

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

Исследование взаимодействия с портами ввода-вывода

Отчет по лабораторной работе №6

по дисциплине «Системное программирование»

студента 3 курса группы ПИ-б-о-231(2)

Аметов Кемран Ленверович

Направления подготовки 09.03.04 «Программная инженерия»

Цель:

1. Практически закрепить знание теоретических сведений о возможностях взаимодействия МП с портами ввода-вывода;
2. Привить практические навыки написания сложных программ и программирования различных операций в машинных кодах для микропроцессора.

Ход работы

Вариант 11

Задание 1:

$$11) F = \frac{A+3}{B+3} - \frac{D+3}{A+3}$$

Код

```
C:\Users\user\Desktop\Telegram Desktop> g++ 11.cpp
1  ; Ввод значений A, B, D
2  mvi c,1
3  mvi d,1
4  mvi b,1
5
6  ; A+3
7  mov a,c
8  adi 3
9  mov c,a
10 ; D+3
11 mov a,d
12 adi 3
13 mov d,a
14 ; B+3
15 mov a,b
16 adi 3
17 mov b,a
18
19 ; (A+3)/(B+3)
20 mov h,c
21 mov e,b
22 call DIV_SIGNED
23 mov b,1
24
25 ; (D+3)/(A+3)
26 mov h,d
27 mov e,c
28 call DIV_SIGNED
29 mov d,1
30
31 ; (A+3)/(B+3) - (D+3)/(A+3)
32 mov a,b
33 sub d
34
35 hlt
36
37 div:
38 ; Сохраняю итоговый знак в первый бит 1
39 mov a, e
40 ora h
41 ani 80h
42 mov l, a
43
44 ; Получаю модули
45 mov a, e
46 ani 80h
47 mov a, e
48 jz skip_1
49 xri fffh
50 inr a
51 mov e, a
52 skip_1:
53
54 mov a, h
55 ani 80h
56 mov a, h
57 jz skip_2
58 xri fffh
59 inr a
60 skip_2:
61 mov a, a
```

C: > Users > USER > Downloads >

```
58    xri ffh
59    inr a
60    mov h, a
61    skip_2:
62
63    ; Основной цикл
64    mov a, e
65    loop:
66    sub h
67    jc over
68    inr l
69    jnc loop
70
71    over:
72    mov a, l
73    ani 80h
74    mov a, l
75    jz skip_3
76    ani 7fh
77    dcr a
78    xri ffh
79    mov l, a
80    skip_3:ret
81
```

Проверка

Run Max steps: 10000000 Step prev Step next

Code sample: Set

SUCCESS

Code

```
; Ввод значений A, B, D
movi c,1
movi d,1
movi b,1

; A+3
mov a,c
adi 3
mov c,a
; D+3
mov a,d
adi 3
mov d,a
; B+3
mov a,b
adi 3
mov b,a

; (A+3)/(B+3)
mov h,c
mov e,b
call div
mov b,1

; (D+3)/(A+3)
mov h,d
mov e,c
call div
mov d,1

; (A+3)/(B+3) - (D+3)/(A+3)
mov a,b
sub d
hlt
```

CPU State

a: 00 b: 01 c: 04 d: 01 e: 04
h: 04 l: 01 pc: 0021 sp: 0000 halted: 1

Flags

Sign: 00 Zero: 01 Parity: 01 Carry: 00

RAM

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0E	01	16	01	06	01	79	C6	03	4F	7A	C6	03	57	78	C6
001	03	47	61	58	CD	21	00	45	62	59	CD	21	00	55	78	92
002	76	7B	B4	E6	80	6F	7B	E6	80	7B	CA	31	00	EE	FF	3C
003	5F	7C	E6	80	7C	CA	3C	00	EE	FF	3C	67	7B	94	DA	45
004	00	2C	D2	3D	00	7D	E6	80	7D	CA	52	00	E6	7F	3D	EE
005	FF	6F	C9	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

From address (hex): 0000 Show

Labels

div: 0021
skip_1: 0031
skip_2: 003C
loop: 003D
over: 0045
skip_3: 0052

Задание 2:

11. Количество элементов не больше данного

```
C:\> Users\USER\Downloads\Telegram Desktop> 2_task
1 ; Место начала массива
2 LXI H, ARRAY
3 ; В регистре В буду хранить размер массива
4 mvi b, 7
5 ; В регистре С буду хранить число с которым будет проходить сравнение
6 mvi c, 25h
7
8 mov d, b
9 mov a, c
10
11
12 LOOP:
13 mov e, m
14 cmp M
15 jnc LABEL
16 DCR d ; Буду производить декремент от общего кол-во элементов, каждый раз когда буду находить число большее данного
17 LABEL:
18 inx h
19 dcr b
20 jnz LOOP
21
22 hit
23 ARRAY:
24 DB 05H
25 DB 0AH
26 DB 1CH
27 DB FFH
28 DB 80H
29 DB 33H
30 DB 77H
```

Проверка:

SUCCESS

Code

```
; Место начала массива
LXI H, ARRAY
; В регистре В буду хранить размер массива
mvi b, 7
; В регистре С буду хранить число с которым будет проходить
сравнение
mvi c, 25h

mov d, b
mov a, c

LOOP:
mov e, m
cmp M
jnc LABEL
DCR d ; Буду производить декремент от общего кол-во элементов,
каждый раз когда буду находить число большее данного
LABEL:
inx h
dcr b
jnz LOOP

hit
ARRAY:
DB 05H
DB 0AH
DB 1CH
DB FFH
DB 80H
DB 33H
DB 77H
```

CPU State

a: 25b: 00c: 25d: 03e: 77

h: 00l: 1Cpc: 0015sp: 0000halted: 1

Flags

Sign: 00Zero: 01Parity: 01Carry: 01

RAM

Hex

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	21	15	00	06	07	0E	25	50	79	5E	BE	D2	0F	00	15	23
001	05	C2	09	00	76	05	0A	1C	FF	80	33	77	00	00	00	00
002	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

From address (hex): 0000Show

Labels

LOOP:0009

LABEL:000F

ARRAY:0015

Задание 3:

$$11. \quad F = (a1-b1) * (a2-b2) * (a3-b3) * \dots$$

C:\> Users\USER\Downloads\Telegram Desktop\popitki.ini

```
1  mvi b, 05h
2  mvi c, 00h
3  lxi h, ARRAY1
4  lxi d, ARRAY2
5  lxi sp, 1000h
6
7  ; Оформил в цикл
8  LOOP1:
9      push b
10     mvi c, 0h
11
12     mov a, m; A = a1
13     mov b, a
14
15     ani 80h ; Достаяю знак элемента массива
16     ora c
17     mov c, a
18
19     xchg
20     mov a, m
21     ani 80h ; Достаяю знак второго элемента массива
22     rrc ; Т.к есть жесткое ограничение по использованию памяти использую первые 3 старших бита регистра C
23     ora c ; Как знак первого числа, знак второго числа, знак результата из этого я пойму есть ли переполнение или нет
24     mov c, a
25
26     mov a, b
27     sub m
28     mov m, a ; Вычисляю сам результат и помещаю его обратно в ОЗУ
29
30     ani 80h ; Достаяю знак результата
31     rrc
32     rrc
33     ora c
34     mov c, a
35
36     xri 80h ; Переполнение происходит в двух случаях, когда при сложении оба числа отрицательные, а ответ положительный и наоборот
37     jz SKIP ; То есть 110 и 001 это случаи переполнения, так как тут вычитание второй бит инвертируется
38     ; То есть 100 и 011, то есть если регистр C равен 10000000(80h) или 01100000(60h) то произошло переполнение
39
40     mov c, a
41     xri 60h
42     jz SKIP
43
44     jmp next_loop ; если переполнения нет, то просто переходим на следующий цикл
45
46     SKIP: ; если переполнение есть то увеличиваем счетчик переполнений
47     pop b
48     inc c
49     jmp skip_2
50
51     next_loop:
52     pop b
53     skip_2:
54     xchg
55     inc h
56     inc d
57     dec b
58     JNZ LOOP1
59
60     ; End of program
```

```

59
60 lxi h, ARRAY2
61 mvi b, 04
62 mov a, m
63 inx h
64
65
66 LOOP2:
67     mov d, m
68     mov e, a
69
70     ani 80h ; Записываю итоговый знак при умножении в первый бит регистра C
71     ora c
72     mov c, a
73     mov a, d
74     ani 80h
75     ora c
76     mov c, a
77     mov a, e
78
79
80     ani 80H ; Модуль E
81     mov a, e
82     jz skip_inv_1
83     xri ffh
84     inr a
85     mov e, a
86     skip_inv_1:
87
88     mov a, d
89     ani 80H ; Модуль D
90     mov a, d
91     jz skip_inv_2
92     xri ffh
93     inr a
94     mov d, a
95     skip_inv_2:
96
97     mvi a, 0; путем циклического сложения
98     MUL_LOOP:
99     add D
100    dcr E
101    JNZ MUL_LOOP
102
103    mov e, a; Проверка на знаковое переполнение
104    ani 80h
105    jz skip_inr_c
106    inr c
107
108    skip_inr_c:
109    mov a, c; Присваивание результату корректного знака при умножении
110    ani 80h
111    jz skip_chg ; Если первый бит C равен 0(C положительное) то пропускаем изменение знака результата
112    mov a, e ; Иначе делаем результат отрицательным
113    ori 80h
114    mov e, a
115
116    skip_chg: ; Делаем C положительным для следующего цикла

```

```

117     mov a, c
118     ani 7Fh
119     mov c, a
120
121     mov a, e ; Переходим к следующему элементу массива
122     inc h
123     dec b
124     jnz LOOP2
125
126 hlt
127 ; Добавил нули в промежутках чтобы было удобнее их разделять в ОЗУ
128 DB 0H
129 DB 0H
130 ARRAY1:
131     DB 02H, 03H, 04H, 05H, 06H
132
133 DB 0H
134 DB 0H
135 ARRAY2:
136     DB FFH, 02H, 01H, 01H, 01H
137 DB 0H
138 DB 0H

```

Проверка:

SUCCESS

Code

```

inc c
skip_inr_c:
mov a, c; Присваивание результату корректного знака при
умножении
ani 80h
jz skip_chg ; Если первый бит C равен 0(С положительное)
то пропускаем изменение знака результата
mov a, e ; Иначе делаем результат отрицательным
ori 80h
mov e, a

skip_chg: ; Делаем C положительным для следующего цикла
mov a, c
ani 7Fh
mov c, a

mov a, e ; Переходим к следующему элементу массива
inc h
dec b
jnz LOOP2

hlt
; Добавил нули в промежутках чтобы было удобнее их разделять в
ОЗУ
DB 0H
DB 0H
ARRAY1:
    DB 02H, 03H, 04H, 05H, 06H

DB 0H
DB 0H
ARRAY2:
    DB FFH, 02H, 01H, 01H, 01H
DB 0H
DB 0H

```

CPU State

a: B4	b: 00	c: 01	d: 05	e: B4
h: 00	l: 9A	pc: 008C	sp: 1000	halted: 1

Flags

Sign: 00	Zero: 01	Parity: 01	Carry: 00
----------	----------	------------	-----------

RAM Hex

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	06	05	0E	00	21	8E	00	11	95	00	31	00	10	C5	0E	00
001	7E	47	E6	80	B1	4F	EB	7E	E6	80	0F	B1	4F	78	96	77
002	E6	80	0F	0F	B1	4F	EE	80	CA	34	00	4F	EE	60	CA	34
003	00	C3	39	00	C1	0C	C3	3A	00	C1	EB	23	13	05	C2	0D
004	00	21	95	00	06	04	7E	23	56	5F	E6	80	B1	4F	7A	E6
005	80	B1	4F	7B	E6	80	7B	CA	5E	00	EE	FF	3C	5F	7A	E6
006	80	7A	CA	69	00	EE	FF	3C	57	3E	00	82	1D	C2	6B	00
007	5F	E6	80	CA	77	00	0C	79	E6	80	CA	81	00	7B	F6	80
008	5F	79	E6	7F	4F	7B	23	05	C2	48	00	76	00	00	02	03
009	04	05	06	00	00	03	01	03	04	05	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

From address (hex): 0000 Show

Labels

LOOP1:	000D
SKIP:	0034
next_loop:	0039
skip_2:	003A
LOOP2:	0048
skip_inv_1:	005E
skip_inv_2:	0069
MUL_LOOP:	006B
skip_inr_c:	0077
skip_chg:	0081
ARRAY1:	008E
ARRAY2:	0095

P.S Итоговое выражение выглядит как $3*1*3*4*5=180=B4h$. Так как тут была реализована знаковая арифметика число B4h на самом деле равно -52 произошло знаковое переполнение что и отображено в регистре C=1(то есть произошло одно знаковое переполнение).