**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Кубанский Государственный Университет**

Кафедра информационных технологий

ОТЧЕТ

о выполнении лабораторной работы №4

по дисциплине «Системы реального времени»

Выполнил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр. 44 Иванов А.А.\_\_\_\_\_\_\_

Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. ИТ Полетайкин А.Н.

Краснодар

2024

**Программирование ввода-вывода в режиме реального времени**

Цель: изучение принципов организации ввода информации извне в УВМ и вывода информации из УВМ вовне, организации временной задержки при обработке данных на языке ассемблера, а также приобретение практических навыков программирования указанных операций.

**Задание**

Имеется СРВ, включающая в себя некоторую аппаратную часть периферийных устройств (ПУ) и ядро в виде УВМ (Рисунок 1), которая осуществляет обмен с периферией через один 16-разрядный порт ввода с заданным адресом А\_IN и один 16-разрядный порт вывода с заданным адресом А\_OUT. Входные 8-разрядные данные поступают на младший байт порта ввода.

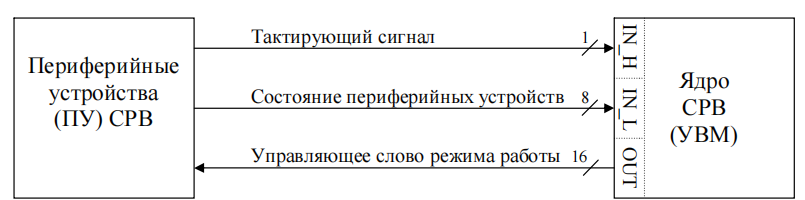


Рисунок 1 – Структурная схема СРВ

Для актуализации входных данных необходимо предварительно выводить через порт вывода заданное управляющее слово режима работы (УСРР). Отправление УСРР тактируется высоким уровнем сигнала через заданный разряд *R\_C* порта ввода. Перед отправлением УСРР его необходимо модифицировать в 2 этапа:

1. Начальное значение старшего байта УСРР (УСРР[1]) – заданная комбинация U0. При каждом следующем запросе данных значение УСРР[1] меняется на заданное приращение ΔU. Вследствие аппаратных временных затрат периферийной части актуализация данных происходит через заданный интервал времени τ.

2. Младший байт УСРР (УСРР[0]) подвергается побитовой модификации следующим образом:

− на каждой нечётной итерации заданные биты SR устанавливаются, а биты RS сбрасываются;

− на каждой чётной итерации заданные биты SR сбрасываются, а биты RS устанавливаются.

Составить программу на языке ассемблера, которая осуществляет опрос внешних устройств через порт ввода и записывает отправленные УСРР и соответствующие им данные о состоянии периферийных устройств в одномерные массивы заданной длины *N*. Реализацию задержки времени τ осуществлять при помощи подпрограммы DELAY.

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер варианта | Адреса портов | | Параметры УСРР, байт | | | | Параметры синхронизации | | |
| старший | | младший | |
| A\_IN | A\_OUT | U\_0 | ΔU | SR | RS | τ, мс | R\_C | N |
| 10 | 8A7h | 8A8h | 8Ch | 06h | 6 | 4, 7 | 51 | 10 | 20 |

Таблица 1 – индивидуальные условия лабораторной работы

**Ход работы**

*1. Подпрограмма задержки с расчётом времени задержки.*

Теоретическая часть реализации задержки.

Так как желаемое время работы подпрограммы задержки – 51 миллисекунда, то сначала нужно понять, сколько тактов на вычислительном процессоре будет выполняться такая задержка. Так как используемый процессор (12th Gen Intel(R) Core(TM) i5-12500H) имеет тактовую частоту 2.50 ГГц, то в тактах это будет:



Требуемая задержка в 51 миллисекунду это:

 секунд

Тогда требуемая задержка в тактах для вышеописанного процессора считается по формуле:



Практическая часть.

Код подпрограммы задержки представлен ниже:

\_delay:

;сохранение регистров и флагов в стеке

push ecx

push ebx

push eax

pushf

;основная программа задержки

xor eax, eax

xor ebx, ebx

xor ecx, ecx

mov cx, 1448

first\_for:

mov bx, 3516

second\_for:

dec bx

mov ax, 0

cmp bx, ax

jz second\_for

loop first\_for

;восстановление регистров и флагов из стека

popf

pop eax

pop ebx

pop ecx

ret

Вышеописанная программа работает следующим образом:

1. Так как в подпрограмме используются те же регистры, которые используются и в основной программе, то первым делом значения важных регистров и всех флагов сохраняются в стеке;

2. Зануляются все используемые в подпрограмме регистры для корректной работы со счётчиками циклов;

3. Объявляется внешний цикл в 1448 итераций, а в нём объявляется внутренний цикл в 3516 итераций, в котором происходит всего несколько действий: счётчик внутренних итераций уменьшается на 1 и сравнивается с 0; если счётчик не равен 0, то происходит повторная работа внутреннего цикла;

4. После выполнения всех циклов выполняется загрузка сохранённых ранее значений регистров и флагов из стека.

Подсчёт времени выполнения подпрограммы задержки:

1. Сохранение значений регистров и флагов в стеке: 15\*3+14=59 тактов

2. Зануление регистров: 3\*3=9 тактов

3. Внешний цикл: 4+(17 или 5)=(21 или 9) тактов (в зависимости от повторения цикла или пропуска при 0)

4. Внутренний цикл: 2+4+3+(16 или 4)=(25 или 13) тактов (в зависимости от перехода к метки или пропуска при 0)

5. Восстановление сохранённых данных: 12+12\*3=48 тактов

Общее время работы подпрограммы:

59+9+(1447\*(3515\*25+13)+4+17)+9=127 174 034 такта

127 174 034 / 2.5 = 50 869 613,6 наносекунд = 50,87 миллисекунд

*2. Основная программа.*

Код основной программы представлен ниже:

.486

.model flat

.data

port\_A\_in DW 8A7h

port\_A\_out DW 8A8h

YSRR\_1 DB 8Ch

YSRR\_0 DB 0

delta\_YSRR DB 06h

receiv\_input DW 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

new\_YSRR DW 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

.code

\_delay:

;сохранение регистров и флагов в стеке

push ecx

push ebx

push eax

pushf

;основная программа задержки

xor eax, eax

xor ebx, ebx

xor ecx, ecx

mov cx, 1448

first\_for:

mov bx, 3516

second\_for:

dec bx

mov ax, 0

cmp bx, ax

jz second\_for

loop first\_for

;восстановление регистров и флагов из стека

popf

pop eax

pop ebx

pop ecx

ret

\_start: ;основная программа

;зануление регистров перед работой

xor ecx, ecx

xor ebx, ebx

xor edx, edx

xor eax, eax

;основной цикл программы

mov cx, 20

mov ebx, 0

main\_for:

;если бит R\_c не установлен

not\_one:

;проверка R\_c на 1

mov dx, port\_A\_in

;in ax, dx

mov ax, 0FFFFh

and ax, 0400h

jz not\_one

;модификация УСРР[0]

test cx, 1

jz chet

mov YSRR\_0, 90h

jmp dalee

chet:

mov YSRR\_0, 40h

dalee:

;вывод УСРР

mov dh, YSRR\_1

mov dl, YSRR\_0

mov ax, dx

mov dx, port\_A\_out

;out dx, ax

;задержка

call \_delay

;ввод состояния ПУ СРВ

mov dx, port\_A\_in

;in ax, dx

mov ax, 0001h

mov dx, ax

;сохранение данных в массивы

mov eax, offset receiv\_input

mov [eax+ebx], dx

mov eax, offset new\_YSRR

mov dh, YSRR\_1

mov dl, YSRR\_0

mov [eax+ebx], dx

add ebx, 2

;приращение УСРР[1]

mov dl, YSRR\_1

mov dh, delta\_YSRR

add dl, dh

mov YSRR\_1, dl

;уменьшение счётчика итераций

dec cx

jnz main\_for

ret

end \_start

Вышеописанная программа работает следующим образом:

1. Зануляет используемые регистры для корректной обработки итераций и работы с адресацией к результирующим массивам;

2. Устанавливается главный счётчик итераций цикла и заранее устанавливается позиция первого числа в результирующих массивах;

3. Выполняется получение входных данных из порта ввода и проверка байта на высокий уровень сигнала (если проверка прошла успешно – выполняется тело цикла, иначе – пропуск);

4. В зависимости от чётности текущей операции выполняется модификация младшего байта УСРР (УСРР[0]);

5. Составляется УСРР и отправляется в порт вывода;

6. Вызывается подпрограмма задержки для актуализации входных данных;

7. Выполняется получение новых входных данных;

8. Выполняется сохранение результирующих данных в соответствующих массивах (полученные входные данные после актуализации и текущее УСРР);

9. Выполняется увеличение старшего байта УСРР (УСРР[1]) на заданное приращение;

10. Уменьшается счётчик итераций и выполняется проверка счётчика на 0.

*3. Работа программы и дамп памяти.*

На рисунках (Рисунок 2 – Рисунок 4) представлена работа программы на разных этапах:

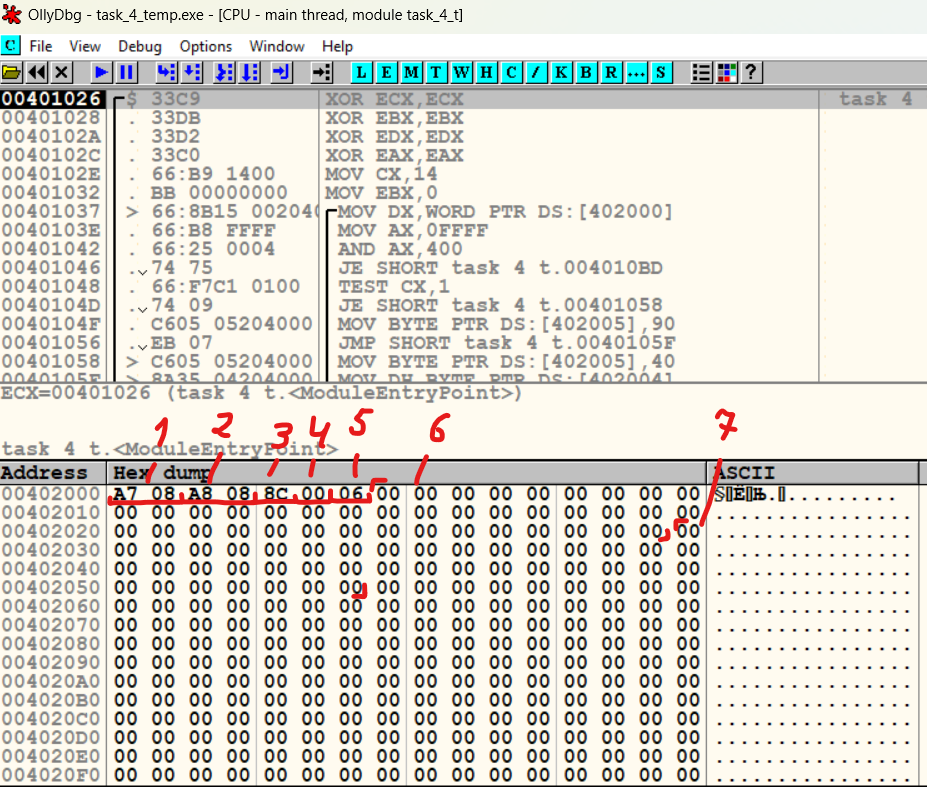


Рисунок 2 – старт работы программы

Где:

1, 2 – адреса портов ввода/вывода;

3, 4 – УСРР[1] и УСРР[0];

5 – приращение УСРР

6, 7 – адресное пространство результирующих массивов (полученных данных и изменённых УСРР).

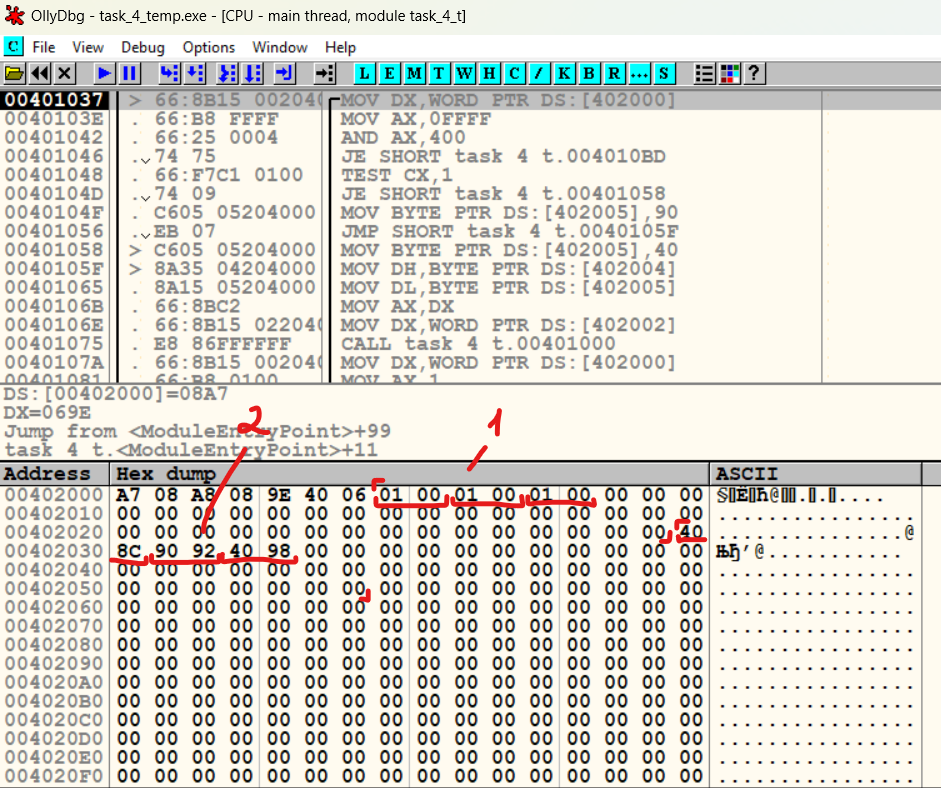


Рисунок 3 – промежуточный этап программы (прошло 3 итерации)

Где:

1 – массив полученных данных (т.к. эмулируем работу с портами, то получаем всегда одно и то же значение - 1);

2 – массив изменённых УСРР (здесь видно, что младший байт то 40, то 90, т.е. чередуется обозначение чётности итерации; а старший байт увеличивается на величину приращения).

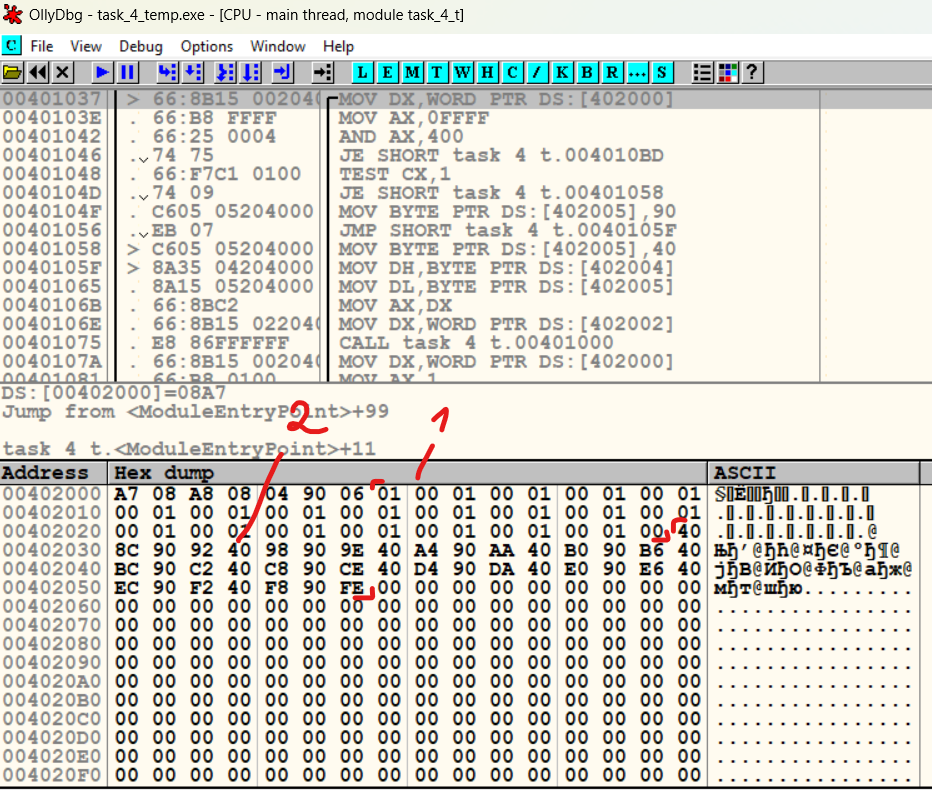


Рисунок 4 – окончание работы программы (массивы заполнены)

Где:

1 и 2 – полностью заполненные массивы входных данных и изменённых УСРР (по 20 значений).

*4. Выводы.*

В ходе выполнения данной лабораторной работы была проделана работа по вычислению точного времени задержки подпрограммы. Было решено ограничиться минимальным количеством операций внутри самих циклов (только уменьшение счётчика итераций и сравнение с 0) с целью упрощения работы по подсчёту точного времени через количество итераций циклов (например, команды NOP вообще не использовались за ненадобностью).

Также было проведено разделение кода на 2 блока – основная программа и вспомогательная подпрограмма.

Для более гибкой и корректной работы также пришлось использовать стек для хранения значений регистров и флагов, что позволило избежать непредсказуемого поведения программы.