Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Лабораторная работа по ОПД №5
Вариант 6504

Выполнил
Пчелкин Илья Игоервич
P3106

Проверила
Ткешелашвили Н.М.

Санкт-Петербург 2025

Оглавление

Текст задания

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.

1. Программа осуществляет асинхронный вывод данных на ВУ-1
2. Программа начинается с адреса 228_{16}. Размещаемая строка находится по адресу 552_{16}.
3. Строка должна быть представлена в кодировке Windows-1251.
4. Формат представления строки в памяти: АДР1: СИМВ2 СИМВ1 АДР2: СИМВ4 СИМВ3 ... СТОП_СИМВ.
5. Ввод или вывод строки должен быть завершен по символу с кодом 0D (CR). Стоп символ является обычным символом строки и подчиняется тем же правилам расположения в памяти что и другие символы строки.

Адрес	Содержимое	Мнемоника	Описание
223	0552	-	first_symbol_address
224	0000	-	encoded_string
225	000D	-	stop_symbol
226	0552	-	current_symbol
227	00FF	-	mask
228	0200	CLA	0 → AC
229	1203	IN 3	Ожидание готовности ВУ-1
22A	2F40	AND #40	
22B	F0FD	BEQ(IP-3 = 22B)	
22C	AAF9	LD (IP-7 = 226)+	MEM(552) → AC current_symbol : 552 + 1 = 553
22D	EEF6	ST (IP-10 = 224)	AC → MEM(224)
22E	2EF8	AND (IP-8 = 227)	AC & mask → AC
22F	7EF5	CMP (IP-11 = 225)	Проверка на стоп символ
230	F00B	BEQ (IP+11 = 23C)	if Z == 1 then IP + 9 → IP
231	1302	OUT 2	Вывод 1 символа на ВУ-1
232	1203	IN 3	Ожидание готовности ВУ-1
233	2F40	AND #40	
234	F0FD	BEQ(IP-3 = 233)	
235	AEEE	LD (IP-18 = 224)	MEM(224) → AC
236	680	SWAB	AC7 … AC0 <-> AC15 … AC8
237	2EEF	AND (IP-17 = 227)	AC & mask → AC
238	7EEC	CMP (IP-20 = 225)	Проверка на стоп символ
239	F002	BEQ (IP+2 = 23C)	if Z == 1 then IP + 2 → IP
23A	1302	OUT 2	Вывод 2 символа на ВУ-1
23B	CEEC	JUMP (IP-20 = 228)	IP - 20 = 228 → IP
23C	AEE6	LD MEM(IP-26 = 223)	552 → AC
23D	EEE8	ST (IP-24 = 226)	552 → MEM(226)
23E	0100	HLT	Останов

Область представления

Сводная таблица кодов ASCII

ASCII таблица кодов символов Windows (Win-1251)

Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ
0	0	спец. NOP	32	20	спец. SP (Пробел)	64	40	@	96	60	`	128	80	Ђ	160	A0	Ў	192	C0	А	224	E0	а
1	1	спец. SOH	33	21	!	65	41	A	97	61	a	129	81	Ѓ	161	A1	ў	193	C1	Б	225	E1	б
2	2	спец. STX	34	22	"	66	42	B	98	62	b	130	82	‚	162	A2	Ј	194	C2	В	226	E2	в
3	3	спец. ETX	35	23	#	67	43	C	99	63	c	131	83	ѓ	163	A3	Ј	195	C3	Г	227	E3	г
4	4	спец. EOT	36	24	$	68	44	D	100	64	d	132	84	„	164	A4	¤	196	C4	Д	228	E4	д
5	5	спец. ENQ	37	25	%	69	45	E	101	65	e	133	85	…	165	A5	Ґ	197	C5	Е	229	E5	е
6	6	спец. ACK	38	26	&	70	46	F	102	66	f	134	86	†	166	A6	¦	198	C6	Ж	230	E6	ж
7	7	спец. BEL	39	27	'	71	47	G	103	67	g	135	87	‡	167	A7	§	199	C7	З	231	E7	з
8	8	спец. BS	40	28	(	72	48	H	104	68	h	136	88	€	168	A8	Ё	200	C8	И	232	E8	и
9	9	спец. Tab	41	29	)	73	49	I	105	69	i	137	89	‰	169	A9	©	201	C9	Й	233	E9	й
10	0A	спец. LF	42	2A	*	74	4A	J	106	6A	j	138	8A	Љ	170	AA	Є	202	CA	К	234	EA	к
11	0B	спец. VT	43	2B	+	75	4B	K	107	6B	k	139	8B	‹	171	AB	«	203	CB	Л	235	EB	л
12	0C	спец. FF	44	2C	,	76	4C	L	108	6C	l	140	8C	Њ	172	AC	¬	204	CC	М	236	EC	м
13	0D	спец. CR	45	2D	-	77	4D	M	109	6D	m	141	8D	Ќ	173	AD		205	CD	Н	237	ED	н
14	0E	спец. SO	46	2E	.	78	4E	N	110	6E	n	142	8E	Ћ	174	AE	®	206	CE	О	238	EE	о
15	0F	спец. SI	47	2F	/	79	4F	O	111	6F	o	143	8F	Џ	175	AF	Ї	207	CF	П	239	EF	п
16	10	спец. DLE	48	30	0	80	50	P	112	70	p	144	90	ђ	176	B0	°	208	D0	Р	240	F0	р
17	11	спец. DC1	49	31	1	81	51	Q	113	71	q	145	91	‘	177	B1	±	209	D1	С	241	F1	с
18	12	спец. DC2	50	32	2	82	52	R	114	72	r	146	92	’	178	B2	І	210	D2	Т	242	F2	т
19	13	спец. DC3	51	33	3	83	53	S	115	73	s	147	93	“	179	B3	і	211	D3	У	243	F3	у
20	14	спец. DC4	52	34	4	84	54	T	116	74	t	148	94	”	180	B4	ґ	212	D4	Ф	244	F4	ф
21	15	спец. NAK	53	35	5	85	55	U	117	75	u	149	95	•	181	B5	µ	213	D5	Х	245	F5	х
22	16	спец. SYN	54	36	6	86	56	V	118	76	v	150	96	–	182	B6	¶	214	D6	Ц	246	F6	ц
23	17	спец. ETB	55	37	7	87	57	W	119	77	w	151	97	—	183	B7	·	215	D7	Ч	247	F7	ч
24	18	спец. CAN	56	38	8	88	58	X	120	78	x	152	98	[illegible]	184	B8	ё	216	D8	Ш	248	F8	ш
25	19	спец. EM	57	39	9	89	59	Y	121	79	y	153	99	™	185	B9	№	217	D9	Щ	249	F9	щ
26	1A	спец. SUB	58	3A	:	90	5A	Z	122	7A	z	154	9A	љ	186	BA	є	218	DA	Ъ	250	FA	ъ
27	1B	спец. ESC	59	3B	;	91	5B	[	123	7B	{	155	9B	›	187	BB	»	219	DB	Ы	251	FB	ы
28	1C	спец. FS	60	3C	<	92	5C	\	124	7C	|	156	9C	њ	188	BC	ј	220	DC	Ь	252	FC	ь
29	1D	спец. GS	61	3D	=	93	5D	]	125	7D	}	157	9D	ќ	189	BD	S	221	DD	Э	253	FD	э
30	1E	спец. RS	62	3E	>	94	5E	^	126	7E	~	158	9E	ћ	190	BE	s	222	DE	Ю	254	FE	ю
31	1F	спец. US	63	3F	?	95	5F	_	127	7F		159	9F	џ	191	BF	ї	223	DF	Я	255	FF	я

first_symbol_address – 11-разрядный адрес первой ячейки с символами
current_symbol – 11-разрядный адрес текущей ячейки с символами
encoded_string ∈ [21_{16}; FF_{16}] – 16-разрядная ячейка для хранения текущих двух символов;
Старший байт – код второго символа, младший байт – код первого символа.
stop_symbol – 8-разрядный стоп символ
mask = FF_{16} – маска (нужна в ходе программы для сравнения текущего символа со стоп символом)
n – длина слова/2
Максимальная длина передаваемого слова = 2946
2*(7FF – 23E) = 2*5C1 = $B82_{16}$ = 2946_{10}

ОДЗ

$$\begin{cases} \text{first\_symbol\_address} \in [0; 223_{16} - n] \\ n \in [1; 547] \end{cases}$$
$$\text{current\_symbol} \in [0; 223_{16}]$$
$$n \in [1; 547]$$

[

$$\begin{cases} \text{first\_symbol\_address} \in [23F; 7FF - n] \\ n \in [1; 1473] \end{cases}$$
$$\text{current\_symbol} \in [23F; 7FF]$$
$$n \in [1; 1473]$$

старший байт encoded_string ∈ [21_{16}; FF_{16}]
младший байт encoded_string ∈ [21_{16}; FF_{16}]
mask = FF_{16}

Передаваемое слово:
ЯСЕНЬ

Win-1251: DFD1 CDC5 0DDC

UTF-16LE: FFFE 2F04 2104 1504 1D04 2C04 000D

UTF-8: A1AF 9D95 0DAC

Код программы на ассемблере БЭВМ

```
ORG 0x223
word_beginning: WORD 0x552
encoded_string: WORD ?
stop_symbol:    WORD 0x000D
current_symbol: WORD 0x552
mask:           WORD 0x00FF


start:          CLA

symbol_1:       IN 3
                AND #0x40
                BEQ symbol_1
                LD (current_symbol)+
                ST encoded_string
                AND mask
                OUT 2
                CMP stop_symbol
                BEQ end_program

symbol_2:       IN 3
                AND #0x40
                BEQ symbol_2
                LD encoded_string
                SWAB
                AND mask
                OUT 2
                CMP stop_symbol
                BEQ end_program
                JUMP start

end_program:    LD word_beginning
                ST current_symbol
                HLT


ORG 0x552
WORD 0xD1DF ; С, Я
ORG 0x553
WORD 0xCDC5 ; Н, Е
ORG 0x554
WORD 0x0DDC ; СТОП_СИМВ, Ь
```

Трассировка для первых двух символов `С, Я: D1DF`

Адр	Знчн	IP	CR	AR	DR	SP	BR	AC	PS	NZVC	Адр	Знчн
228	0200	229	0200	228	0200	000	0228	0000	004	0100		
229	1203	22A	1203	229	1203	000	0229	0040	004	0100		
229	1203	22A	1203	229	1203	000	0229	0040	004	0100		
22A	2F40	22B	2F40	22A	0040	000	0040	0040	004	0000		
22B	F0FD	22C	F0FD	22B	F0FD	000	022B	0040	004	0000		
22C	AAF9	22D	AAF9	552	D1DF	000	FFF9	D1DF	008	1000	226	0553
22D	EEF6	22E	EEF6	224	D1DF	000	FFF6	D1DF	008	1000	224	D1DF
22E	2EF8	22F	2EF8	227	00FF	000	FFF8	00DF	000	0000		
22F	1302	230	1302	22F	1302	000	022F	00DF	000	0000		

Дополнительное задание

Калькулятор. Умножение знаковых чисел. Ввод пары чисел с ВУ-9 (цифровая клавиатура), разделитель чисел - операция умножения. По нажатию "=" вывод результата на ВУ-7 (семисегментный индикатор).

Код программы на ассемблере БЭВМ

```
a:                word 0x0 ; первый множитель
b:                word 0x0 ; второй множитель
res:              word 0x0 ; резульат умножения

sign:             word 0x0 ; знак результата
minus:            word 0xA ; код символа "-"
equate:           word 0xF ; код символа "="
multiplication:   word 0xD ; код символа "*"


START:
    cla
    ; очистка значений
    ld sign
    cla
    st sign

; цикл ввода a
input_a:
    in 0x1C
    cmp minus
    beq handle_minus_a
    cmp #0x0
    blt input_a   ; если in < 0 то не цифра
    cmp #0xB
    bge input_a   ; если in >= B то не цифра и не минус
    st a          ; сохраняем первый символ - a
    jump multiply_input

; изменение знака
handle_minus_a:
    ld sign
    cmp #0x1
    beq set_zero_a
    ld #0x1         ; установка "-"
    jump save_sign_a
set_zero_a:         ; установка "+"
    ld #0x0
save_sign_a:        ; сохранение знака переход к вводу a
    st sign
    jump input_a


; цикл ввода "*"
multiply_input:
    in 0x1C    ; ввод с клавиатуры
    cmp multiplication
    bne input_a


; цикл ввода b
input_b:
```

```
    in 0x1C
    cmp minus
    beq handle_minus_b ;
    cmp #0x0
    blt input_b   ; если in < 0 то не цифра
    cmp #0xB
    bge input_b   ; если in >= B то не цифра и не минус
    st b          ; сохраняем первый символ - a
    jump equate_input

; изменение знака
handle_minus_b:
    ld sign
    cmp #0x1
    beq set_zero_b
    ld #0x1           ; установка "-"
    jump save_sign_b

set_zero_b:           ; установка "+"
    ld #0x0
save_sign_b:          ; сохранение знака переход к вводу b
    st sign
    jump input_b



; цикл ввода "="
equate_input:
    in 0x1C
    cmp equate
    bne input_b


; умножение
multiply:
    ld a
    push
    ld b
    push
    ld #0x0             ; локальная переменная для накопления результата умножения
    push
    call $multiply_func
    pop
    pop
    pop
    st res

; вывод результата
FINISH:
    ld res
    push
    ld #0x0         ; локальная переменная счетчика десятков
    push
    call $hex_to_bcd
    pop
    pop
    st res


    and #0x00f0               ; маска для сравнения 2 тетрады
    cmp #0x0010
```

```
    bge two_digits            ; если во второй тетраде число >= 1, значит двузначное
число
    blt one_digit             ; если во второй тетраде число < 1, значит цифра


; вывод цифры
one_digit:
    ld #0x002B
    out 0x14    ; сброс 2 разряда индикатора
    ld res
    cmp #0x0
    beq zero_out

    ld sign
    cmp #0x1
    beq negative_res_one_digit  ; если "-" то переходим к выводу цифры с минусом

    ld #0x001B
    out 0x14    ; сброс 1 разряда индикатора
    ld res
    out 0x14                    ; иначе выводим цифру на позицию 0 и завершаем
программу
    jump START

; вывод цифры с минусом
negative_res_one_digit:
    ld #0x001A
    out 0x14          ; выводим "-" на позицию 1
    ld res
    out 0x14          ; выводим цифру на позицию 0
    jump START

zero_out:
    ld #0x001B  ;
    out 0x14    ; сброс 1 разряда индикатора
    ld res
    out 0x14
    jump START

; вывод двузначного числа
two_digits:
    ld sign
    cmp #0x1
    beq negative_res_two_digits  ; если "-" то переходим к выводу числа с минусом


    ld #0x002B
    out 0x14    ; сброс 2 разряда индикатора

    ld res
    asr
    asr
    asr
    asr             ; сдвигаем на 4 бита вправо => кол-во десятков t в первой тетраде
=> AC = 0x000t
    or #0x0010      ; объединяем t и 0x0010 для вывода на 1 позицию индикатора => AC
= 0x001t
    out 0x14
    ld res
    and #0x000f     ; оставляем только 1 тетраду res (кол-во единиц) и выводим на 0
позицию индикатора
```

```
    out 0x14
    jump START



; вывод двузначного числа с минусом
negative_res_two_digits:
    ld #0x002A
    out 0x14          ; выводим "-" на позицию 2
    ld res
    asr
    asr
    asr
    asr
    or #0x0010
    out 0x14
    ld res
    and #0x000f
    out 0x14
    jump START



org 0x200
multiply_func:
    ld &3               ; ld a
    add &1              ; add temp
    st &1               ; st temp
    loop &2             ; b - 1
    jump multiply_func
    ld &1
    st &3               ; результат умножения сохранить в 7FF
    ret



org 0x250
; первод числа из 16 СС в 2-10 СС
hex_to_bcd:
    ld &2             ; ldres
    cmp #0x000A
    blt done          ; если res < 10 то завершаем подсчёт десятков
    sub #0x000A
    st &2
    ld &1
    inc
    st &1  ; иначе вычитаем из него 10 и увеличиваем счетчик десятков на 1
    jump hex_to_bcd       ; переходим к следующей итерации

done:
    ld &1              ; количество десятков числа E[0x0000, 0x0009]
    asl
    asl
    asl
    asl               ; сдвигаем количество десятков из 1 тетрады во 2
    or &2             ; объединяем с количеством единиц
    st &2
    ret
```