ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		Т. В. Семененко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2

ПРЕДСТАВЛЕНИЕ ДАННЫХ. АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОПЕРАЦИИ

по курсу:

АРХИТЕКТУРА ЭВМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1 Цель работы

Цель работы: изучение архитектуры МП Intel 8086, изучение структуры простейшей ассемблерной программы, ознакомление с системой арифметико-логических команд процессора, организация вычислений на языке ассемблера.

2 Задание

Практическая часть работы включает выполнение следующих действий:

- формирование числовых значений в соответствии с индивидуальным заданием, определение минимального формата представления исходных данных;
- по заданному алгоритму составление и выполнение программы работы с данными.

Правильность разработки и выполнения программ арифметико-логической обработки данных контролируется путем ручной трассировки заданных алгоритмов с последующим сравнением результатов работы программ с результатами ручной трассировки.

Значения исходных данных, которые должны храниться в сегменте данных (согласно №В=7, №Г=26):

$$X1 = -7;\ X2 = 182;\ X3 = -208;\ X4 = 26.$$

Алгоритм построчно (№В=7):

1. X3+X2
2. X1/X3
3. X2 xor X1
4. X3-X1-CF
5. $X4*2^4$

3 Ход выполнения

3.1 Исходные данные

Сперва необходимо входные данные перевести в шестнадцатеричную систему в формате 8- и 16-битных двухкомплементарных чисел:

$$-\mathrm{X1} = 7_{10} = 07_{16} \rightarrow \mathrm{X1} = -7_{10} = \mathrm{F9}_{16};$$

$$\mathrm{X2} = 182_{10} = \mathrm{00B6}_{16};$$

$$-\mathrm{X3} = 208_{10} = \mathrm{D0}_{16} \rightarrow \mathrm{X3} = -208_{10} = \mathrm{FF30}_{16};$$

$$\mathrm{X4} = 26_{10} = \mathrm{1A}_{16}.$$

3.2 Трассировка заданного алгоритма

В таблице 1 приведена трассировка заданного алгоритма.

Таблица 1 — Трассировка алгоритма

Шаг	Операция	Входные данные	Результат (10)	Результат (16, 16-bit)
0	Исходные данные	X1=F9h, X2=00B6h, X3=FF30h, X4=1Ah	—	—
1	X3 + X2	–208 + 182	–26	FFE6
2	X1 / X3	–7 / –208	0 (ост. –7)	0000 (ост. FFF9)
3	X2 XOR X1	00B6 XOR FFF9	–177	FF4F
4	X3 – X1 – CF	–26 – (–7)	–19	FFED
5	X4 * 2^4	26*16	416	01A0

3.3 Программа заданного алгоритма в мнемокодах

На языке Assembly под архитектуру Intel 8086 была написана программа, реализующая описанный алгоритм.

Листинг 1 — Листинг программы

```
; input
MOV AL, 0F9h      ; X1 = -7
MOV BX, 000B6h    ; X2 = 182
MOV CX, 0FF30h    ; X3 = -208
MOV DL, 01Ah      ; X4 = 26

; X3 = X3 + X2
ADD CX, BX        ; CX = X3 + X2

; DX:AX = X1 / X3
CBW               ; sign extend byte AL -> word AX
```

```
CWD               ; sign-extend word AX to  dword DX:AX (if AX<0: DX = FFFFh, else
zero) - IDIV takes 32-bit DWORD DX:AX as devidend
IDIV CX           ; (DX:AX) / CX -> AX=quotient, DX=remainder

; X2 = X2 XOR X1
MOV AX, 0FFF9h    ; AX = X1 after IDIV
XOR BX, AX        ; BX = X2 XOR X1

; X3 = X3 - X1 - CF
SBB CX, AX        ; CX = X3 - X1 - carry_flag

; X4 = X4 * 2^4
MOV DX, 0001Ah    ; DX = X4 after IDIV
SHL DX, 4         ; DX = X4 * 16 (bitwise left shift)
```

4 Вывод

В ходе выполнения лабораторной работы была изучена работа с целыми числами в различных разрядных форматах и проведена трассировка алгоритма с использованием исходных данных. Были рассмотрены операции сложения, вычитания, деления, побитового XOR и умножения на степень двойки в контексте архитектуры ЭВМ, что позволило на практике закрепить понимание того, как компьютер обрабатывает знаковые и беззнаковые числа.

Особое внимание уделялось выбору минимально необходимого формата представления чисел. На примере совместного использования 8- и 16-битного формата было показано, что корректная работа алгоритма зависит от того, чтобы все значения помещались в выбранный разрядный диапазон, а также учитывались особенности представления отрицательных чисел в дополнительном коде. Это позволило понять, как правильно интерпретировать результаты операций и контролировать переполнение и перенос.

Кроме того, был составлен ассемблерный код алгоритма, что дало возможность на практике увидеть соответствие машинных команд математическим операциям. Создание таблицы трассировки позволило проследить пошаговое изменение значений переменных и убедиться в правильности выполнения каждой операции. В результате работы были закреплены навыки перевода чисел между системами счисления, побитовой арифметики и анализа работы программ на уровне команд процессора.