1. Реализация функции для VM-Edem, принимающей на вход N (считывает со стека) и вычисляющей N-ое число Фибоначчи (результат записывает на стек и выводится на экран).

\_**func fibonacci**

\_**RAdd** 1 1 // в 1 спец. регистре будет счетчик

\_**Pop** 2 // сохраняем со стека аргумент в регистр 2

\_**RAdd** 3 1 // первое число Фибоначчи в регистр 3

\_**RAdd** 4 1 // второе число Фибоначчи в регистр 4

// основное ветвление-цикл

\_**label** 0

\_**if** **R**-1 < **R**-2 // проверяем не дошли ли до нужного числа

\_**goto** 1 // если нет, то идем в блок вычислений

\_**goto** 2 // если да, то идем в блок выводы

// блок для вычисления следующего числа

\_**label** 1

\_**RSum** **R**-3 **R**-4 4 // вычисляем следующее число Фибоначчи

\_**RDiff** **R**-4 **R**-3 3 // сохраняем предыдущее число

\_**RSum** **R**-1 1 1 // увеличиваем переменную счетчик на 1

\_**goto** 0 // возвращаемся к проверке условия

// блок для вывода результата

\_**label** 2

\_**Add R**-4 // помещаем значение регистра 4 на стек

\_**Print** // выводим на экран значение с вершины стека

// вызов функции из основной программы

**Add** 36 // помещаем на вершину стека аргумент для функции

**func fibonacci** // вызываем функцию

1. Реализация функции для VM-Edem, принимающей на вход N1, N2 и сортирующий значения регистров в диапазоне от N1 до N2, алгоритмом сортировкой выбором.

\_**func selectionsort**

\_**Pop** 2 // читаем со стека верхнюю границу в регистр 2

\_**Pop** 1 // читаем со стека нижнюю границу в регистр 1

\_**RAdd** 3 **R**-1 //в регистре 3 храним номер текущего эл-та

\_**RAdd** 4 **R**-1 //в регистре 4 храним номер максимального //элемента, найденного за итерацию цикла

\_**RAdd** 5 0 //регистр 5 является промежуточной переменной, //для перемещения значений регистров

//цикл с основным условием

\_**label** 0

\_**if** **R**-3 < **R**-2 //если в пределах границы сортировки

\_**goto** 1 //проверяем не наибольший ли это элемент

\_**goto** 2 //если граница сортировки, проверяем на наибольший и меняем местами элементы (+уменьшаем гран.)

//проверяем текущий элемент на максимум

\_**label** 1

\_**if** **R-R-**3 > **R**-**R**-4 //если данный элемент максимальный

\_**RAdd** 4 **R**-3 //сохраняем номер нового максимального

\_**RAdd** 5 0 //пустое действие для else

\_**RSum** **R**-3 1 3 //берем следующий элемент для проверки

\_**goto** 0 //возвращаемся к основному циклу

//сравниваем с последним элементом

\_**label** 2

\_**if** **R-R-**3 < **R**-**R**-4 //если данный элемент НЕ максимальный

\_{

\_**RAdd** 5 **R**-**R**-3 //запоминаем значение на границе

\_**RAdd R**-3 **R**-**R**-4 //записываем максим. в элемент на границе

\_**RAdd R**-4 **R**-5 //сохраняем старое значение

\_}

\_**RAdd** 5 0 //пустое действие для else

//возвращаем инвариант цикла

\_**RDiff** **R**-2 1 2 //уменьшаем верхнюю границу (так как там //уже оказалось максимальное значение)

\_**RAdd** 3 **R**-1 //обнуляем «счетчик цикла»

\_**RAdd** 4 **R**-1 //«обнуляем» максимальный элемент

\_**if** **R**-1 <> **R**-2 //проверяем не сомкнулись ли границы

\_**goto** 0 //если нет, возвращаемся к основному циклу

\_**RAdd** 5 0 //если да, значит диапазон отсортирован

// вызов функции из основной программы

// заполняем регистры значениями

**RAdd** 350 34

**RAdd** 351 75

**RAdd** 352 29

**RAdd** 353 3

**RAdd** 354 18

**RAdd** 355 724

**Add** 350 // помещаем на стек нижнюю границу сортировки

**Add** 355 // помещаем на стек верхнюю границу сортировки

**func selectionsort** // вызываем функцию

// после завершения функции в данном диапазоне будут //отсортированные значения