Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4
по основам профессиональной деятельности

Вариант: 1984

Группа: P3114

Студент: Лагус

Максим Сергеевич

Преподаватель: Перминов Илья Валентинович

г. Санкт-Петербург

Февраль, 2021

Задание:

Лабораторная работа №4

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

Введите номер варианта 1984

```
3A0: + 0200  |  3AE:   4E0D  |  3BC:   0731  |  72B:   EC01
3A1:   EE1A  |  3AF:   EE0C  |  ----------  |  72C:   0A00
3A2:   AE16  |  3B0:   AE09  |  71F:   AC01  |  72D:   F8CA
3A3:   0700  |  3B1:   0740  |  720:   F001  |  72E:   00FD
3A4:   0C00  |  3B2:   0C00  |  721:   F308  |
3A5:   D71F  |  3B3:   D71F  |  722:   7E0A  |
3A6:   0800  |  3B4:   0800  |  723:   F806  |
3A7:   0740  |  3B5:   0740  |  724:   F005  |
3A8:   4E13  |  3B6:   6E05  |  725:   0500  |
3A9:   EE12  |  3B7:   EE04  |  726:   0500  |
3AA:   AE10  |  3B8:   0100  |  727:   4C01  |
3AB:   0C00  |  3B9:   ZZZZ  |  728:   4E05  |
3AC:   D71F  |  3BA:   YYYY  |  729:   CE01  |
3AD:   0800  |  3BB:   XXXX  |  72A:   AE02  |
```

Выполнение работы:

Расшифровка текста исходной программы

Адрес	Код	Мнемоника	Комментарии
3A0	0200	CLA	Очистка регистра
3A1	EE1A	ST IP+1A	Обнуляем переменную RES
3A2	AE16	LD IP+16	Z --> AC, Загружаем в AC переменную Z
3A3	0700	INC	Z+1 --> AC
3A4	0C00	PUSH	Кладём на вершину стека Z+1
3A5	D71F	CALL 71F	Кладём на вершину стека адрес возврата, и переходим по адресу 71F
3A6	0800	POP	Кладём в аккумулятор результат работы подпрограммы
3A7	0740	DEC	AC-1 --> AC

3A8	4E13	ADD IP+13	AC+RES --> AC
3A9	EE12	ST IP+12	AC --> RES
3AA	AE10	LD IP+10	X --> AC, Загружаем в AC переменную X
3AB	0C00	PUSH	Кладём на вершину стека переменную X
3AC	D71F	CALL 71F	Кладём на вершину стека адрес возврата, и переходим по адресу 71F
3AD	0800	POP	Кладём в аккумулятор результат работы подпрограммы
3AE	4E0D	ADD IP+D	AC+RES --> AC
3AF	EE0C	ST IP+C	AC --> RES
3B0	AE09	LD IP+9	Y --> AC, Загружаем в AC переменную Y
3B1	0740	DEC	Y-1 --> AC
3B2	0C00	PUSH	Кладём на вершину стека Y-1
3B3	D71F	CALL 71F	Кладём на вершину стека адрес возврата, и переходим по адресу 71F
3B4	0800	POP	Кладём в аккумулятор результат работы подпрограммы
3B5	0740	DEC	AC-1 --> AC
3B6	6E05	SUB IP+5	AC-RES --> AC
3B7	EE04	ST IP+4	AC --> RES
3B8	0100	HLT	Завершение работы программы
3B9	ZZZ	Z	Переменная

3BA	YYY	Y	Переменная
3BB	XXX	X	Переменная
3BC	0731	RES	Переменная, результат работы программы
-------	------	------------------	----------
71F	AC01	LD &1	Загрузка в AC первого сверху элемента стека.
720	F001	BEQ 01	Если элемент равен нулю, то пропускаем следующую операцию
721	F308	BPL 08	Если элемент положительный, переходим на 72A
722	7E0A	CMP IP+A	Сравниваем AC и COMPARATOR
723	F806	BLT 06	Если AC меньше чем COMPARATOR, то переходим в 72A
724	F005	BEQ 05	Если AC равно COMPARATOR, то переходим в 72A
725	0500	ASL	AC * 2 --> AC
726	0500	ASL	AC * 2 --> AC
727	4C01	ADD &1	Прибавляем к AC первый сверху элемент стека
728	4E05	ADD IP+5	Прибавляем к AC переменную ADDITIONAL
729	CE01	JUMP IP+1	Пропускаем следующую операцию
72A	AE02	LD IP+2	Загружаем в AC переменную COMPARATOR
72B	EC01	ST &1	Загружаем содержимое AC в первый сверху элемент стека
72C	0A00	RET	Возвращаемся из подпрограммы
72D	F8CA	COMPARATOR	Переменная
72E	00FD	ADDITIONAL	Переменная

Описание программы

1) Расположение программы в памяти

Переменные, поступающие на вход программы, расположены в ячейках

3B9 - 3BC

Сама программа расположена в ячейках 3A0 - 3B8

Вызываемая из основной программы подпрограмма расположена в ячейках 71F - 72C

Переменные для подпрограммы расположены в 72D , 72E

2) Область представления

X, Y, Z — обрабатываемые программой переменные, 16 разрядные знаковые числа

RES — результат работы программы, 16-разрядное знаковое число

COMPARATOR, ADDITIONAL - переменные, задающие работу подпрограммы, 16 разрядные знаковые числа

3) Назначение программы

Сначала определим семантику работы подпрограммы, записав алгоритм её работы на псевдокоде

```
let additional
let comparator
    func f(x):
        if (x ≤ 0 && x > comparator):
            x = 5x + additional
        else:
            x = comparator
        return x
```

Теперь мы можем записать результат работы всей программы

```
let res = 0
    func main(x,y,z):
        res += f(z + 1) - 1
        res += f(x)
        res = f(y - 1) - 1 - res
        return res
```

4) Область допустимых значений

Рассмотрим два варианта

- COMPARATOR ≥ 0

В таком случае, результатом функции f() всегда будет COMPARATOR (следует из логики работы подпрограммы).

Тогда, $X, Y, Z \in [-2^{15} + 1 ; 2^{15}]$, так как они не учавствуют не в каких арифметических операциях, кроме ±1, и, как следствие, не могут вызвать переполнения при таком ОДЗ.

После второго вызова подпрограммы, в RES хранится f(z - 1) + f(x), что в нашем случае равно 2 * COMPARATOR. Чтобы не случилось переполнения, мы должны требовать: $COMPARATOR \in [0; 2^{14}]$.

Так как ADDITIONAL вообще не учавствует в исполнении программы при данных условиях, он может принимать любые значения.

$ADDITIONAL \in [-2^{15} ; 2^{15} - 1]$.

В переменной RES, после исполнения программы, будет лежать значение, равное: - COMPARATOR. Значит, $RES \in [-2^{14} ; 0]$.

- COMPARATOR < 0

Если X,Y,Z < COMPARATOR, то подпрограмма возвращает COMPARATOR, и ограничения на такой случай мы уже рассматривали в первом пункте. Если же обрабатываемые переменные больше чем COMPARATOR и меньше нуля, то мы получаем:

COMPARATOR $<$ X,Y,Z $\leq$ 0

Затем подпрограмма вычисляет значение 5X + ADDITIONAL.

Чтобы не допустить переполнения, мы должны наложить ограничения на ADDITIONAL и COMPARATOR, так как именно эти две переменные задают возвращаемое из подпрограммы значение.

Чтобы 2 * (5X + ADDITIONAL) не вызвало переполнения из отрицательного числа в положительное:

COMPARATOR $\in [-2^9; 0]$.

Чтобы 2 * (5X + ADDITIONAL) не вызвало переполнения в положительную сторону:

ADDITIONAL $\in [0; 2^{14} - 1]$.

Заметим, что на сами переменные X,Y,Z не накладывается дополнительных ограничений, так как все возможные переполнения с участием этих переменных уже учтены в ограничениях на COMPARATOR, ADDITIONAL:

X, Y, Z $\in [-2^{15}; 2^{15} - 1]$.

Трассировка программы