For the program1, I think it should be divided into three parts: In, Compare, and Out. In the first part, it should be a loop for receiving the input numbers. So I choose the instruction SOB to loop. And the increase or decrees of the subscripts is use an index register to achieve. (By the way, LIX is a customized instruction invented by us, used for loading an immediate value to X.)

// LDR 3,0,20

//LDX 1,24

//IN 2,0

//STR 2,1,0

//LDA 2,1,0

//SIR 2,1

//STR 2,0,24

//LIX 1,0

//SOB 3,3,1

It is a sample loop, the index 1 is a like a subscripts.

And then is the part for comparing, it also use the SOB and the index register to loop, and use two new instructions CMB, CMT to compare and replace.

This is one loop:

// LDA 2,1,1

// STR 2,0,30

// LDX 2,30

// LDR 0,2,0

// LDR 1,1,0

// CMT 1,0

// LDR 0,1,0

// LDR 2,2,0

// CMB 0,2

// STR 0,0,27

// LDR 0,2,0

// SMR 1,0,27

// STR 1,0,27

// LDR 2,0,29

// CMB 1,2

// TRR 1,2

// JCC 4,3,21

// STR 0,0,2

// STR 1,0,29

Memory[29] is the mark to compare if it less than last one’s result. If did, then replace, and record the number to Memory[28].

And the final part is quite simple for those program, so I just use the LDR and OUT to make it.

//LDR 1,1,0

//LDR 0,0,28

// OUT 1,1

// OUT 0,1

There are some test sample (this piece of code can be put in the load() method of the RomLoader class)：

this.memory.write(101, 42);

this.memory.write(102, 18468);

this.memory.write(103, 6335);

this.memory.write(104, 26501);

this.memory.write(105, 19170);

this.memory.write(106, 15725);

this.memory.write(107, 11479);

this.memory.write(108, 29359);

this.memory.write(109, 26963);

this.memory.write(110, 24465);

this.memory.write(111, 5706);

this.memory.write(112, 28146);

this.memory.write(113, 23282);

this.memory.write(114, 16828);

this.memory.write(115, 9962);

this.memory.write(116, 492);

this.memory.write(117, 2996);

this.memory.write(118, 11943);

this.memory.write(119, 4828);

this.memory.write(120, 5437);

And if the required one is 140, then the output should be :140,42

With those samples given, we still recommend you input your own test cases in order to validate the program though.

This is the total program1 in the ROM (this piece of code has already been put in the load() method of the RomLoader class, and is deemed as the initial program):

// Data

this.memory.write(20, 21);

this.memory.write(24, 120);

this.memory.write(26, 200);

this.memory.write(25, 300);

this.memory.write(31, 20);

this.memory.write(29, 65535);

this.memory.write(30, 100);

// in：

// 8 LDX 3, 26 1000010011011010

this.memory.write(8, 34010);

// 9 JSR 3, 0 0011000011000000

this.memory.write(9, 12480);

// LDR 3,0,20

this.memory.write(200, 1812); //0000011100010100

// LDX 1, 24 1000010001011000

this.memory.write(201, 33880);

// IN 2, 0 1100011000000000

this.memory.write(202, 50688);

// STR 2, 1, 0 0000101001000000

this.memory.write(203, 2624);

// LDA 2, 1, 0 0000111001000000

this.memory.write(204, 3648);

// SIR 2, 1 0001111000000001

this.memory.write(205, 7681);

// STR 2,0,24 0000101000011000

this.memory.write(206, 2584);

// LIX 1, 0 1000110001000000

this.memory.write(207, 35904);

// SOB 3, 3, 1 0011101111000001

this.memory.write(208, 15297);

// AIR 3, 10

this.memory.write(209, 6922); // 0001101100001010

// RFS 0 0011010000000000

this.memory.write(210, 13312);

// M[25]=300

// compare:

// 10 LIX 1, 0 1000110001000000

this.memory.write(10, 35904);

// 11 LIX 3, 0 1000110011000000

this.memory.write(11, 36032);

// 12 LDX 3, 25 1000010011011001

this.memory.write(12, 34009);

// 13 JSR 3, 0 0011000011000000

this.memory.write(13, 12480);

// LDR 3,0,31

this.memory.write(300, 1823); //0000011100011111

// LDX 1, 30

this.memory.write(301, 33886); //1000010001011110

// LDA 2, 1, 1

this.memory.write(302, 3649); //0000111001000001

// STR 2,0,30

this.memory.write(303, 2590); //0000101000011110

// LDX 2, 30

this.memory.write(304, 33950); //1000010010011110

// LDR 0, 2, 0

this.memory.write(305, 1152); //0000010010000000

// LDR 1, 1, 0

this.memory.write(306, 1344); //0000010101000000

// CMT 1, 0

this.memory.write(307, 22784); //0101100100000000

// LDR 0, 1, 0

this.memory.write(308, 1088); //0000010001000000

// LDR 2, 2, 0

this.memory.write(309, 1664); //0000011010000000

// CMB 0, 2

this.memory.write(310, 23680); //0101110010000000

// STR 0,0,27

this.memory.write(311, 2075); //0000100000011011

// LDR 0, 2, 0

this.memory.write(312, 1152); //0000010010000000

// SMR 1,0,27

this.memory.write(313, 5403); //0001010100011011

// STR 1,0,27

this.memory.write(314, 2331); //0000100100011011

// LDR 2,0,29

this.memory.write(315, 1565); //0000011000011101

// CMB 1, 2

this.memory.write(316, 23936); //0101110110000000

// TRR 1, 2

this.memory.write(317, 18816); //0100100110000000

// JCC 4, 3, 21

this.memory.write(318, 11221); //0010101111010101

// STR 0, 0, 28

this.memory.write(319, 2076); //0000100000011100

// STR 1, 0, 29

this.memory.write(320, 2333); //0000100100011101

// LDR 2, 0 ,30

this.memory.write(321, 1566); //0000011000011110

// AIR 2, 1

this.memory.write(322, 6657); //0001101000000001

// STR 2, 0, 30

this.memory.write(323, 2590); //0000101000011110

// LIX 2, 0

this.memory.write(324, 35968); //1000110010000000

// SOB 3, 3, 4

this.memory.write(325, 15300); //0011101111000100

// AIR 3, 14

this.memory.write(326, 6926); // 0001101100001110

// RFS 0

this.memory.write(327, 13312); //0011010000000000

// out：

// 14 LDR 1, 1, 0 0000010101000000

this.memory.write(14, 1344);

// 15 LDR 0, 0, 28 0000010000011100

this.memory.write(15, 1052);

// 16 OUT 1, 1 1100100100000001

this.memory.write(16, 51457);

// 17 OUT 0, 1 1100100000000001

this.memory.write(17, 51201);