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»

Вариант 1351

Выполнил:

Студент группы P3113

Иванов Евгений Дмитриевич

Преподаватель:

Ткешелашвили Нино Мерабиевна

г. Санкт-Петербург

2020

Условие варианта(вариант 1351):

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 1351

1. Программа осуществляет асинхронный вывод данных на ВУ-3
2. Программа начинается с адреса $44A_{16}$. Размещаемая строка находится по адресу $5C6_{16}$.
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.

Программа:

Адрес	Значение	Мнемоника	Комментарий
200	1207	IN 0x07	Бесконечный цикл, ожидающий готовность ву-3(по флагу), если флаг установлен, то IP меняется к 203 и происходит последующий вывод символа
201	2F40	AND #0x40	
202	F0FD	BEQ 200	
203	AC01	LD (SP+1)	Загрузка символа для вывода
204	0600	SXTB	Расширения знака символа(для сравнения с стоп-символом в основной программе)
205	1306	OUT 0x06	Вывод символа на РДВУ-3, сброс флага
206	EC01	ST (SP+1)	Сохранение в стек только символа(с расширенным знаком)
207	0A00	RET	Выход из подпрограммы

Адрес	Значение	Мнемоника	Комментарий
447	0200	F_inp	Адрес подпрограммы для вывода на ВУ-3
448	05C6	FIRST	Выбранный сейчас символ/по совместительству первый адрес строки
449	0000	ENDLN	Код окончания строки(NULL)
44A	+ 0200	CLA	Очистка аккумулятора 0->AC
44B	A8FC	LD (FIRST)	Загрузили символ. (FIRST)->AC
44C	0680	SWAB	Перевернули байты, чтобы работать с первым символом.
44D	0C00	PUSH	Положили в стек симв2симв1 у данного адреса, чтобы подпрограмма его вывела.
44E	D8F8	CALL (F_inp)	Вызов подпрограммы вывода младшего байта(из переданного раннее числа)
44F	0800	POP	Очистка стека/принятие символа с расширенным знаком для последующей проверки с концом строки

450	7EF8	CMP ENDLN	Установка знаков по сравнению конца строки и данного символа
451	F007	BEQ 459	Если символ оказался конечным, то конец программы
452	AAF5	LD (FIRST)+	Взяли следующий символ(символ 2, который и так лежит в младших значениях) Также сместили адрес до следующего.
453	0C00	PUSH	Положили в стек симв1симв2 у данного адреса, чтобы подпрограмма его вывела.
454	D8F2	CALL (F_inp)	Вызов подпрограммы вывода младшего байта(из переданного раннее числа)
455	0800	POP	Очистка стека/принятие символа с расширенным знаком для последующей проверки с концом строки
456	7EF2	CMP ENDLN	Установка знаков по сравнению конца строки и данного символа
457	F001	BEQ 459	Если символ оказался конечным, то конец программы
458	CEF2	JUMP 44B	Переход к обработке следующего адреса данных.
459	0100	HLT	Выход из программы.

Код на ассемблере(с минимальными пояснениями)**:**

```
            ORG   0x200          => Подпрограмма выборки символа
INPUT:      IN    0x07           взяли флаг ВУ
            AND   #0x40          проверили его на готовность
            BEQ   INPUT          в случае неготовности ожидание готовности
            LD    (SP+1)         загрузили символ
            SXTB                 расширили символ, чтобы при возврате в main легко
.                                сравнить его с нуль-символом
            OUT   0x06
            ST    (SP+1)
            RET

            ORG   0x447
F_INP:      WORD      $INPUT
FIRST:      WORD      0x5C6
ENDLN:      WORD      0x00

START:      CLA

FOREACH:    LD    (FIRST)
            SWAB
            PUSH
            CALL  (F_INP)
            POP
            CMP   ENDLN
```

```
            BEQ   EXIT

            LD    (FIRST)+
            PUSH
            CALL  (F_INP)
            POP
            CMP   ENDLN
            BEQ   EXIT

            JUMP FOREACH
EXIT:       HLT                         выход

            ORG         0x5C6
            WORD        0xB2B5          “B” & ”E”
            WORD        0xC1BD          “C” & ”H”
            WORD        0xB000          “A” & NULL
```

Информация о программе:

- Программа находится в ячейках с адресами 448-45A. Из них ячейки 448, 449 являются вспомогательными, содержат ссылку на данный символ, а также значение стоп-символа.
- Выводимая строка начинается с ячейки с адресом 5C6.

Выданные преподавателем буквы:

- ВЕСНА
- В кодировке это B2, B5, C1, BD, B0 и конец строки 00

Код и адрес команды		Регистры БЭВМ после выполнения команды								Изменения в памяти	
Адрес	Значение	IP	CR	AR	DR	SP	BR	AC	**NZVC**	Адрес	Новый код
44A	0200	44B	0200	44A	0200	000	44A	0000	0100		
Обработка первого адреса.											
44B	A8FC	44C	A8FC	5C6	B2B5	000	FFFC	B2B5	1000		
44C	0680	44D	0680	44C	0680	000	066C	B5B2	1000		
44D	0C00	44E	0C00	7FF	B5B2	7FF	44D	B5B2	1000	7FF	B5B2
44E	D8F8	200	D8F8	7FE	044F	7FE	0200	B5B2	1000	7FE	044F
Вывод первого символа											
200	1207	201	1207	200	1207	7FE	0200	B540	1000		
201	2F40	202	2F40	201	0040	7FE	0040	0040	0000		
202	F0FD	203	F0FD	202	F0FD	7FE	0202	0040	0000		
203	AC01	204	AC01	7FF	B5B2	7FE	0001	B5B2	1000		
204	0600	205	0600	204	0600	7FE	0204	FFB2	1000		
205	1306	206	1306	205	1306	7FE	0205	FFB2	1000		

206	EC01	207	EC01	7FF	FFB2	7FE	0001	FFB2	1000	7FF	FFB2
207	0A00	44F	0A00	7FE	044F	7FF	0207	FFB2	1000		
Сравнение с концом строки и переход к обработке второго символа по этому адресу											
44F	0800	450	0800	7FF	FFB2	000	044F	FFB2	1000		
450	7EF8	451	7EF8	449	0000	000	FFF8	FFB2	1001		
451	F007	452	F007	451	F007	000	0451	FFB2	1001		
452	AAF5	453	AAF5	5C6	B2B5	000	FFF5	B2B5	1001	448	05C7
453	0C00	454	0C00	7FF	B2B5	7FF	0453	B2B5	1001	7FF	B2B5
454	D8F2	200	D8F2	7FE	0455	7FE	0200	B2B5	1001	7FE	0455
Вывод второго символа											
200	1207	201	1207	200	1207	7FE	0200	B240	1001		
201	2F40	202	2F40	201	0040	7FE	0040	0040	0001		
202	F0FD	203	F0FD	202	F0FD	7FE	0202	0040	0001		
203	AC01	204	AC01	7FF	B2B5	7FE	0001	B2B5	1001		
204	0600	205	0600	204	0600	7FE	0204	FFB5	1001		
205	1306	206	1306	205	1306	7FE	0205	FFB5	1001		
206	EC01	207	EC01	7FF	FFB5	7FE	0001	FFB5	1001	7FF	FFB5
207	0A00	455	0A00	7FE	0455	7FF	0207	FFB5	1001		
Переход к дальнейшей обработке(проверка на конец строки)											
455	0800	456	0800	7FF	FFB5	000	0455	FFB5	1001		
456	7EF2	457	7EF2	449	0000	000	FFF2	FFB5	1001		
457	F001	458	F001	457	F001	000	0457	FFB5	1001		
458	CEF2	44B	CEF2	458	044B	000	FFF2	FFB5	1001		
Дальше происходит переход к обработке следующего адреса											
44B	...										

Дополнительное задание:

Задание:

- Чтение символов с клавиатуры (ВУ-8), после ввода символа "точка" чтение прекращается, и введённая строка выводится на ВУ-6 (бегущая строка).

Реализация задания:

- Программа разбита на блоки, сначала блок ввода с ВУ-8 данных в память в ячейки с адресом 600+. В каждой ячейке хранится два символа SYMBOL N, SYMBOL N+1, так как ячейки 16 разрядные, а буквы 8ми разрядные.
- Следующий блок это СИНХРОННЫЙ вывод на ВУ-6 при вводе СТОП-символа, который так же, как и любой другой символ, записывается в память, а потом выводится(это символ точка)

- При этом для вывода каждого символа определенно нужна подпрограмма, а для этого нужен алгоритм поиска подпрограммы для нужного символа, я решил эту проблему тем, что в ячейке с НОМЕРОМ, равным коду символа, лежит АДРЕС первой ячейки отрисовки этого символа, таким образом, с основной программы при получении символа надо просто вызвать функцию от адреса, лежащего по адресу значения символа. По этой причине первые 256 адресов заняты и не могут быть использованы, ведь в них(не во всех, а только для русских символов) лежат данные о положении в памяти подпрограмм для отрисовки каждого из них.
- Сами подпрограммы отрисовки лежат сразу за данными и имеют соответствующие метки(на языке ассемблера), то есть с адреса 100
- По адресу, начиная с 400 лежит подпрограмма получения данных с ВУ-8(клавиатуры).
- Начиная с ячейки с адресом 500 лежит основная программа, получающая значения с ВУ-8 и выводящая их отрисовку на ВУ-6 соответственно, данные при этом записываются в ячейки, с адресом 600+, следующим правилом

  Строка должна быть представлена в кодировке ISO-8859-5.
  Формат представления строки в памяти: АДР1: СИМВ1 СИМВ2 АДР2: СИМВ3 СИМВ4 ... СТОП_СИМВ.
  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.
- В данном случае стоп символ это точка.
- В отрисовке каждого символа я делаю пробел(пустой байт) вначале, это не эффективно по памяти, было бы логичнее добавить вывод “пробела” перед отрисовкой, однако я посчитал, что тогда в основной программе будет труднее разобраться, также это не всегда выгодно, так как может быть символ, который не должен быть отрисован с пробелом(по типу апострофа в английском языке), поэтому я выбрал такой способ реализации.
- Программа не является реентерабельной.

Пример работы: по понятным причинам нам интересно лишь ВУ-6 :-)

Код программы на ассемблере:

	ORG	0xA1
	WORD	$symbEo
	ORG	0xB0
	WORD	$symbA
	WORD	$symbB
	WORD	$symbV
	WORD	$symbG
	WORD	$symbD
	WORD	$symbE
	WORD	$symbJ
	WORD	$symbZ
	WORD	$symbI
	WORD	$symbIo
	WORD	$symbK
	WORD	$symbL

	WORD	$symbM
	WORD	$symbN
	WORD	$symbO
	WORD	$symbP
	WORD	$symbR
	WORD	$symbS
	WORD	$symbT
	WORD	$symbU
	WORD	$symbF
	WORD	$symbH
	WORD	$symbC
	WORD	$symbCh
	WORD	$symbHa
	WORD	$symbCha
	WORD	$symbTT
	WORD	$symbnoI
	WORD	$symbTM
	WORD	$symbAa
	WORD	$symbyou
	WORD	$symbYa
	ORG	0x2E
	WORD	$symbDT
	ORG	0x100
symbA:	LD	#0x00
	OUT	0x10
	LD	#0x7
	OUT	0x10
	LD	#0x1C
	OUT	0x10
	LD	#0xE8
	OUT	0x10
	LD	#0x88
	OUT	0x10
	LD	#0xE8
	OUT	0x10
	LD	#0x1C
	OUT	0x10
	LD	#0x7
	OUT	0x10
	RET	
symbB:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10

	LD	#0x89
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0x8F
	OUT	0x10
	RET	
symbV:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x99
	OUT	0x10
	OUT	0x10
	LD	#0x69
	OUT	0x10
	LD	#0x06
	OUT	0x10
	RET	
symbG:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x80
	OUT	0x10
	OUT	0x10
	OUT	0x10
	OUT	0x10
	RET	
symbD:	LD	#0x00
	OUT	0x10
	LD	#0x3
	OUT	0x10
	LD	#0xFE
	OUT	0x10
	LD	#0x42
	OUT	0x10
	LD	#0x3C
	OUT	0x10
	LD	#0x2
	OUT	0x10
	LD	#0x3
	OUT	0x10
	RET	
symbE:	LD	#0x00

	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x99
	OUT	0x10
	OUT	0x10
	LD	#0x81
	OUT	0x10
	OUT	0x10
	RET	
symbEo: LD	#0x00	
	OUT	0x10
	LD	#0x3F
	OUT	0x10
	LD	#0xAD
	OUT	0x10
	LD	#0x2D
	OUT	0x10
	LD	#0xAD
	OUT	0x10
	LD	#0x2D
	OUT	0x10
	RET	
symbJ:	LD	#0x00
	OUT	0x10
	LD	#0xC3
	OUT	0x10
	LD	#0x24
	OUT	0x10
	LD	#0x18
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x18
	OUT	0x10
	LD	#0x24
	OUT	0x10
	LD	#0xC3
	OUT	0x10
	RET	
symbZ:	LD	#0x00
	OUT	0x10
	LD	#0x81
	OUT	0x10
	OUT	0x10

	LD	#0x91
	OUT	0x10
	LD	#0xAA
	OUT	0x10
	LD	#0xCC
	OUT	0x10
	RET	
symbI:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x02
	OUT	0x10
	LD	#0x0C
	OUT	0x10
	LD	#0x18
	OUT	0x10
	LD	#0x30
	OUT	0x10
	LD	#0x40
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbIo:	LD	#0x00
	OUT	0x10
	LD	#0x1F
	OUT	0x10
	LD	#0x82
	OUT	0x10
	LD	#0x44
	OUT	0x10
	LD	#0x88
	OUT	0x10
	LD	#0x1F
	OUT	0x10
	RET	
symbK:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x18
	OUT	0x10
	LD	#0x24
	OUT	0x10

	LD	#0x42
	OUT	0x10
	LD	#0x81
	OUT	0x10
	RET	
symbL:	LD	#0x00
	OUT	0x10
	LD	#0x03
	OUT	0x10
	LD	#0x0C
	OUT	0x10
	LD	#0x30
	OUT	0x10
	LD	#0xC0
	OUT	0x10
	LD	#0x30
	OUT	0x10
	LD	#0x0C
	OUT	0x10
	LD	#0x03
	OUT	0x10
	RET	
symbM:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x60
	OUT	0x10
	LD	#0x30
	OUT	0x10
	LD	#0x60
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbN:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x18
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0xFF
	OUT	0x10

	RET	
symbO:	LD	#0x00
	OUT	0x10
	LD	#0x7E
	OUT	0x10
	LD	#0x81
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0x7E
	OUT	0x10
	RET	
symbP:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x80
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbR:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x90
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0x60
	OUT	0x10
	RET	
symbS:	LD	#0x00
	OUT	0x10
	LD	#0x7E
	OUT	0x10
	LD	#0x81
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0x66
	OUT	0x10
	RET	

symbT:	LD	#0x00
	OUT	0x10
	LD	#0x80
	OUT	0x10
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x80
	OUT	0x10
	OUT	0x10
	RET	
symbU:	LD	#0x00
	OUT	0x10
	LD	#0x81
	OUT	0x10
	LD	#0x42
	OUT	0x10
	LD	#0x24
	OUT	0x10
	LD	#0x2C
	OUT	0x10
	LD	#0xF0
	OUT	0x10
	RET	
symbF:	LD	#0x00
	OUT	0x10
	LD	#0xF0
	OUT	0x10
	LD	#0x90
	OUT	0x10
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x90
	OUT	0x10
	OUT	0x10
	LD	#0xF0
	OUT	0x10
	RET	
symbH:	LD	#0x00
	OUT	0x10
	LD	#0xC3
	OUT	0x10
	LD	#0x3C
	OUT	0x10

	LD	#0x3C
	OUT	0x10
	LD	#0xC3
	OUT	0x10
	RET	
symbC:	LD	#0x00
	OUT	0x10
	LD	#0xFE
	OUT	0x10
	LD	#0x02
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0xC3
	OUT	0x10
	LD	#0x03
	OUT	0x10
	RET	
symbCh:	LD	#0x00
	OUT	0x10
	LD	#0xF0
	OUT	0x10
	LD	#0x10
	OUT	0x10
	OUT	0x10
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbHa:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x01
	OUT	0x10
	OUT	0x10
	LD	#0x7F
	OUT	0x10
	LD	#0x01
	OUT	0x10
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbCha:LD	#0x00	

	OUT	0x10
	LD	#0xFE
	OUT	0x10
	LD	#0x02
	OUT	0x10
	LD	#0x7E
	OUT	0x10
	LD	#0x02
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x03
	OUT	0x10
	RET	
symbTT:	LD	#0x00
	OUT	0x10
	LD	#0xC0
	OUT	0x10
	LD	#0x80
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x09
	OUT	0x10
	OUT	0x10
	LD	#0x06
	OUT	0x10
	RET	
symbnoI:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x09
	OUT	0x10
	OUT	0x10
	LD	#0x06
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbTM:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x09

	OUT	0x10
	OUT	0x10
	LD	#0x06
	OUT	0x10
	RET	
symbAa:	LD	#0x00
	OUT	0x10
	LD	#0x81
	OUT	0x10
	LD	#0x91
	OUT	0x10
	OUT	0x10
	LD	#0x52
	OUT	0x10
	LD	#0x3C
	OUT	0x10
	RET	
symbyou:	LD	#0x00
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	LD	#0x18
	OUT	0x10
	LD	#0x7E
	OUT	0x10
	LD	#0x81
	OUT	0x10
	OUT	0x10
	LD	#0x7E
	OUT	0x10
	RET	
symbYa:	LD	#0x00
	OUT	0x10
	LD	#0x61
	OUT	0x10
	LD	#0x92
	OUT	0x10
	LD	#0x94
	OUT	0x10
	LD	#0xFF
	OUT	0x10
	RET	
symbDT:	LD	#0x00
	OUT	0x10
	LD	#0x03

	OUT	0x10
	OUT	0x10
	RET	
	ORG	0x400
INPUT:	IN	0x19
	AND	#0x40
	BEQ	INPUT
	IN	0x18
	ST	(SP+1)
	RET	
	ORG	0x500
inp:	WORD	$INPUT
STOP:	WORD	0x2E
ARRAY:	WORD	0x600
START:	CLA	
inVU9:	PUSH	
	CALL	(inp)
	LD	(SP+0)
	SWAB	
	ST	(ARRAY)
	POP	
	CMP	STOP
	BEQ	outVU6
	PUSH	
	CALL	(inp)
	LD	(SP+0)
	ADD	(ARRAY)
	ST	(ARRAY)+
	POP	
	CMP	STOP
	BEQ	outVU6
	JUMP	inVU9
ARGS:	WORD	0x600
outVU6:	LD	(ARGS)
	SWAB	
	AND	NULLF
	ST	NOWWORD

	ST	NOWSYMB
	LD	(NOWSYMB)
	ST	NOWSYMB
	CALL	(NOWSYMB)
	LD	NOWWORD
	CMP	STOP
	BEQ	EXIT
	LD	(ARGS)+
	AND	NULLF
	ST	NOWWORD
	ST	NOWSYMB
	LD	(NOWSYMB)
	ST	NOWSYMB
	CALL	(NOWSYMB)
	LD	NOWWORD
	CMP	STOP
	BEQ	EXIT
	JUMP	outVU6
EXIT:	HLT	
NOWWORD:	WORD	0x00
NULLF:	WORD	0xFF
NOWSYMB:	WORD	0x00

Вывод: я научился работать с контроллерами ВУ, обращаться к разным регистрам устройств и обрабатывать информацию с них/выводить на них. Также я ознакомился с ассемблером, что упрощает *жизнь, особенно после выполненного допа, где около 500 строк на ассемблере.*