Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление системного и прикладного программного обеспечения

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

курса «Основы профессиональной деятельности»

по теме: «Исследование работы БЭВМ»

Вариант № 1025

Выполнил студент:
Тюрин Иван Николаевич
группа: P3110

Преподаватель:
Клименков С. В.,
Ларочкин Г. И.

Санкт-Петербург, 2022 г.

Содержание

Лабораторная работа № 3. Исследование работы БЭВМ **2**
1. Задание варианта № 1025 2
2. Описание программы 3
    1. Назначение программы и реализуемые ею функция 3
    2. Область представления и допустимых значений 4
    3. Трассировка программы 5
3. Программа 5
4. Эквивалентная программа 8
5. Вывод 8
6. Дополнительное задание: обратная польская нотация 8

Лабораторная работа № 3
Исследование работы БЭВМ

1. Задание варианта № 1025

, , ,

По выданному преподавателем варианту восстановить текст заданного варианта программы, определить предназначение и составить описание программы, определить область представления и область допустимых значений исходных данных и результата, выполнить трассировку программы.

Ход работы, содержание отчета и контрольные вопросы описаны в методических указаниях

```
0A5: + 0200  |  0B3:   0740  |  ----------  |  676:   002E
0A6:   EE19  |  0B4:   4E0B  |  669:   AC01  |
0A7:   AE17  |  0B5:   EE0A  |  66A:   F203  |
0A8:   0700  |  0B6:   AE06  |  66B:   7E09  |
0A9:   0C00  |  0B7:   0C00  |  66C:   F005  |
0AA:   D669  |  0B8:   D669  |  66D:   F804  |
0AB:   0800  |  0B9:   0800  |  66E:   0500  |
0AC:   4E13  |  0BA:   4E05  |  66F:   0500  |
0AD:   EE12  |  0BB:   EE04  |  670:   4E05  |
0AE:   AE0F  |  0BC:   0100  |  671:   CE01  |
0AF:   0740  |  0BD:   ZZZZ  |  672:   AE02  |
0B0:   0C00  |  0BE:   YYYY  |  673:   EC01  |
0B1:   D669  |  0BF:   XXXX  |  674:   0A00  |
0B2:   0800  |  0C0:   1F3F  |  675:   0F8B  |
```

, , ,

2. Описание программы

2. 1. Назначение программы и реализуемые ею функция

Описание программы представлено в таблице 1.1. В результате выполнения программы происходит вычисление суммы: $f(Y+1)+f(X)+1+f(Z)-1$, где $f(n)$ - подпрограмма, принимающая операндом n.

Адрес	**Данные/ /Команда**	**Мнемоника**	**Описание**
0x075	0x0200	`cla`	Очистка AC
0x076	0xee19	`st VAR`	Обнуление переменной VAR
0x077	0xae16	`ld Y`	Загрузка Y в AC
0x078	0x0700	`inc`	Увеличение AC на 1
0x079	0x0c00	`push`	Положить значение AC на стэк
0x07a	0xd694	`call $FUNC`	Вызов функции FUNC
0x07b	0x0800	`pop`	Загрузка из стэка в AC
0x07c	0x4e13	`add VAR`	Прибавление переменной VAR к AC
0x07d	0xee12	`st VAR`	Сохранение значения в VAR
0x07e	0xae10	`ld X`	Загрузка переменной X в AC
0x07f	0x0c00	`push`	Положить значение AC на стэк
0x080	0xd694	`call $FUNC`	Вызов функции FUNC
0x081	0x0800	`pop`	Загрузка из стэка в AC
0x082	0x0700	`inc`	Увеличение AC на 1
0x083	0x4e0c	`add VAR`	Прибавление переменной VAR к AC
0x084	0xee0b	`st VAR`	Сохранение значения в VAR
0x085	0xae07	`ld Z`	Загрузка переменной X в AC
0x086	0x0c00	`push`	Положить значение AC на стэк
0x087	0xd694	`call $FUNC`	Вызов функции FUNC
0x088	0x0800	`pop`	Загрузка из стэка в AC
0x089	0x0740	`dec`	Увеличение AC на 1
0x08a	0x4e05	`add VAR`	Прибавление переменной VAR к AC
0x08b	0xee04	`st VAR`	Сохранение значения в VAR
0x08c	0x0100	`hlt`	Останов
0x08d	0xzzzz	Z	Параметр Z
0x08e	0xyyyy	Y	Параметр Y
0x08f	0xxxxx	X	Параметр X
0x090	0x023e	VAR	Переменная, хранящая результат

Таблица 1.2: Описание работы программы

Адрес	Данные/ /Команда	Мнемоника	Описание
0x694	0xac01	ld &1	Загрузка операнда в AC
0x695	0xf204	bmi LTEQZVL1	Переход, если операдн отрицательный
0x696	0xf003	beq LTEQZVL1	Переход, если операнд 0
0x697	0x7e09	cmp VAL1	Сравнение AC с VAL1
0x698	0xf005	beq LTEQVAL1	Переход, если операнд равен VAL1
0x699	0xf804	blt LTEQVAL1	Переход, если операнд меньше VAL1
0x69a	0x4c01	add &1	Увеличить операнд на еще одно его значение, метка LTEQZVL1:
0x69b	0x4c01	add &1	Увеличить операнд на еще одно его значение
0x69c	0x4e05	add VAL2	Увеличить операнд на VAL2
0x69d	0xce01	br STOP	Переход в конец функции
0x69e	0xae02	ld VAL1	Приравнять операнд (AC) к VAL1, метка LTEQVAL1:
0x69f	0xec01	st &1	Сохранить результат функции по (SP+1)
0x6a0	0x0a00	ret	Выход из функции
0x6a1	0x00d0	VAL1	
0x6a2	0x00b7	VAL2	

Таблица 1.4: Описание работы подпрограммы

2. 2. Область представления и допустимых значений

Значения переменных X, Y, Z ограничены из-за функции f и суммы значений f. Таким образом, крайние нижне и верхнее возможные значения X и Z – -1881 и 1760, соответственно. Крайние значения Y – -1880 и 1761. Они представлены на графике 1.2. Максимальное и минимальнео значение f – -5461 и 5461.

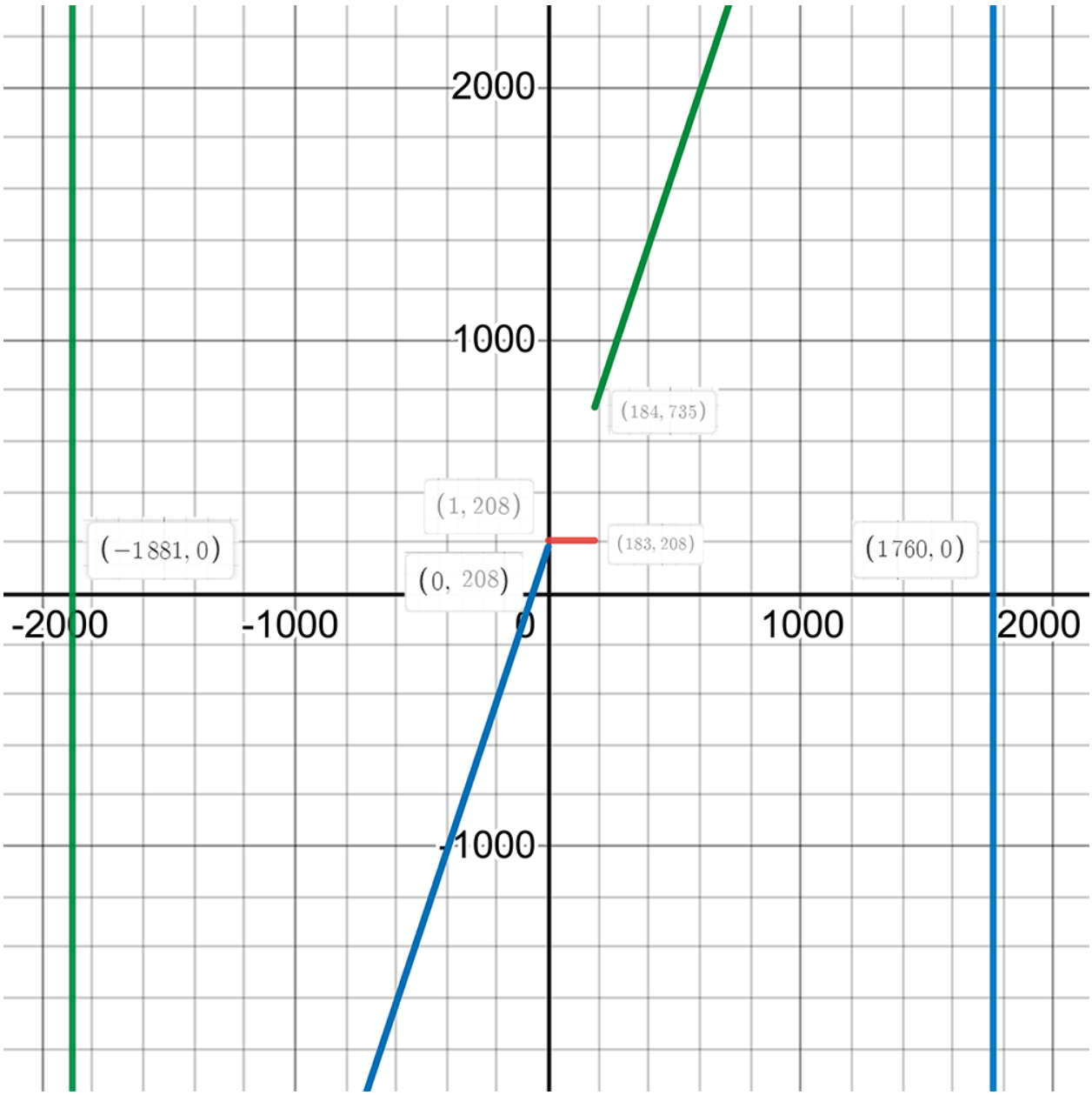

Рис. 1.2: Область допустимых значений параметров X, Y, Z и область значений функции.

2. 3. Трассировка программы

Трассировка программы для параметров X=`0x00aa`, Y=`0x00b8`, Z=`0xffff` представлена в таблице 1.2.??.

3. Программа

Была составлена программа на языке ассемблера. Она представлена в листинге 1.10.

Выполняемая команда		Содержимое регистров процессора послевыполнения команды.								Ячейка, содержимое которой изменилось после выполнения команды	
Адрес	**Значение**	**IP**	**CR**	**AR**	**DR**	**SP**	**BR**	**AC**	**NZVC**	**Адрес**	**Значение**
075	0200	075	0000	000	0000	000	0000	0000	0100		
075	0200	076	0200	075	0200	000	0075	0000	0100		
076	EE19	077	EE19	090	0000	000	0019	0000	0100	090	0000
077	AE16	078	AE16	08E	00B8	000	0016	00B8	0000		
078	0700	079	0700	078	0700	000	0078	00B9	0000		
079	0C00	07A	0C00	7FF	00B9	7FF	0079	00B9	0000	7FF	00B9
07A	D694	694	D694	7FE	007B	7FE	D694	00B9	0000	7FE	007B
694	AC01	695	AC01	7FF	00B9	7FE	0001	00B9	0000		
695	F204	696	F204	695	F204	7FE	0695	00B9	0000		
696	F003	697	F003	696	F003	7FE	0696	00B9	0000		
697	7E09	698	7E09	6A1	00D0	7FE	0009	00B9	1000		
698	F005	699	F005	698	F005	7FE	0698	00B9	1000		
699	F804	69E	F804	699	F804	7FE	0004	00B9	1000		
69E	AE02	69F	AE02	6A1	00D0	7FE	0002	00D0	0000		
69F	EC01	6A0	EC01	7FF	00D0	7FE	0001	00D0	0000	7FF	00D0
6A0	0A00	07B	0A00	7FE	007B	7FF	06A0	00D0	0000		
07B	0800	07C	0800	7FF	00D0	000	007B	00D0	0000		
07C	4E13	07D	4E13	090	0000	000	0013	00D0	0000		
07D	EE12	07E	EE12	090	00D0	000	0012	00D0	0000	090	00D0
07E	AE10	07F	AE10	08F	00AA	000	0010	00AA	0000		
07F	0C00	080	0C00	7FF	00AA	7FF	007F	00AA	0000	7FF	00AA
080	D694	694	D694	7FE	0081	7FE	D694	00AA	0000	7FE	0081
694	AC01	695	AC01	7FF	00AA	7FE	0001	00AA	0000		
695	F204	696	F204	695	F204	7FE	0695	00AA	0000		
696	F003	697	F003	696	F003	7FE	0696	00AA	0000		
697	7E09	698	7E09	6A1	00D0	7FE	0009	00AA	1000		
698	F005	699	F005	698	F005	7FE	0698	00AA	1000		
699	F804	69E	F804	699	F804	7FE	0004	00AA	1000		
69E	AE02	69F	AE02	6A1	00D0	7FE	0002	00D0	0000		
69F	EC01	6A0	EC01	7FF	00D0	7FE	0001	00D0	0000	7FF	00D0
6A0	0A00	081	0A00	7FE	0081	7FF	06A0	00D0	0000		
081	0800	082	0800	7FF	00D0	000	0081	00D0	0000		
082	0700	083	0700	082	0700	000	0082	00D1	0000		
083	4E0C	084	4E0C	090	00D0	000	000C	01A1	0000		
084	EE0B	085	EE0B	090	01A1	000	000B	01A1	0000	090	01A1
085	AE07	086	AE07	08D	FFFF	000	0007	FFFF	1000		
086	0C00	087	0C00	7FF	FFFF	7FF	0086	FFFF	1000	7FF	FFFF
087	D694	694	D694	7FE	0088	7FE	D694	FFFF	1000	7FE	0088
694	AC01	695	AC01	7FF	FFFF	7FE	0001	FFFF	1000		
695	F204	69A	F204	695	F204	7FE	0004	FFFF	1000		
69A	4C01	69B	4C01	7FF	FFFF	7FE	0001	FFFE	1001		
69B	4C01	69C	4C01	7FF	FFFF	7FE	0001	FFFD	1001		
69C	4E05	69D	4E05	6A2	00B7	7FE	0005	00B4	0001		
69D	CE01	69F	CE01	69D	069F	7FE	0001	00B4	0001		
69F	EC01	6A0	EC01	7FF	00B4	7FE	0001	00B4	0001	7FF	00B4
6A0	0A00	088	0A00	7FE	0088	7FF	06A0	00B4	0001		
088	0800	089	0800	7FF	00B4	000	0088	00B4	0001		
089	0740	08A	0740	089	0740	000	0089	00B3	0001		
08A	4E05	08B	4E05	090	01A1	000	0005	0254	0000		
08B	EE04	08C	EE04	090	0254	000	0004	0254	0000	090	0254
08C	0100	08D	0100	08C	0100	000	008C	0254	0000		

Таблица 1.6: Трассировка программы

```
org     0x075
START:
cla
st      VAR     ;VAR=0
ld      Y       ;
inc             ;
push            ;
call    $FUNC   ;
pop             ;
add     VAR     ;
st      VAR     ;VAR+=f(y+1)
ld      X       ;
push            ;
call    $FUNC   ;
pop             ;
inc             ;
add     VAR     ;
st      VAR     ;VAR+=f(x)+1
ld      Z       ;
push            ;
call    $FUNC   ;
pop             ;
dec             ;
add     VAR     ;
st      VAR     ;VAR+=f(z)-1
hlt
Z:      word    0xZZZZ
Y:      word    0xYYYY
X:      word    0xXXXX
VAR:    word    0x023e  ;:0x090
org 0x694
FUNC:                   ;f(x)
ld      &1              ;
bmi     LTEQZVL2        ;
beq     LTEQZVL2        ;if(x<=0||x>183): return x*3+183
cmp     VAL1            ;
beq     LTEQVAL2        ;
blt     LTEQVAL2        ;if(x<=183): return 208
LTEQZVL2:
add     &1
add     &1
add     VAL2
br      STOP
LTEQVAL2:
ld      VAL1
STOP:
st      &1
ret
VAL1:   word    0x00d0  ;=208
VAL2:   word    0x00b7  ;=183
```

Листинг 1.2: Код программы на языке ассемблена БЭВМ

4. Эквивалентная программа

Была составлена эквивалентная программа на языке Python. Она представлена в листинге 1.4.

```
def f(x):
    if(x<=0 or x>208):
        return x*3+183
    if(x<=208):
        return 208

var = f(y+1)+f(x)+1+f(z)-1
```

Листинг 1.4: Код эквивалентной программы на языке Python

5. Вывод

Научился читать коды БЭВМ, сдерживать ярость, проверять номер варианта. Научился писать программы с ветвлениями и подпрограммами на ассемблере БЭВМ.

6. Дополнительное задание: обратная польская нотация

Обратная польская нотация – форма записи выражений, при которой символ операции ставится после своих операндов, например выражение из "обычной инфиксной, нотации $((2 + 2) \cdot 2)$ в обратной польской нотации будет выглядеть следующим образом: $(2\ 2 + 2\ \cdot)$.

Такой способ записи удобен для написания низкоуровневых программ работающих со стеком, т.к. не требует обработки приоритета операций, чем позволяет упростить микросхему ВУ.

Была написана программа на языке ассемблера БЭВМ, которая выполняет рассчеты по записаному в коде скрипту в формате обратной польской нотации. Обладает возможными операциями `_ADD` – сложение двух операндов, `_SUB` – вычесть из первого операнда второй, `_MUL` – быстрое перемножение операндов с учетом знака. Подпрограммы используют два верхних значения на стеке (до адреса возврата), при помощи подпрограммы `GOBACK` завершается работа других подпрограмм: указатель стека смещается вниз и на вершине оказывается результат выполнения операций, что требуется для продолжения работы. Результат выполнения скрипта находтся в ячейке `0x7ff` (вершина стека).

```
1  org 0x010
2
3  _ADD:
4      ld &1        ;arg1
5      add &2       ;+arg2
6  GOBACK:          ;moves AC to arg1 position, erases ret address,
       return to script
7      st &2
8      ld &0
9      st &1
10     pop
11     ret
12
13 _SUB:
14     ld &2            ;arg1
15     sub &1           ;-arg2
16 jump GOBACK
```

Листинг 1.6: Код подпрограмм сложения и вычитания на языке ассемблена БЭВМ

```
_MUL:
    cla
    st &-1          ;temp res
    st &-2          ;res sign
    ld &2
    bpl PLUS1       ;arg1>0
    neg
    st &2
    ld &-2
    not
    st &-2
    PLUS1:
    ld &1
    bpl MULLOOP     ;arg2>0
    neg
    st &1
    ld &-2
    not
    st &-2
    MULLOOP:
        ld &1       ;arg2
        beq BREAK   ;multiply by zero
        asr
        st &1
        bcc ODD
        ld &-1
        add &2
        st &-1
        ODD:
        ld &2       ;arg1*2
        asl
        st &2
        jump MULLOOP
    BREAK:
    ld &-2
    bpl PLUSRES     ;change result sign
    ld &-1
    neg
    br ENDMUL
    PLUSRES:
    ld &-1
    ENDMUL:
jump GOBACK
```

Листинг 1.8: Код подпрограммы умножени на языке ассемблена БЭВМ

```
1
2 START:
3       ld #2
4       push
5       ld #3
6       push
7       call _ADD
8
9       ld #7
10      push
11      ld #5
12      push
13      call _SUB
14 call _MUL
15 hlt
```

Листинг 1.10: Код скрипта в формате обратной польской нотации на языке ассемблена БЭВМ ((2+3)·(7-5) $\sim$ 2 3 + 7 5 - ·)