|  |  |
| --- | --- |
| 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

Студент гр. ИУ6 - 42Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**Г. Д. Нефедов

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2024

**Цель работы:** изучение форматов машинных команд, команд

целочисленной арифметики ассемблера и программирование целочисленных

вычислений.

**Задание**

Вычислить целочисленное выражение:

**Код программы:**

Разберем код программы по секциям:

Секция данных:

section .data

ib db "Enter B: ", 0

ic db "Enter C: ", 0

id db "Enter D: ", 0

ia db "Your result: ", 0

В секции данных (.data) объявляем строки и инициализируем их значениями. Каждая строка заканчивается нулевым байтом (0), что означает конец строки в ассемблере.

Секция .bss:

section .bss

msgB resb 10

lenMsgB equ $-msgB

msgC resb 10

lenMsgC equ $-msgC

msgD resb 10

lenMsgD equ $-msgD

msgA resb 13

lenMsgA equ $-msgA

inbuf resb 10

lenIn equ $-inbuf

outbuf resb 10

lenout equ $-outbuf

A resw 1

B resd 1

C resd 1

D resd 1

В секции BSS мы зарезервировали память для переменных, которые будут использоваться в программе, но им еще не присвоены начальные значения. В строке «msgB resb 10», резервируется 10 байт для вывода сообщения ib, аналогично и для сообщений ic, id, ia. В строке «lenMsgB equ $-msgB» определяется длина сообщения lenMsgB как разница между текущим адресом $ и адресом начала строки msgB, аналогично происходит вычисление длины для вывода сообщений ic, id, ia. В данных строках «inbuf resb 10», «lenIn equ $-inbuf» мы резервируется память для вводимых значений и считаем длину вводимого сообщения, аналогично и для outbuf. Далее резервируем память для переменных A, B, C и D.

Секция .text

Секция .text разбита для наглядности на несколько меток, рассмотрим код каждой метки.

Метка InputB:

InputB:

;; вывовдим сообщение ib

mov eax, 4

mov ebx, 1

mov ecx, ib

mov edx, lenMsgB

int 0x80

;; считываем введеное число

mov eax, 3

mov ebx, 0

mov ecx, inbuf

mov edx, 10

int 0x80

;; передаем параметры в функцию

mov esi, inbuf

call StrToInt

cmp EBX, 0

mov [B], ax

jmp InputC

Рассмотрим каждую строку кода по отдельности:

1. «mov eax, 4» - устанавливаем в регистр eax значение 4, что представляет собой системный вызов для вывода строки.

2. «mov ebx, 1» - устанавливаем в регистр ebx значение 1, что указывает на стандартный вывод (stdout).

3. «mov ecx, ib» - устанавливаем в регистр ecx адрес строки ib.

4. «mov edx, lenMsgB» - устанавливаем в регистр edx длину строки сообщения B.

5. «int 0x80» - системный вызов, который приводит к выводу строки на экран.

6. «mov eax, 3» - системный вызов для ввода.

7. «mov ebx, 0» - устанавливаем в регистр ebx значение 0, что указывает на стандартный ввод (stdin).

8. «mov ecx, inbuf» - устанавливаем в регистр ecx адрес буфера ввода.

9. «mov edx, 10» - устанавливаем в регистр edx максимальную длину ввода.

10. «int 0x80» - системный вызов, который считывает введенное значение и сохраняет его в буфер ввода.

11. «mov esi, inbuf» - устанавливаем в регистр esi адрес буфера ввода.

12. «call StrToInt» - вызов подпрограммы для преобразования строки в число.

13. «cmp EBX, 0» - сравниваем значение в регистре ebx с 0, чтобы проверить, произошла ли ошибка преобразования.

14. «mov [B], ax» - сохраняем результат преобразования в переменную B.

15. «jmp InputC» - переход к следующему блоку кода.

Аналогичный код для меток InputС, InputD.

Метка Calc:

Calc:

;;записываем значения переменных в регистры

mov bx, [B]

mov ax, [C]

mov dx, [D]

xor ecx, ecx

mov cx, ax

sub cx, 5

imul cx, dx

add cx, bx

mov ax, cx

imul bx, bx

add bx, 1

xor edx, edx

div bx

shl edx, 16

or eax, edx

mov [A], ax

jmp Output

Рассмотрим каждую строку кода по отдельности:

1. «mov bx, [B]» - устанавливает значение переменной B в регистр bx.
2. «mov ax, [C]» - устанавливает значение переменной C в регистр ax.
3. «mov dx, [D]»-устанавливает значение переменной D в регистр dx.
4. «xor ecx, ecx» - обнуляет регистр ecx.
5. «mov cx, ax» - устанавливает значение регистра ax в регистр cx.
6. «sub cx, 5» - вычитает 5 из значения в регистре cx (С- 5).
7. imul cx, dx: Умножает значение регистра cx на значение в регистре dx.
8. add cx, bx: Прибавляет значение регистра bx к значению в регистре cx.
9. mov ax, cx: Загружает значение регистра cx в регистр ax.
10. imul bx, bx: Умножает значение регистра bx само на себя.
11. add bx, 1: Прибавляет 1 к значению в регистре bx.
12. xor edx, edx: Обнуляет регистр edx.
13. div bx: Делит значение в регистре dx:ax на значение в регистре bx. Результат сохраняется в регистре ax, а остаток - в регистре dx.
14. shl edx, 16: Сдвигает биты в регистре edx на 16 позиций влево.
15. or eax, edx: Применяет логическую операцию ИЛИ между значениями в регистрах eax и edx.
16. mov [A], ax: Загружает значение регистра ax в переменную A.
17. jmp Output: Безусловный переход к метке Output, вероятно, для вывода результата.

Output:

;;конвертирум из A в строку

mov esi, outbuf

mov ax, [A]

call IntToStr

mov eax, 4

mov ebx, 1

mov ecx, ia

mov edx, lenMsgA

int 0x80

;; выводим результат

mov eax, 4

mov ebx, 1

mov ecx, outbuf

mov edx, lenout

int 0x80

jmp Exit

mov esi, outbuf: Устанавливает указатель строки esi на начало буфера outbuf.

1. mov ax, [A]: Устанавливает значение переменной A в регистр ax.
2. call IntToStr: Вызывает процедуру IntToStr для конвертации значения регистра ax в строку и сохранения ее в буфере outbuf.
3. mov eax, 4: Устанавливает номер системного вызова для вывода строки в регистр eax.
4. mov ebx, 1: Устанавливает дескриптор файла (stdout) в регистр ebx.
5. mov ecx, ia: Устанавливает указатель на строку ia в регистр ecx.
6. mov edx, lenMsgA: Устанавливает длину строки lenMsgA в регистр edx.
7. int 0x80: Выполняет системный вызов для вывода строки на экран.
8. mov eax, 4: Устанавливает номер системного вызова для вывода строки в регистр eax (повторно).
9. mov ebx, 1: Устанавливает дескриптор файла (stdout) в регистр ebx (повторно).
10. mov ecx, outbuf: Устанавливает указатель на строку outbuf в регистр ecx.
11. mov edx, lenout: Устанавливает длину строки lenout в регистр edx.
12. int 0x80: Выполняет системный вызов для вывода результата строки на экран.
13. jmp Exit: Безусловный переход к метке Exit, вероятно, для завершения программы.

Exit:

mov eax, 1

xor ebx, ebx

int 0x80

Exit - это метка, обозначающая конец программы:

1. mov eax, 1: Загружает номер системного вызова для завершения программы в регистр eax.
2. xor ebx, ebx: Обнуляет регистр ebx (обычно используется как код возврата).
3. int 0x80: Выполняет системный вызов для завершения программы.