |  |
| --- | --- | --- | --- |
| **Команда** | **Математическое представление** | **Пояснение** | **Содержание регистров** |
| mov dx, [m] | - | Загрузка числа m в регистр dx |  |
| sub dx, 5 | m - 5 = 5 - 5 = 0 | Нахождение разности числа, находящегося в регистре dx, и 5 с последующим сохранением результата в регистр dx |  |
| mov ax, [m] | - | Загрузка числа m в регистр ax |  |
| add ax, 2 | m + 2 = 5 + 2 = 7 | Нахождение суммы числа, находящегося в регистре ax, и 2 с последующим сохранением результата в регистр ax |  |
| imul dx, ax | (m — 5) \* (m + 2) =  = 0 \* 7 = 0 | Нахождение произведения чисел, находящихся в регистрах dx и ax, с последующим сохранением результата в регистр dx |  |
| add dx, [m] | (m — 5) \* (m + 2) + m = 0 + 5 = 5 | Нахождение суммы числа, находящегося в регистре dx, и m с последующим сохранением результата в регистр dx |  |
| mov cx, dx | - | Перемещение числа из регистра dx в регистр cx |  |
| mov rax, 0  mov rdx, 0 | - | Очистка регистров rax и rdx |  |
| mov ax, [a] | - | Загрузка числа a в регистр ax |  |
| mov bx, 2 | - | Загрузка числа 2 в регистр bx |  |
| idiv bx | a / 2 = 2 / 2 = 1 | Нахождение частного числа, находящегося в регистре ax, и числа, находящегося в регистре bl, с последующим сохранением результата в регистр ax |  |
| add cx, ax | (m — 5) \* (m + 2) + m + a / 2 = 5 + 1 = 6 | Нахождение суммы чисел, находящихся в регистрах dx и ax, с последующим сохранением результата в регистр dx |  |

Результаты переменных после вычислений можно наблюдать на рисунке 2.

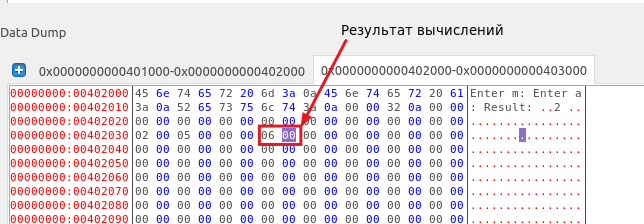


Рисунок 2 - Представление результата вычислений в памяти

Проведём тестирование программы. Результаты представлены в таблице 2.

*Таблица 2 — Тестирование программы*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| m = 2 a = 4 | -8 | -8 |
| m = 10 a = 10 | 75 | 75 |
| m = -2 a = 2 | -1 | -1 |
| m = -7 a = -6 | 50 | 50 |
| m = 0 a = 0 | -10 | -10 |

Расшифруем команды. Расшифровка произведена в таблицах 3-6.

*Таблица 3 — Расшифровка команды mov rdx, rax*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Команда | | | | | mov rdx. rax | | |
| Вид команды в отладчике | | | | | mov rdx. rax | | |
| Внутреннее представление команды | | | | | 89 С2 | | |
| Двоичный эквивалент команды | | | | | 1000 1001 1100 0010 | | |
| Расшифровка | | | | | | | |
| Префикс | mov | D | W | mod | | rax | rdx |
| - | 100010 | 0 | 0 | 11 | | 000 | 010 |
| - | Пересылка | Из регистра | 8 байт | Операнды - регистры | | Источник | Приёмник |

*Таблица 4 — Расшифровка команды mov dx, [m]*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда | | | | | mov dx, [m] | | | | | |
| Вид команды в отладчике | | | | | mov dx, [0x402032] | | | | | |
| Внутреннее представление команды | | | | | 66 8B 14 25 32 20 40 00 | | | | | |
| Двоичный эквивалент команды | | | | | 1000 1011 0001 0100 0010 0101 | | | | | |
| Расшифровка | | | | | | | | | | |
| Преф. | mov | D | W | mod | dx | sib | ss | index | base | Смещ. |
| 66h | 100010 | 1 | 1 | 00 | 010 | 100 | 00 | 100 | 101 | 32 20 40 00 |
| 16 разр. операнды | Пересылка | В регистр | 4 байт | Смещение отсут. | Приёмник | Указ на sib | Масштаб | Индекс | База |  |

*Таблица 5 — Расшифровка команды mov [x], dx*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда | | | | | mov dx, [m] | | | | | |
| Вид команды в отладчике | | | | | mov [0x402036], dx | | | | | |
| Внутреннее представление команды | | | | | 66 89 14 25 36 20 40 00 | | | | | |
| Двоичный эквивалент команды | | | | | 1000 1001 0001 0100 0010 0101 | | | | | |
| Расшифровка | | | | | | | | | | |
| Преф. | mov | D | W | mod | dx | sib | ss | index | base | Смещ. |
| 66h | 100010 | 0 | 1 | 00 | 010 | 100 | 00 | 100 | 101 | 32 20 40 00 |
| 16 разр. операнды | Пересылка | Из регистра | 4 байт | Смещение отсут. | Иточник | Указ на sib | Масштаб | Индекс | База | 36 20 40 00 |

**Контрольные вопросы**

1. Что такое машинная команда? Какие форматы имеют машинные команды процессора IA32? Чем различаются эти форматы?

Машинная команда – это код, дающий указания процессору какие действия нужно выполнять. Размер машинной команды различается для каждой отдельной команды. Также команды могут иметь различные префиксы, например 66h и 67h и др.

2. Назовите мнемоники основных команд целочисленной арифметики. Какие форматы для них можно использовать?

- ADD – Сложение двух операндов и помещение результата по адресу первого. Форматы: REG, MEM;MEM, REG;REG, REG;MEM, IMM;REG, IMM. 12

- SUB – Вычитание двух операндов и помещение результата по адресу первого. Форматы: REG, MEM;MEM, REG;REG, REG;MEM, IMM;REG, IMM.

- DIV – Разделить то, что находится в регистре DX:AX (или AX, EDX:EAX в зависимости от разрядности операции) на операнд. Результат помещается в регистр AX (AL,EAX), остаток в регистр DX(AH, EDX). Форматы: REG;MEM.

- MUL – Умножить то что находится в регистре AX (или AL, EAX в зависимости от разрядности операции) на операнд. Результат помещается в регистр DX:AX (AX,EDX:EAX). Форматы: REG;MEM.

- INC – Увеличить операнд на 1. Форматы: REG;MEM.

- DEC – Уменьшить операнд на 1. Форматы: REG;MEM.

3. Сформулируйте основные правила построения линейной программы вычисления заданного выражения.

Для вычисления заданного выражения на ассемблере необходимо выбрать регистр-аккумулятор и записать в него результат вычисления одной из частей выражения. Затем последовательно выполнять операции и записывать получившиеся значения в аккумулятор.

4. Почему ввод-вывод на языке ассемблера не программируют с использованием соответствующих машинных команд? Какая библиотека используется для организации ввода-вывода в данной лабораторной?

На языке ассемблера невозможно запрограммировать вводвывод, так как для начала нужно преобразовать все данные в символьный вид. Для этого используется библиотека lib64.asm.

5. Расскажите, какие операции используют при организации ввода вывода.

Для вывода:

mov rax, 1 ; системная функция 1 (write)

mov rdi, 1 ; дескриптор файла stdout=1

mov rsi, prsM ; адрес выводимой строки

mov rdx, lenPrsM ; длина выводимой строки

syscall ; вызов системной функции

Для ввода:

mov rax, 0 ; системная функция 0 (read)

mov rdi, 0 ; дескриптор файла stdin=0

mov rsi, InBuf ; адрес вводимой строки

mov rdx, lenIn ; длина вводимой строки

syscall ; вызов системной функции

***Вывод:*** в процессе выполнения лабораторной работы изучены форматы машинных команд, команд целочисленной арифметики ассемблера и программирование целочисленных вычислений. Разработанная программа на заданных исходных данных работает корректно.