МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

#### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по дисциплине

«Цифровые устройства и микропроцессоры»

Лабораторная работа №1

«СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА X86»

Вариант №15

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИНБс-3301-01-00 | \_\_\_\_\_\_\_\_\_\_\_\_ | С.А. Симахин |
| Проверил: доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | М. А. Земцов |

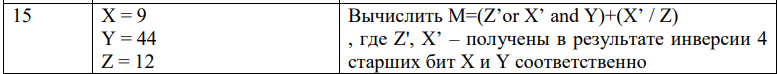
#### Киров 2025

**Цель работы:** изучение системы команд и способов адресации микропроцессоров с архитектурой x86.

**Ход работы:**

**1**.Исходные данные представлены в таблице 1 и пояснении ниже.

Таблица 1 – Исходные данные



Из формулировки задания сказали cделать: Z 🡪 Z’

X 🡪 X’

**2.** Вычислить M=(Z’or X’ and Y)+(X’ / Z), где Z', X’ – получены в результате инверсии 4 старших бит Z и X соответственно.

Х = 9 = 9h 🡪 X’ = 11111001 = F9h

Y = 44 = 2Ch

Z = 12 = Ch 🡪 Z’ = 11111100 = FCh



Рисунок 1 - очистка памяти в регистре EAX (17 строка)



Рисунок 2 - очистка памяти в регистре EBX (18 строка)



Рисунок 3 - очистка памяти в регистре EDX (19 строка)



Рисунок 4 – ввод в регистр EBX значения z (20 строка)



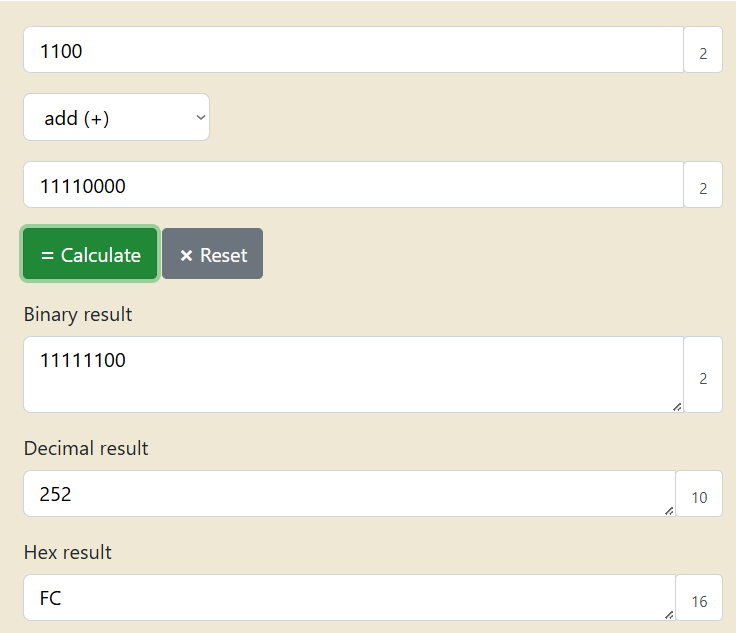


Рисунок 5 – ввод значения Z’ в регистр EBX (22 строка)



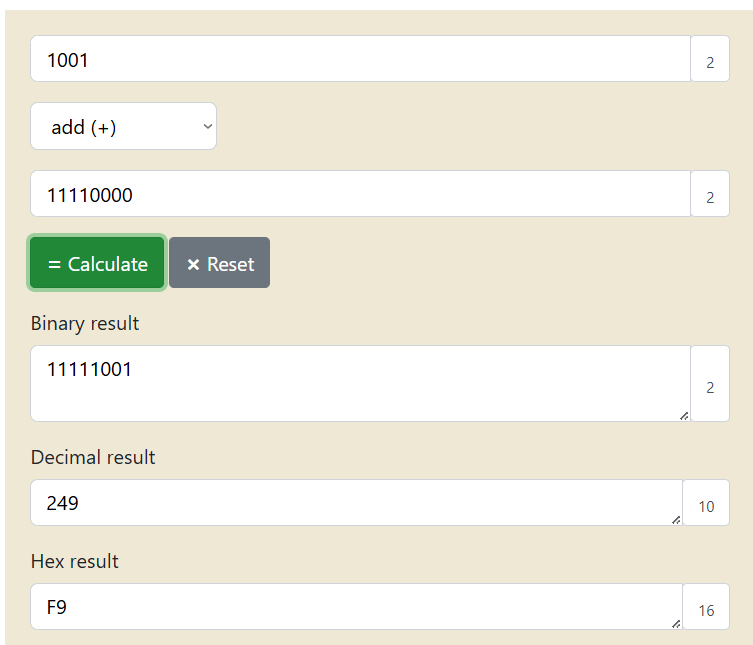


Рисунок 6 – ввод значения x в регистр EAX (25 строка)



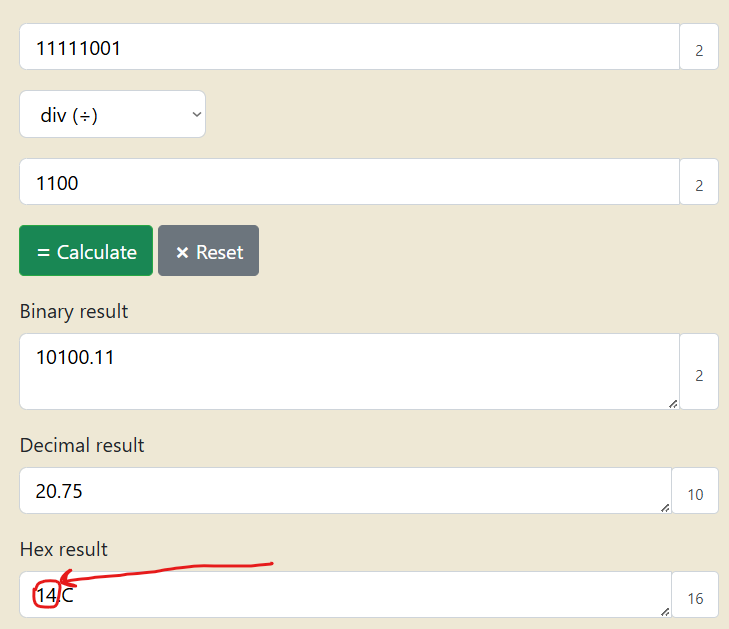


Рисунок 7 – деление на переменную z значения, находящегося в регистре EAX (26 строка)



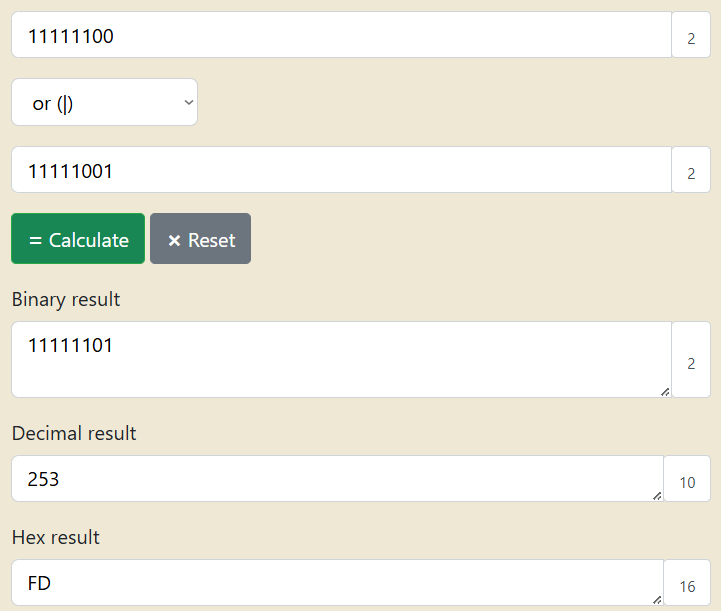


Рисунок 8 – операция or полученного ранее значения в регистре EBX и переменной x



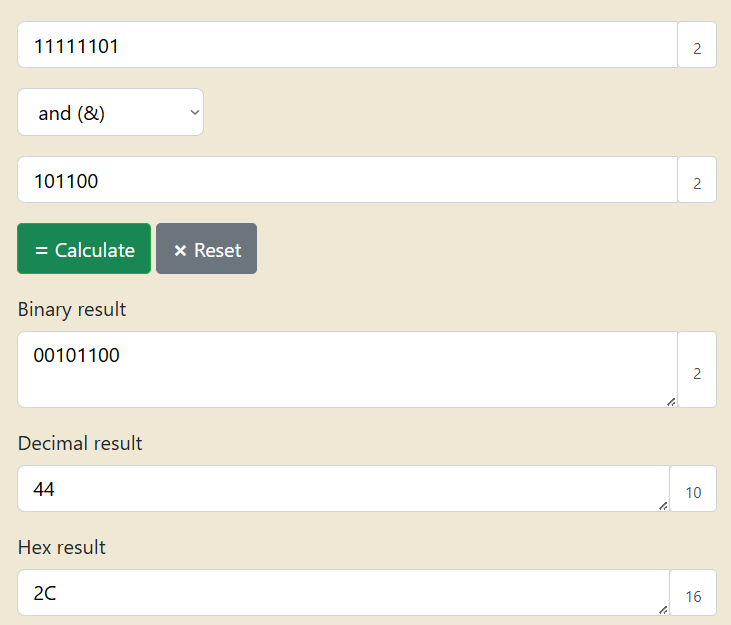


Рисунок 9 – операция and полученного ранее значения в регистре EBX и переменной y



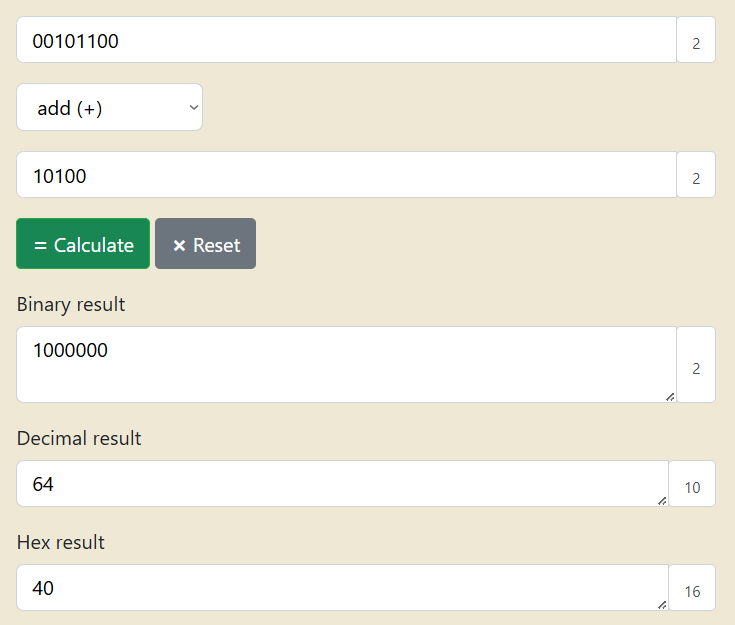


Рисунок 10 – сумма значений, находящихся в регистрах EAX и EBX, конечный результат

Исходный код:

.stack 100h

.data

x dw 9 ; задание переменных

y dw 44

z dw 12

.code

ExitProcess PROTO STDCALL :DWORD

Start:

xor eax,eax

xor ebx,ebx

xor edx,edx ; очищение памяти для исключения перегрузки

mov bx, z ; ввод в регистр BX значение числа z == Ch == 12 (10)

xor x,11110000b ;X` == 11111001 == F9

xor bx,11110000b ;Z` == 11111100 == FC

; скобка (X' / z)

mov ax, x ; ввод в регистр AX значение числа х

div z ; деление на переменную z == 14h == 20 (10)

; скобка (Z' or X' and Y)

or bx, x ; операция z or x, так как в регистре BX уже находится значение z

and bx, y ; операция полученного значения в регистре BX and y

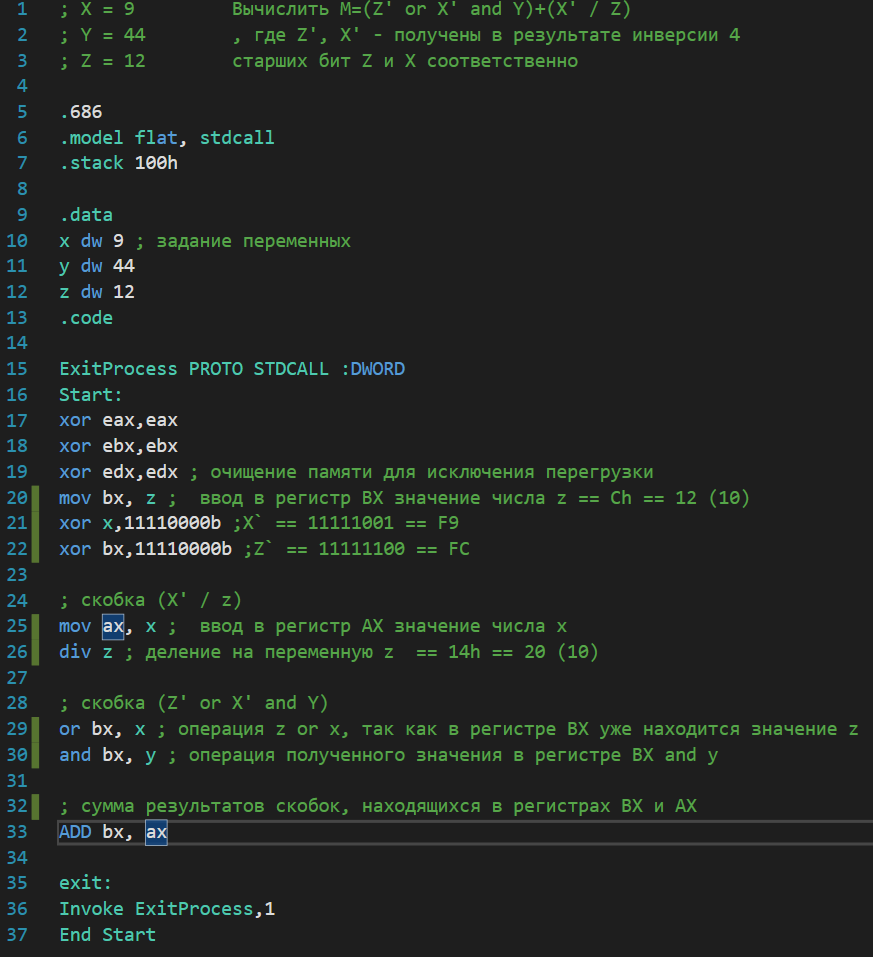
; сумма результатов скобок, находящихся в регистрах BX и AX

ADD bx, ax

exit:

Invoke ExitProcess,1

End Start



Вывод:

В данной работе я научился выполнять простейшие логические операции на языке MASM и работал с регистрами в нем.

* .686 – данная директива указывает ассемблеру использовать набор команд процессора Pentium Pro или Pentium II.
* .model flat - это несегментированная модель памяти, используемая в 32-разрядных операционных системах. В этой модели все данные и код программы размещаются в одном логическом адресном пространстве, что упрощает адресацию памяти. Сегментные регистры не используются для адресации данных, а вместо них используются 32-разрядные смещения
* .model stdcall - это соглашение о вызовах процедур, которое определяет порядок передачи параметров и очистки стека. При использовании stdcall, параметры передаются через стек в обратном порядке, а очистка стека производится вызываемой процедурой
* .stack100h – данная директива определяет размер стека программы. В данном случае, стек будет иметь размер 256 байт (100h — это шестнадцатеричное представление 256 в десятичной системе)
* ExitProcess PROTO STDCALL: DWORD:
  + ExitProcess:
    - Это имя функции из Windows API, которая завершает выполнение текущего процесса.
    - Она принимает один параметр — код завершения процесса (тип DWORD).
  + PROTO:
    - Ключевое слово MASM для объявления прототипа функции.
    - Прототип позволяет ассемблеру знать, как вызывать функцию, какие параметры она принимает и в каком порядке.
  + STDCALL:
    - Указывает соглашение о вызове функции.
    - В соглашении stdcall параметры передаются через стек в обратном порядке (сначала последний параметр), а очистка стека выполняется вызываемой функцией.
    - Это стандартное соглашение для большинства функций Windows API.
  + DWORD:
    - Описывает тип параметра функции. В данном случае функция принимает один параметр типа DWORD (32-битное беззнаковое целое число).
* API (Application Programming Interface) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.