Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**Отчёт о выполнении лабораторной работы №2 по дисциплине «Системы реального времени»**

Работу выполнил студент 44 группы А. А. Иванов

Работу проверил доц. каф. ИТ А. Н. Полетайкин

Краснодар

2024

**Изучение команд арифметических и логических операций**

Цель: изучение команд арифметических и логических операций, приобретение практических навыков при их использовании в различных вычислительных задачах.

**Задание**

1. Составить программу для расчета заданного арифметического выражения. Длину и значение переменных A, B, C выбрать самостоятельно. Константы, заданные в выражении, использовать в кодовом сегменте. Программа должна корректно работать при любых допустимых значениях переменных.

2. Описать команды умножения и деления, используемые в программе на предмет длины операндов, участвующих в операции. Охарактеризовать длину результата и место его хранения.

3. Получить загрузочный модуль и протестировать выполнение программы в отладчике.

4. На основе составленной программы выполнить следующие действия:

− загрузить в аккумулятор маскирующее слово, позволяющее определить заданную характеристику содержимого регистра DX;

− выполнить заданную проверку и ее результат сохранить в переменной RESULT, объявленной в сегменте данных.

5. Перекомпилировать загрузочный модуль и протестировать выполнение программы в отладчике.

6. Произвести расчет времени выполнения программы.

7. Сделать выводы.

Индивидуальные условия лабораторной работы представлены в таблице (Таблица 1):

|  |  |  |
| --- | --- | --- |
| Номер варианта | Заданные операции | |
| Арифметическая | Логическая |
| 10 | A/255–B/(7+C)⋅78 | Определение наличия 1 в разряде 9 |

Таблица 1 – индивидуальные условия лабораторной работы

**Ход работы**

*1. Составить программу для расчета заданного арифметического выражения. Длину и значение переменных A, B, C выбрать самостоятельно. Константы, заданные в выражении, использовать в кодовом сегменте. Программа должна корректно работать при любых допустимых значениях переменных.*

Программа, реализующая заданную арифметическую операцию, представлена ниже:

.486

.model flat

.data

const\_1 DB 255

const\_2 DB 7

const\_3 DB 78

a\_1 DW 22958

b\_1 DW 11

c\_1 DB 3

desiat DB 10

a\_1\_tsel DB ?

a\_1\_ost DB ?

c\_1\_res DB ?

b\_1\_tsel DB ?

b\_1\_ost DB ?

b\_1\_mul\_tsel DW ?

b\_1\_mul\_ost DW ?

res\_tsel DW ?

res\_ost DW ?

check\_tsel DB ?

check\_ost DB ?

RESULT DB ?

.code

\_start:

;A/255

mov ax, a\_1

xor dx, dx

div const\_1

mov a\_1\_tsel, al

mov a\_1\_ost, ah

;7+C

mov ah, const\_2

mov bl, c\_1

add ah, bl

mov c\_1\_res, ah

;B/(7+C)

mov ax, b\_1

xor dx, dx

div c\_1\_res

mov b\_1\_tsel, al

mov b\_1\_ost, ah

;B/(7+C)\*78

mov al, b\_1\_tsel

mul const\_3

mov b\_1\_mul\_tsel, ax

mov al, b\_1\_ost

mul const\_3

mov b\_1\_mul\_ost, ax

mov ax, b\_1\_mul\_ost

xor dx, dx

div desiat

movzx bx, ah

mov b\_1\_mul\_ost, bx

movzx bx, al

mov ax, b\_1\_mul\_tsel

add ax, bx

mov b\_1\_mul\_tsel, ax

;A/255-B/(7+C)\*78

movzx ax, a\_1\_ost

mov bx, b\_1\_mul\_ost

sub ax, bx

mov res\_ost, ax

movzx ax, a\_1\_tsel

mov bx, b\_1\_mul\_tsel

sbb ax, bx

mov res\_tsel, ax

;проверка целой части на наличие 1 в разряде 9

mov ax, res\_tsel

and ax, 0200h

jz not\_set\_tsel

mov check\_tsel, 1

not\_set\_tsel:

mov check\_tsel, 2

;аналогичная проверка остатка

mov ax, res\_ost

and ax, 0200h

jz not\_set\_ost

mov check\_ost, 1

not\_set\_ost:

mov check\_ost, 2

;такая же проверка в DX

mov ax, 0200h

and dx, ax

jz not\_set\_dx

mov RESULT, 1

not\_set\_dx:

mov RESULT, 2

ret

end \_start

В коде программы, представленном выше, подобраны такие значения свободных переменных (A, B, C) чтобы продемонстрировать умение программы работать как с целыми, так и с дробными значениями:

22958/255 = 90 (целая часть) 8 (остаток)

7+3 = 10

11/10 = 1 (целая часть) 1 (остаток)

1.1\*78 = 78 (целая часть) 78 (остаток) -> преобразуется в 85 (целая часть) 8 (остаток)

90 (целая часть) 8 (остаток) – 85 (целая часть) 8 (остаток) = 5

*2. Описать команды умножения и деления, используемые в программе на предмет длины операндов, участвующих в операции. Охарактеризовать длину результата и место его хранения.*

В коде программы используются такие операторы деление и умножения как DIV и MUL.

При делении в первом слагаемом (A/255) используется деление слова (DW) на байт (DB), в результате которого целая часть и остаток помещаются в память (в соответствующие переменные, объявленные в участке .data) по адресу 00402000+10 и 00402000+11.

При делении после сложения (B/(7+C)) используется деление слова (DW) на байт (DB), а результат записывается в соответствующие переменные по адресу 00402000+13 и 00402000+14.

При умножении после деления (B/(7+C)\*78) используется умножение байтов (DB). Здесь отдельно умножается целая часть после деление и остаток, после чего происходит пересчёт остатка: остаток делится на 10 (слово DW на байт DB), после чего целая часть прибавляется к изначальной целой части, а остаток перезаписывается на новый остаток. Результат всех операций сохраняется в формате DW по адресу 00402000+15 и 00402010+1.

Кроме вышеописанных операций перед делением также происходит зануление регистров DX через логическую команду XOR. Это выполняется для того, чтобы данные, хранящиеся в этом регистре после каких-то других операций, не повлияли на результат деления.

*3. Получить загрузочный модуль и протестировать выполнение программы в отладчике.*

Результат выполнения программы представлен ниже на рисунках (Рисунок 1, Рисунок 2):

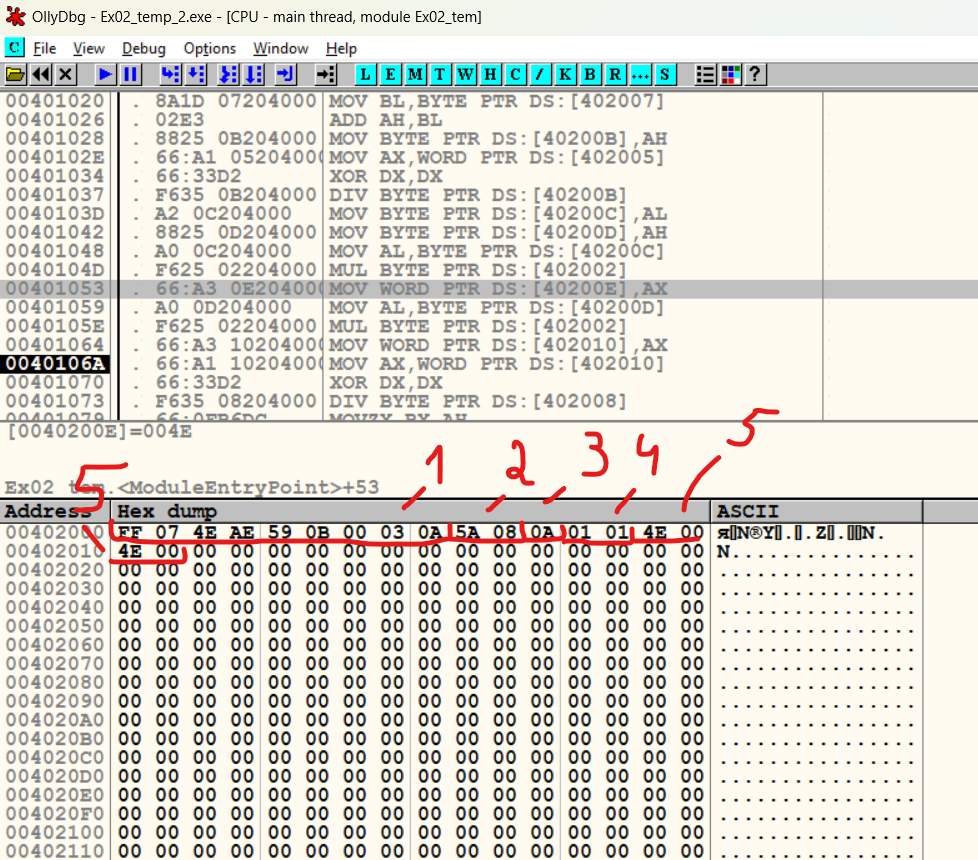


Рисунок 1 – результат работы программы, этап 1

Где:

1 – переменные, объявленные в блоке .data изначально (константы и A, B, C, а также 10 для деления остатка);

2 – результат деления A/255;

3 – результат сложения 7+С;

4 – результат деления B/(7+C);

5 – результат умножения B/(7+C)\*78 без пересчёта остатка.

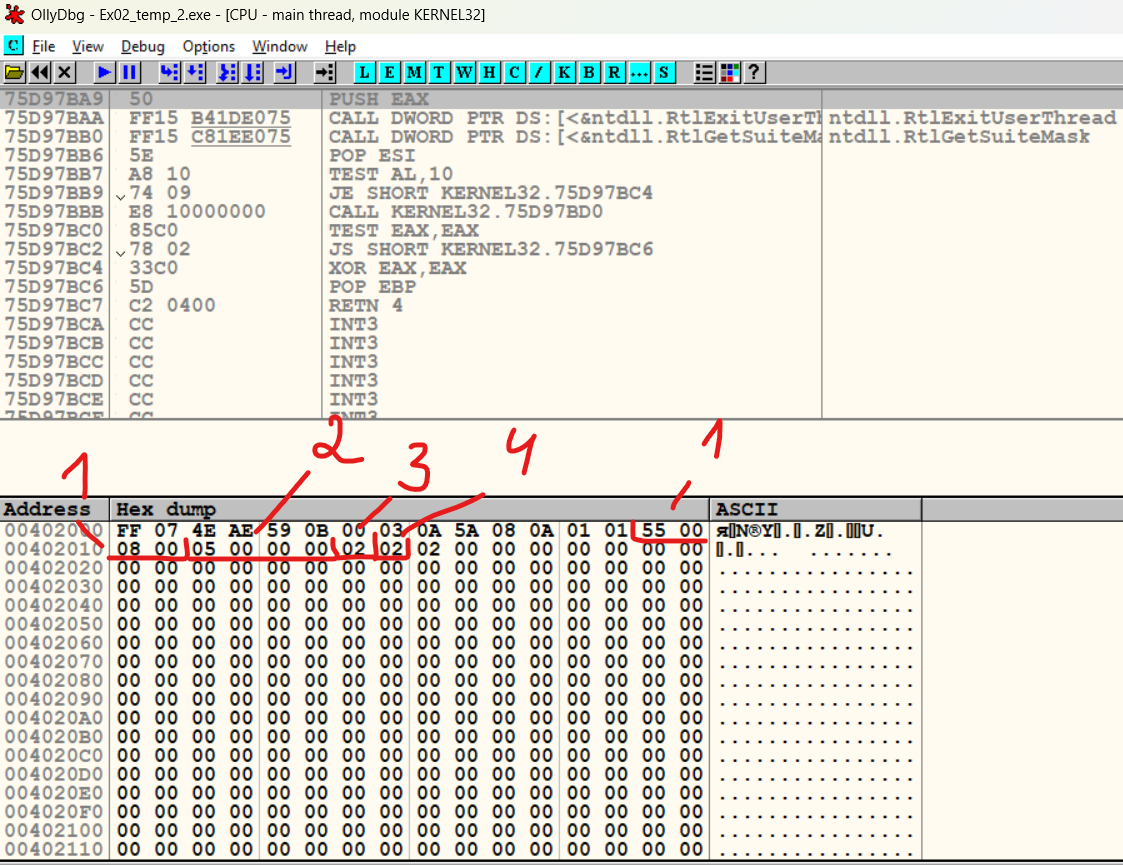


Рисунок 2 – результат работы программы, этап 2

Где:

1 – результат умножения B/(7+C)\*78 с пересчитанным остатком;

2 – результат всей операции A/255-B/(7+C)\*78 (целая часть и остаток);

3 – результат логической операции в целой части результата (2 – 1 отсутствует на месте 9 бита);

4 – результат логической операции в остатке результата (2 – 1 отсутствует на месте 9 бита).

*4. На основе составленной программы выполнить следующие действия:*

*− загрузить в аккумулятор маскирующее слово, позволяющее определить заданную характеристику содержимого регистра DX;*

*− выполнить заданную проверку и ее результат сохранить в переменной RESULT, объявленной в сегменте данных.*

*5. Перекомпилировать загрузочный модуль и протестировать выполнение программы в отладчике.*

Ниже, на рисунке (Рисунок 3) представлен результат работы проверки логической операции в регистре DX с загрузкой маски в аккумулятор:

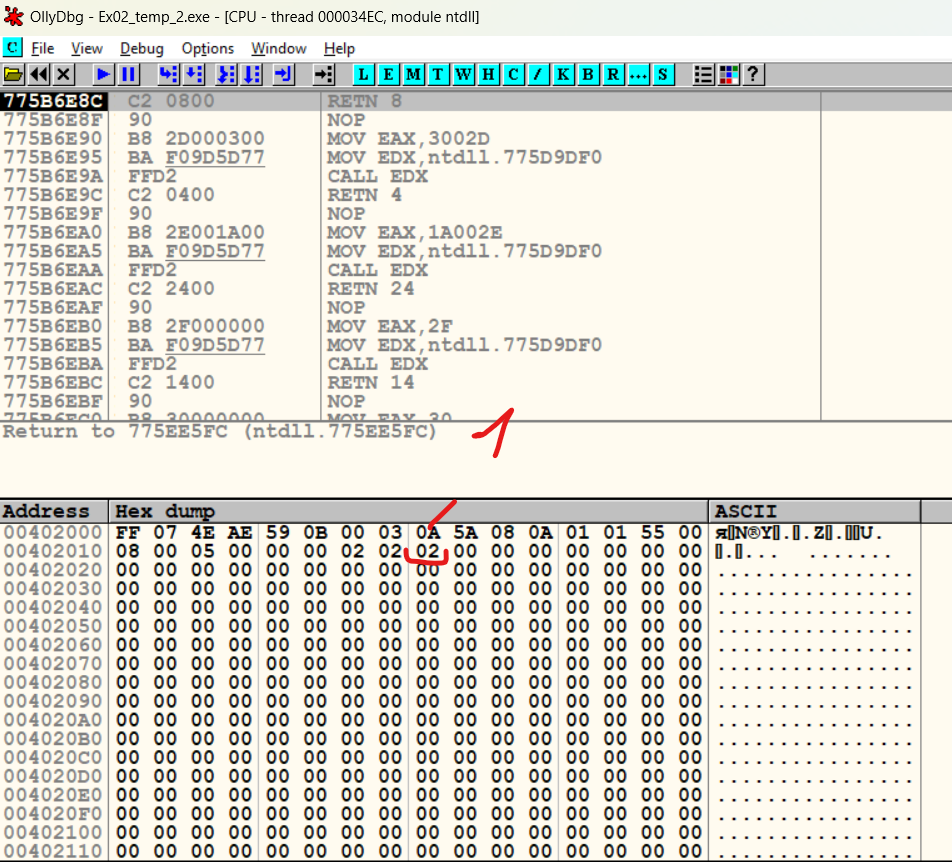


Рисунок 3 – результат проверки DX

Где:

1 – результат логической операции в регистре DX (2 – 1 отсутствует на месте 9 бита).

*6. Произвести расчет времени выполнения программы.*

Ниже представлен код программы с рассчитанным временем выполнения:

.code

\_start:

;A/255

mov ax, a\_1 8(12)+ЕА=12+6=18

xor dx, dx 3

div const\_1 (86-96)+ЕА=96+6=102

mov a\_1\_tsel, al 9(13)+ЕА=13+6=19

mov a\_1\_ost, ah 9(13)+ЕА=13+6=19

;7+C

mov ah, const\_2 8(12)+ЕА=12+6=18

mov bl, c\_1 8(12)+ЕА=12+6=18

add ah, bl 3

mov c\_1\_res, ah 9(13)+ЕА=13+6=19

;B/(7+C)

mov ax, b\_1 8(12)+ЕА=12+6=18

xor dx, dx 3

div c\_1\_res (86-96)+ЕА=96+6=102

mov b\_1\_tsel, al 9(13)+ЕА=13+6=19

mov b\_1\_ost, ah 9(13)+ЕА=13+6=19

;B/(7+C)\*78

mov al, b\_1\_tsel 8(12)+ЕА=12+6=18

mul const\_3 (76-83)+ЕА=83+6=89

mov b\_1\_mul\_tsel, ax 9(13)+ЕА=13+6=19

mov al, b\_1\_ost 8(12)+ЕА=12+6=18

mul const\_3 (76-83)+ЕА=83+6=89

mov b\_1\_mul\_ost, ax 9(13)+ЕА=13+6=19

mov ax, b\_1\_mul\_ost 8(12)+ЕА=12+6=18

xor dx, dx 3

div desiat (86-96)+ЕА=96+6=102

movzx bx, ah 2+ЕА=2+6=8

mov b\_1\_mul\_ost, bx 9(13)+ЕА=13+6=19

movzx bx, al 2+ЕА=2+6=8

mov ax, b\_1\_mul\_tsel 8(12)+ЕА=12+6=18

add ax, bx 3

mov b\_1\_mul\_tsel, ax 9(13)+ЕА=13+6=19

;A/255-B/(7+C)\*78

movzx ax, a\_1\_ost 8(12)+ЕА=12+9=21

mov bx, b\_1\_mul\_ost 8(12)+ЕА=12+6=18

sub ax, bx 3

mov res\_ost, ax 9(13)+ЕА=13+6=19

movzx ax, a\_1\_tsel 8(12)+ЕА=12+9=21

mov bx, b\_1\_mul\_tsel 8(12)+ЕА=12+6=18

sbb ax, bx 3

mov res\_tsel, ax 9(13)+ЕА=13+6=19

;проверка целой части на наличие 1 в разряде 9

mov ax, res\_tsel 8(12)+ЕА=12+6=18

and ax, 0200h 4

jz not\_set\_tsel 16(4)

mov check\_tsel, 1 10(14)+ЕА=14+6=20

not\_set\_tsel: 16

mov check\_tsel, 2 10(14)+ЕА=14+6=20

;аналогичная проверка остатка

mov ax, res\_ost 8(12)+ЕА=12+6=18

and ax, 0200h 4

jz not\_set\_ost 16(4)

mov check\_ost, 1 10(14)+ЕА=14+6=20

not\_set\_ost: 16

mov check\_ost, 2 10(14)+ЕА=14+6=20

;такая же проверка в DX

mov ax, 0200h 10(14)+ЕА=14+6=20

and dx, ax 3

jz not\_set\_dx 16(4)

mov RESULT, 1 10(14)+ЕА=14+6=20

not\_set\_dx: 16

mov RESULT, 2 10(14)+ЕА=14+6=20

ret

end \_start

Итоговое время выполнения программы: 161+58+161+450+122+82+82+83=1199 тактов

Тактовая частоты вычислительного процессора (12th Gen Intel(R) Core(TM) i5-12500H) = 2.50 ГГЦ

Тогда в секундах это получится: 

Или 

*7. Сделать выводы.*

После выполнения данной работы можно сделать такой вывод: в Ассемблере арифметика реализована на базово уровне, из-за чего при работе с операциями деления, умножения, сложения и вычитания нужно учитывать не только размер используемых значений (например, нельзя поделить значение формата DD на формат DB), но и размер используемых регистров, а также нужно уметь особенно внимательно работать с целыми частями и остатками значений, т.к. Ассемблер не умеет сам корректно складывать остатки к целым частям с учётом переноса.