Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5

по дисциплине «Основы профессиональной деятельности»

Асинхронный обмен данными с ВУ

Вариант №43889

Группа: P3112

Выполнил: Балин А. А.

Проверил: Осипов С. В.

г. Санкт-Петербург, 2023

Оглавление

Введение

В данной лабораторной работе я изучу реализацию комплекса программ в БЭВМ.

Задание

По выданному преподавателем варианту разработать программу асинхронного обмена данными с внешним устройством. При помощи программы осуществить ввод или вывод информации, используя в качестве подтверждения данных сигнал (кнопку) готовности ВУ.

Программа по моему варианту

Лабораторная работа №5

По выданному преподавателем варианту разработать программу асинхронного обмена данными с внешним устройством. При помощи программы осуществить ввод или вывод информации, используя в качестве подтверждения данных сигнал (кнопку) готовности ВУ.

Введите номер варианта 43889

1. Программа осуществляет асинхронный вывод данных на ВУ-1
2. Программа начинается с адреса $1C2_{16}$. Размещаемая строка находится по адресу 556_{16}.
3. Строка должна быть представлена в кодировке ISO-8859-5.
4. Формат представления строки в памяти: АДР1: СИМВ1 СИМВ2 АДР2: СИМВ3 СИМВ4 ... СТОП_СИМВ.
5. Ввод или вывод строки должен быть завершен по символу с кодом 00 (NUL). Стоп символ является обычным символом строки и подчиняется тем же правилам расположения в памяти что и другие символы строки.

Текст программы

Текст программы	Описание
ORG 0x1C2	Регистр начала
LPART: WORD 0xFF00	Для очистки младшей части слова
RPART: WORD 0x00FF	Для очистки старшей части слова
CHR: WORD 0x0556	Ячейка с адресом текущих 2 символов
START: CLA	Очистка аккумулятора
WRITING: CALL $CHECK_W1	Вызов цикла spin-loop
LD (CHR)	Загрузка текущих 2 символов
AND $LPART	Очистка младшей части
BEQ STOP	Проверка на NUL (00) символ
SWAB	Старшая часть в младшую
OUT 2	Вывод на ВУ-1
CALL $CHECK_W1	Вызов цикла spin-loop
LD (CHR)	Загрузка текущих 2 символов
AND $RPART	Выделение младшей части
BEQ STOP	Проверка на NUL (00) символ
OUT 2	Вывод на ВУ-1
LD $CHR	Загрузка адреса текущей ячейки
INC	CHR++
ST $CHR	Установка в CHR адрес следующей ячейки
JUMP $WRITING	Переход в новый цикл Writing
STOP: HLT	Останов.
CHECK_W1:	Функция для проверки готовности записи в ВУ-1
IN 3	Бит готовности ВУ-1 в AC
AND #0x40	Если бит был 1, то 0x40, иначе 0
BEQ CHECK_W1	Если 0, в начало функции
RET	Возврат из функции
ORG 0x556	Адрес первого символа
CHR12: WORD 0x77BB	Символы: wЛ
CHR34: WORD 0x3231	Символы: 21
CHR56: WORD 0x21DA	Символы: !к
CHR78: WORD 0x2300	Символы: #NUL

Строка: wЛ21!к#

Символы в ISO-8859-5: w: 77_{16}, Л: BB_{16}, 2: 32_{16}, 1: 31_{16}, ! : 21_{16}, к: DA_{16}, #: 23_{16}

Символы в UTF-8: 77_{16}, $41B_{16}$, 32_{16}, 31_{16}, 21_{16}, $43A_{16}$, 23_{16}

Символы в UTF-16: $FEFF0077_{16}$, $FEFF041B_{16}$, $FEFF0032_{16}$, $FEFF0031_{16}$, $FEFF0021_{16}$, $FEFF043A_{16}$, $FEFF0023_{16}$

Выполнение

Строка: Q!ж$

ISO-8859-5:

Q: 51_{16}

!: 21_{16}

ж: $D6_{16}$

$: 24_{16}

CHR12: WORD 0x5121

CHR34: WORD 0xD624

Адрес	Знчн	IP	CR	AR	DR	SP	BR	AC	PS	NZVC	Адр	Знчн
1C2	FF00	1C3	FF00	1C2	FF00	0	01C2	0	4	100		
1C3	00FF	1C4	00FF	1C3	00FF	0	01C3	0	4	100		
1C4	556	1C5	556	1C4	0	0	01C4	0	4	100		
1C5	200	1C6	200	1C5	200	0	01C5	0	4	100		
1C6	D1D6	1D6	D1D6	7FF	01C7	7FF	D1D6	0	4	100	7FF	01C7
1D6	1203	1D7	1203	1D6	1203	7FF	01D6	0	4	100		
1D7	2F40	1D8	2F40	1D7	40	7FF	40	0	4	100		
1D8	F0FD	1D6	F0FD	1D8	F0FD	7FF	FFFD	0	4	100		
1D9	0A00	1C7	0A00	7FF	01C7	0	01D9	40	0	0		
1C7	A8FC	1C8	A8FC	556	5121	0	FFFC	5121	0	0		
1C8	21C2	1C9	21C2	1C2	FF00	0	01C8	5100	0	0		
1C9	F00B	1CA	F00B	1C9	F00B	0	01C9	5100	0	0		
1CA	680	1CB	680	1CA	680	0	01CA	51	0	0		
1CB	1302	1CC	1302	1CB	1302	0	01CB	51	0	0		
1CC	D1D6	1D6	D1D6	7FF	01CD	7FF	D1D6	51	0	0	7FF	01CD
1D6	1203	1D7	1203	1D6	1203	7FF	01D6	0	0	0		
1D7	2F40	1D8	2F40	1D7	40	7FF	40	0	4	100		
1D8	F0FD	1D6	F0FD	1D8	F0FD	7FF	FFFD	0	4	100		
1D9	0A00	1CD	0A00	7FF	01CD	0	01D9	40	0	0		
1CD	A8F6	1CE	A8F6	556	5121	0	FFF6	5121	0	0		
1CE	21C3	1CF	21C3	1C3	00FF	0	01CE	21	0	0		
1CF	F005	1D0	F005	1CF	F005	0	01CF	21	0	0		
1D0	1302	1D1	1302	1D0	1302	0	01D0	21	0	0		
1D1	A1C4	1D2	A1C4	1C4	556	0	01D1	556	0	0		
1D2	700	1D3	700	1D2	700	0	01D2	557	0	0		
1D3	E1C4	1D4	E1C4	1C4	557	0	01D3	557	0	0	1C4	557
1D4	C1C6	1C6	C1C6	1D4	C1C6	0	01D4	557	0	0		
1C6	D1D6	1D6	D1D6	7FF	01C7	7FF	D1D6	557	0	0	7FF	01C7
1D6	1203	1D7	1203	1D6	1203	7FF	01D6	500	0	0		
1D7	2F40	1D8	2F40	1D7	40	7FF	40	0	4	100		
1D8	F0FD	1D6	F0FD	1D8	F0FD	7FF	FFFD	0	4	100		

1D9	0A00	1C7	0A00	7FF	01C7	0	01D9	40	0	0		
1C7	A8FC	1C8	A8FC	557	D624	0	FFFC	D624	8	1000		
1C8	21C2	1C9	21C2	1C2	FF00	0	01C8	D600	8	1000		
1C9	F00B	1CA	F00B	1C9	F00B	0	01C9	D600	8	1000		
1CA	680	1CB	680	1CA	680	0	01CA	00D6	0	0		
1CB	1302	1CC	1302	1CB	1302	0	01CB	00D6	0	0		
1CC	D1D6	1D6	D1D6	7FF	01CD	7FF	D1D6	00D6	0	0	7FF	01CD
1D6	1203	1D7	1203	1D6	1203	7FF	01D6	0	0	0		
1D7	2F40	1D8	2F40	1D7	40	7FF	40	0	4	100		
1D8	F0FD	1D6	F0FD	1D8	F0FD	7FF	FFFD	0	4	100		
1D9	0A00	1C7	0A00	7FF	01C7	0	01D9	40	0	0		
1C7	A8FC	1C8	A8FC	558	0	0	FFFC	0	4	100		
1C8	21C2	1C9	21C2	1C2	FF00	0	01C8	0	4	100		
1C9	F00B	1D5	F00B	1C9	F00B	0	000B	0	4	100		
1D5	100	1D6	100	1D5	100	0	01D5	0	4	100		

Задание на защиту

Разложение на простейшие. 😒

```
ORG 0x0000
ORG 0x0040
START:
JUMP $INPUT_INIT
START_FIND:CLA
CURRENT_SIMPLE_ID: WORD 0x0000
SUS_SIMPLE: WORD 0x0000
NEEDED: WORD 0x003C
CLA
LD #2
ST (CURRENT_SIMPLE_ID)+
INC
ST $SUS_SIMPLE
FINDING_SIMPLE: LD $CURRENT_SIMPLE_ID
    CMP $NEEDED
    BEQ START_DEREF
    LD $SUS_SIMPLE
    PUSH
    LD $CURRENT_SIMPLE_ID
    PUSH
    CALL $IS_SIMPLE
    SWAP
    POP
    SWAP
    POP
    BPL NEXT_SUS
    LD $SUS_SIMPLE
    ST (CURRENT_SIMPLE_ID)+
    NEXT_SUS: LD $SUS_SIMPLE
    ADD #2
    ST $SUS_SIMPLE
    JUMP $FINDING_SIMPLE

GET_CURRENT_SIMPLE:
    INSIDE_DEREF: WORD 0x0000
    LD &1
    ST $INSIDE_DEREF
    LD (INSIDE_DEREF)
    RET

INPUT_INIT:
    NUM: WORD 0x0000
    NEWNUM: WORD 0x0000
    INPUT_NUM: IN 0x1D
    AND #0x40
    BEQ INPUT_NUM
```

```
    IN 0x1C
    ST $NEWNUM
    PUSH
    CALL CHECK_NUMERIC
    SWAP
    POP
    BMI ERR
    BEQ START_FIND
    LD $NUM
    PUSH
    LD $NEWNUM
    PUSH
    CALL $NEW_DEC_NUM
    SWAP
    POP
    POP
    ST $NUM
    BMI ERR
    JUMP $INPUT_NUM

CHECK_NUMERIC:
    CMP #0xF
    BEQ START_OUT
    CMP #0xA
    BEQ ERR_OUT
    JUMP NUMERIC
    ERR_OUT: LD #-1
    RET
    START_OUT: LD #0
    RET
    NUMERIC: LD #1
    RET

DEREF: WORD ?

IS_SIMPLE: CLA ;STACK: DIVIDED;TEMP;INDEX;RET;CURRENT_SIMPLE;N
    PUSH
    PUSH
    PUSH
    CHECKING_SIMPLE_DEVISORS: LD &2
    CMP &4
    BPL IS_SIMPLE_OUT_P
    LD &2
    ST $DEREF
    LD &5
    OTN: SUB (DEREF)
    ST &1
    LD &0
    INC
    ST &0
    LD &1
```

```
    BMI IS_DIVISOR_OUT
    BEQ IS_DIVISOR_OUT
    JUMP $OTN
    IS_DIVISOR_OUT: BEQ IS_SIMPLE_OUT_N
    LD &2
    INC
    ST &2
    LD &1
    JUMP $CHECKING_SIMPLE_DEVISORS
    IS_SIMPLE_OUT_P: SWAP
    POP
    SWAP
    POP
    SWAP
    POP
    LD #-1
    RET
    IS_SIMPLE_OUT_N:
    LD &0
    SWAP
    POP
    SWAP
    POP
    SWAP
    POP
    RET

START_DEREF:
    JUMP $STARTT

ERR: IN 0x15
    AND #0x40
    BEQ ERR
    LD #0
    OUT 0x14
    LD #0x1B
    OUT 0x14
    LD #0x2B
    OUT 0x14
    LD #0x3B
    OUT 0x14
    LD #0x4B
    OUT 0x14
    LD #0x5B
    OUT 0x14
    LD #0x6B
    OUT 0x14
    LD #0x7B
    OUT 0x14
    HLT
```

```
NEW_DEC_NUM:CLA
    PUSH;COUNT
    PUSH;TEMP
    LD #9
    ST &1
    LD &4
    ST &0
    ADDING: LD &4
    ADD &0
    ST &4
    LD &1
    DEC
    ST &1
    BEQ EXIT_NEW_DEC_NUM
    JUMP $ADDING
    EXIT_NEW_DEC_NUM: LD &4
    ADD &3
    ST &4
    SWAP
    POP
    SWAP
    POP
    RET
ORG 0x0100
STARTT: CLA
    VIVOD_1: WORD 0x000B
    VIVOD_2: WORD 0x001B
    VIVOD_3: WORD 0x002B
    VIVOD_4: WORD 0x003B
    VIVOD_5: WORD 0x004B
    VIVOD_6: WORD 0x005B
    VIVOD_7: WORD 0x006B

    LD $NUM
    CHECK_2_AS_DIVISOR: AND #1
    BEQ TWO_IS_DIVISOR
    JUMP $END_CHECK_2_AS_DIVISOR
    TWO_IS_DIVISOR: LD #2
    PUSH
    CALL $NEW_NUM_TO_PRINT
    SWAP
    POP
    CALL $PRINT
    LD #0xB
    PUSH
    CALL $NEW_NUM_TO_PRINT
    SWAP
    POP
    CALL $PRINT
    LD $NUM
    ASR
```

```
ST $NUM
JUMP $CHECK_2_AS_DIVISOR
END_CHECK_2_AS_DIVISOR:
LD $NUM
CMP #1
BEQ HHLT_DEREF
NUM_ON_VIVOD: WORD 0x0000
NUMERIC_ON_VIVOD: WORD 0x0000
LD #1
ST $CURRENT_SIMPLE_ID
NEXT_SIMPLE: LD $CURRENT_SIMPLE_ID
PUSH
CALL $GET_CURRENT_SIMPLE
SWAP
POP
CUR_SIM_DIVIDED: PUSH
LD $NUM
PUSH
CALL $CHECK_IF_DIVIDED
BMI CLEAR_AND_NEXT_SIMPLE
ST $NUM
POP
POP
ST $NUM_ON_VIVOD
NEXT_NUMERIC: PUSH
CALL $GET_LAST_NUM
ST $NUMERIC_ON_VIVOD
POP
ST $NUM_ON_VIVOD
LD $NUMERIC_ON_VIVOD
PUSH
CALL $SET_T
SWAP
POP
LD $NUM_ON_VIVOD
BNE PRINT_NUMERIC_AND_GET_NEXT
BEQ PRINT_NUMERIC_AND_GO
PRINT_NUMERIC_AND_GET_NEXT:
LD $NUM_ON_VIVOD
JUMP $NEXT_NUMERIC
PRINT_NUMERIC_AND_GO:
T_GETTING: CALL $GET_T
PUSH
CALL $NEW_NUM_TO_PRINT
SWAP
POP
LD $CURRENT_T
BEQ LAST_NUM
JUMP $T_GETTING
LAST_NUM:
LD #0xB
```

```
    PUSH
    CALL $NEW_NUM_TO_PRINT
    SWAP
    POP
    CALL $PRINT
    LD $NUM
    CMP #1
    BEQ HHLT_DEREF
    LD $CURRENT_SIMPLE_ID
    PUSH
    CALL $GET_CURRENT_SIMPLE
    SWAP
    POP
    JUMP $CUR_SIM_DIVIDED
    CLEAR_AND_NEXT_SIMPLE: SWAP
    POP
    SWAP
    POP
    LD $CURRENT_SIMPLE_ID
    INC
    ST $CURRENT_SIMPLE_ID
    CMP $NEEDED
    BEQ PRINT_WHAT_OTHER_AND_HLT
    JUMP $NEXT_SIMPLE
    PRINT_WHAT_OTHER_AND_HLT:
    LD $NUM
    BEQ HHLT_DEREF
    PRINTING_LAST:
    PUSH
    CALL $GET_LAST_NUM
    ST $NUMERIC_ON_VIVOD
    POP
    ST $NUM_ON_VIVOD
    LD $NUMERIC_ON_VIVOD
    PUSH
    CALL $SET_T
    SWAP
    POP
    LD $NUM_ON_VIVOD
    BEQ PRINT_LAST
    JUMP $PRINTING_LAST

PRINT_LAST:
    NOP
    T_GETTING_NEW: CALL $GET_T
    PUSH
    CALL $NEW_NUM_TO_PRINT
    SWAP
    POP
    LD $CURRENT_T
    BEQ HHLT_PRINT
```

```
    JUMP $T_GETTING_NEW
    HHLT_PRINT: CALL $PRINT
    JUMP $HHLT_DEREF

HHLT_DEREF: JUMP $HHLT

NEW_NUM_TO_PRINT: LD $VIVOD_6
    ADD #0x10
    ST $VIVOD_7
    LD $VIVOD_5
    ADD #0x10
    ST $VIVOD_6
    LD $VIVOD_4
    ADD #0x10
    ST $VIVOD_5
    LD $VIVOD_3
    ADD #0x10
    ST $VIVOD_4
    LD $VIVOD_2
    ADD #0x10
    ST $VIVOD_3
    LD $VIVOD_1
    ADD #0x10
    ST $VIVOD_2
    LD &1
    ST $VIVOD_1
    RET

GET_LAST_NUM:
    CLA
    PUSH
    PUSH
    LD &3
    ST &0
    CYCLE_GET_LAST_NUM:
    SUB #10
    BMI RETURN_GET_LAST_NUM
    ST &0
    LD &1
    INC
    ST &1
    LD &0
    JUMP $CYCLE_GET_LAST_NUM
    RETURN_GET_LAST_NUM: LD &1
    ST &3
    LD &0
    SWAP
    POP
    SWAP
    POP
    RET
```

```
PRINT:
    NOP
    OSTANOV:IN 0x1D
    AND #0x40
    BEQ OSTANOV
    LD $VIVOD_1
    OUT 0x14
    LD $VIVOD_2
    OUT 0x14
    LD $VIVOD_3
    OUT 0x14
    LD $VIVOD_4
    OUT 0x14
    LD $VIVOD_5
    OUT 0x14
    LD $VIVOD_6
    OUT 0x14
    LD $VIVOD_7
    OUT 0x14
    RET
HHLT: HLT
CHECK_IF_DIVIDED: CLA
    PUSH
    LD &2
    CYCLE_CHECK_IF_DIVIDED:
    SUB &3
    BMI CHECK_IF_DIVIDED_BAD
    ST &2
    LD &0
    INC
    ST &0
    LD &2
    BEQ CHECK_IF_DIVIDED_GOOD
    JUMP $CYCLE_CHECK_IF_DIVIDED
    CHECK_IF_DIVIDED_BAD: LD #-1
    SWAP
    POP
    RET
    CHECK_IF_DIVIDED_GOOD: LD &0
    SWAP
    POP
    RET

ORG 0x01FF
T0: WORD 0xB
T1: WORD ?
T2: WORD ?
T3: WORD ?
T4: WORD ?
T5: WORD ?
```

```
SET_T:
    LD START_T
    ADD $CURRENT_T
    ST $DEREFT
    LD $CURRENT_T
    INC
    ST $CURRENT_T
    LD &1
    ST (DEREFT)
    RET

GET_T:
    LD START_T
    ADD $CURRENT_T
    DEC
    ST $DEREFT
    LD $CURRENT_T
    DEC
    ST $CURRENT_T
    LD (DEREFT)
    RET

DEREFT: WORD ?
CURRENT_T: WORD ?
START_T: WORD 0x0200
```

Заключение

Я изучил работу БЭВМ с ВУ, создал программу для вывода данных на ВУ-1.

Список литературы

Методические указания к лабораторным работам по курсу "Основы профессиональной деятельности" [В Интернете] / авт. В. В. Кириллов А. А. Приблуда, С. В. Клименков, Д. Б. Афанасьев. - https://se.ifmo.ru/documents/10180/38002/Методические+указания+к+выполнению+лабораторных+работ+и+рубежного+контроля+БЭВМ+2019+bcomp-ng.pdf/d5a1be02-ad3f-4c43-8032-a2a04d6db12e.