University of Dhaka

Department of Computer Science and Engineering

CSE 3113 - Microprocessor and Assembly Language Lab

Batch 28 / 3RD Year 1ST Semester

Lab 5

Submitted To:

Dr. Upama Kabir
Dr. Mosarrat Jahan
Mr. Jargis Ahmed
Mr. Palash Roy

Submitted By:

Farzana Tasnim (14)

1 Lab Tasks

1.1 Task 1

Write assembly language to perform a simple Boolean operation to calculate the bitwise calculation of F = W.X + Y.Z

1.1.1 Screenshot that shows the state of the system after the code has been loaded.

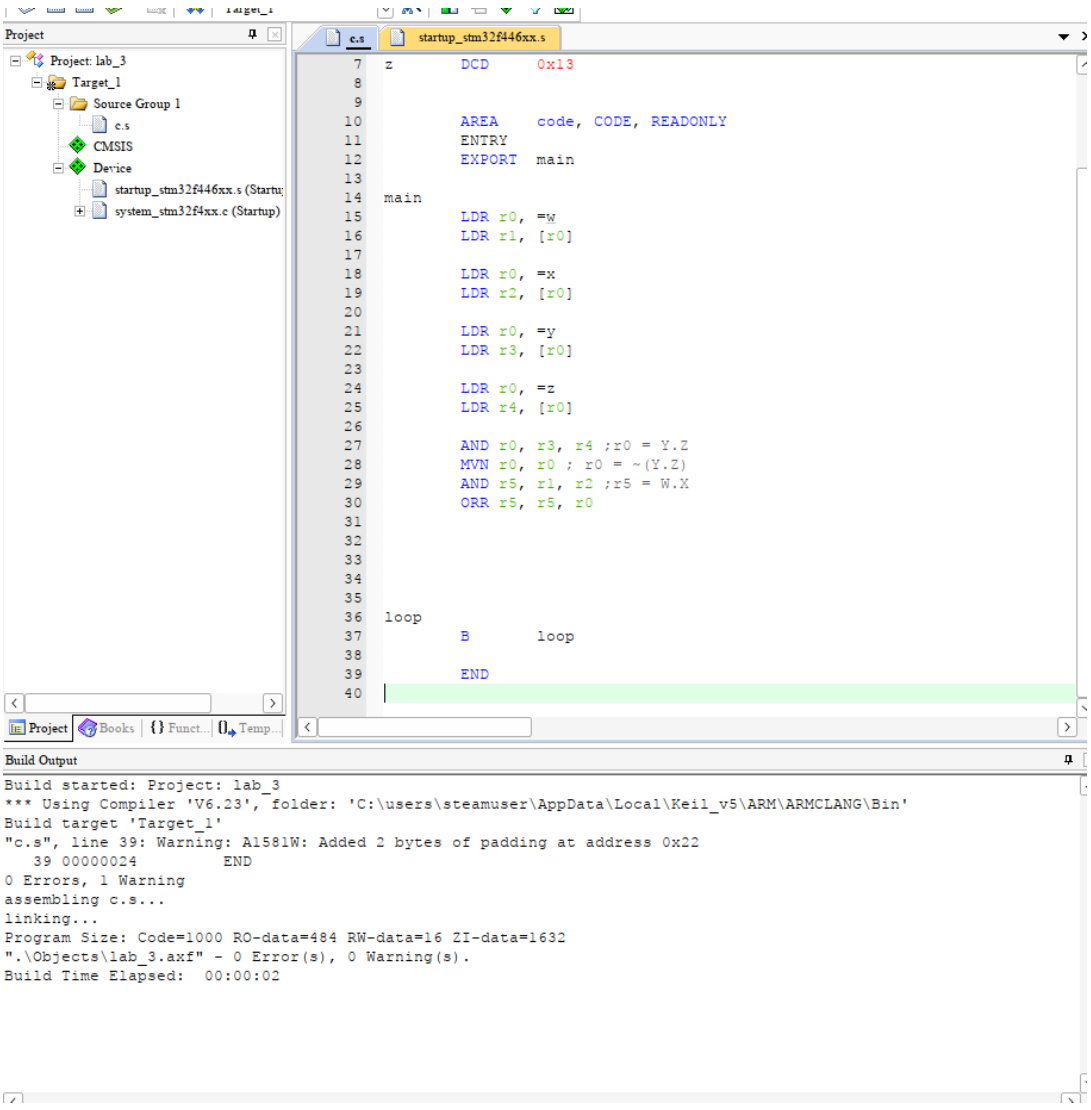

Figure 1: After build and debugging, this is the state

1.1.2 Screenshot that shows the situation after the code has been executed.

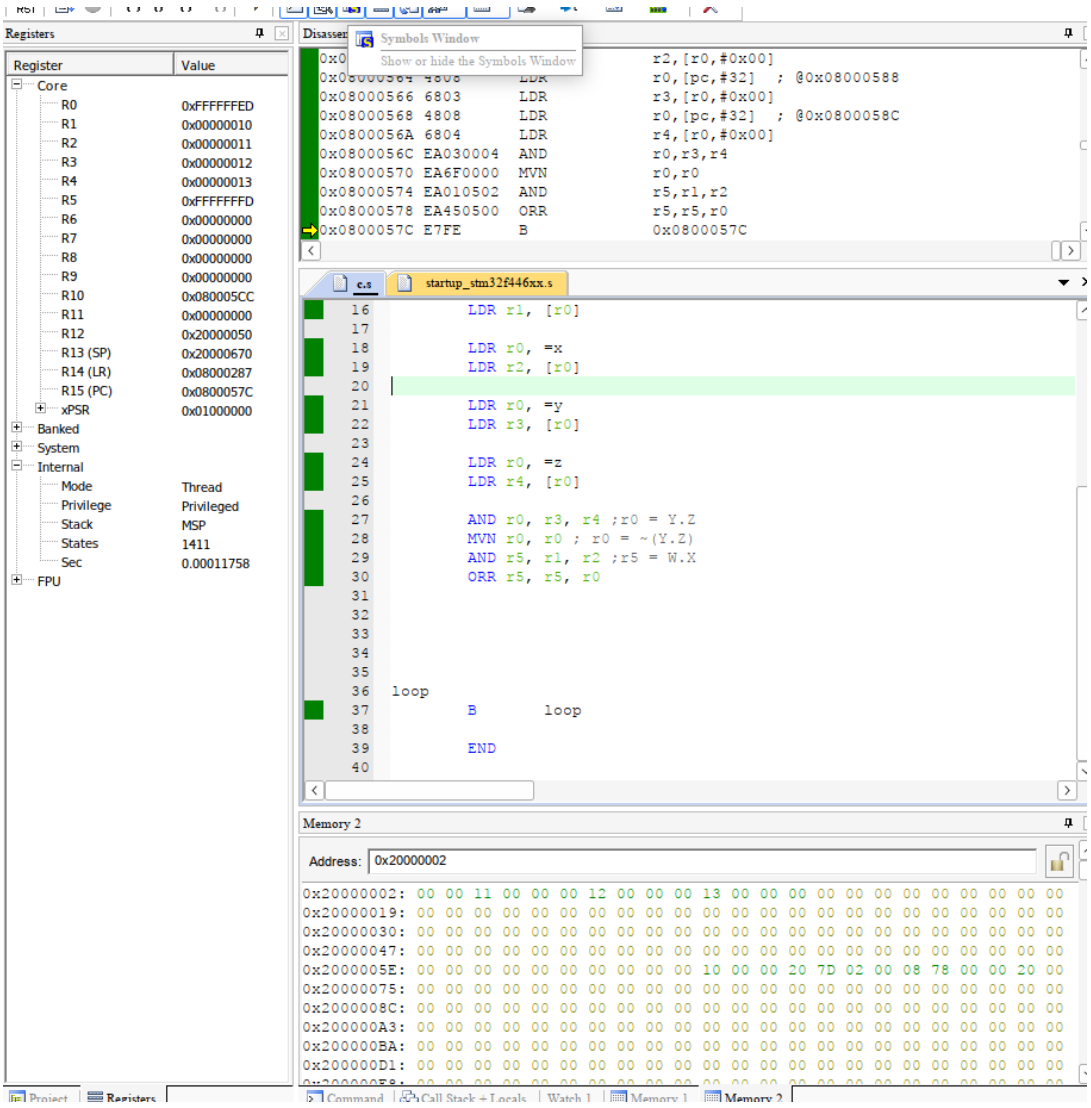

Figure 2: After executing

1.2 Task 2

Suppose we have three words P, Q and R. We are going to apply logical operations to subfields (bit fields) of these registers. We'll use 16-bit arithmetic for simplicity. Suppose that we have three 6-bit bit fields in P, Q, and R as illustrated below. The bit fields are in green and are not in the same position in each word. A bit field is a consecutive sequence of bits that forms a logic entity. Often they are data fields packed in a register, or they may be graphical elements in a display (a row of pixels). However, the following example demonstrates the type of operation you may have to perform on bits. P=p15p14 p13 p12 p11 p10 p9 p8 p7 p6 p5 p4 p3 p2 p1 p0 = 0010000011110010 Q = q15 q14 q13 q12 q11 q10 q9 q8 q7 q6 q5 q4 q3 q2 q1 q0 = 0011000011110000 R = r15 r14 r13 r12 r11 r10 r9 r8 r7 r6 r5 r4 r3 r2 r1 r0 = 1100010011111000 Write assembly language to calculate F = (P + Q xor R).111110 using the three 6- bit bit fields.

1.2.1 Screenshot that shows the state of the system after the code has been loaded.

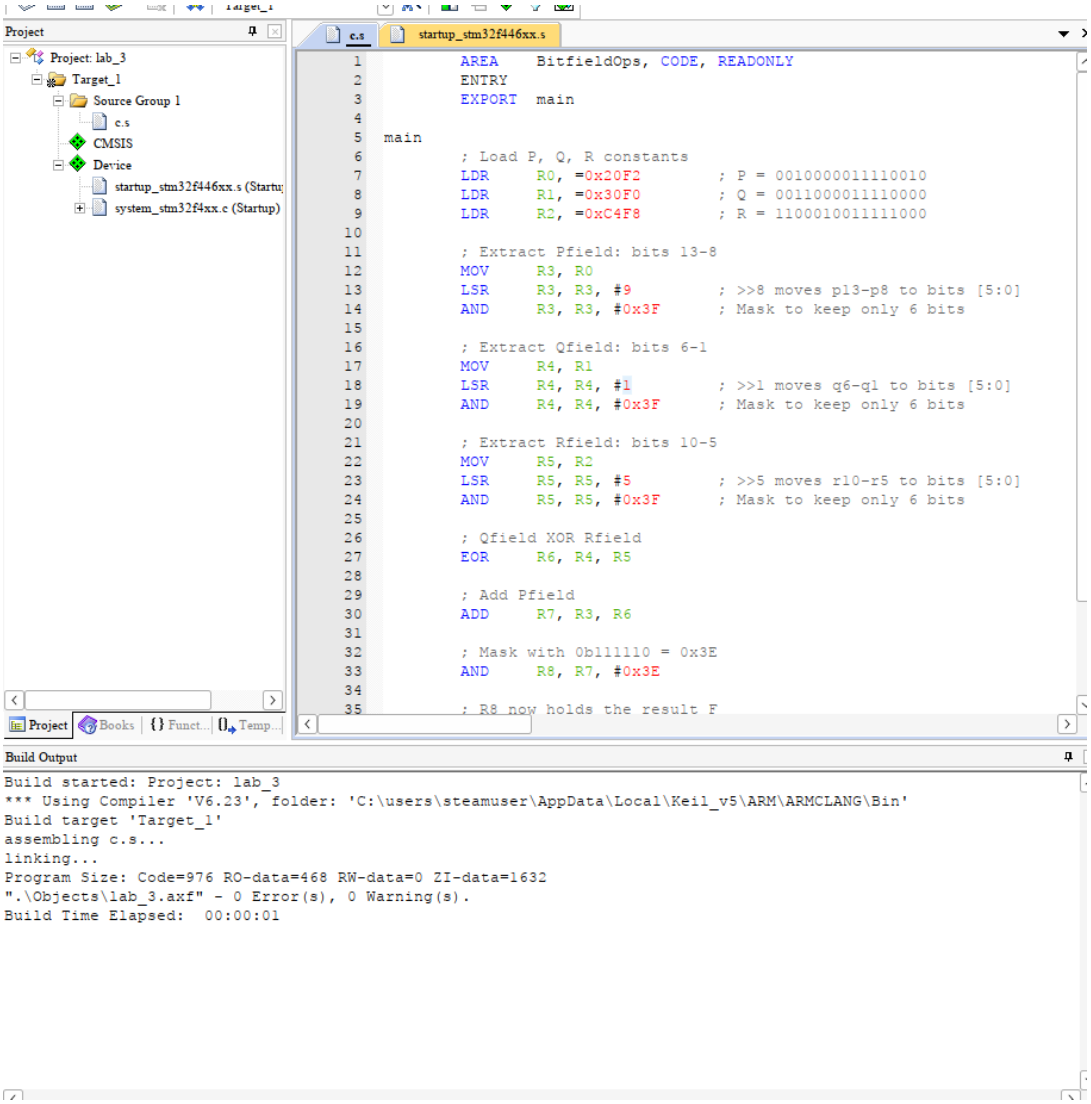

Figure 3: After build and debugging, this is the state

1.2.2 Screenshot that shows the situation after the code has been executed.

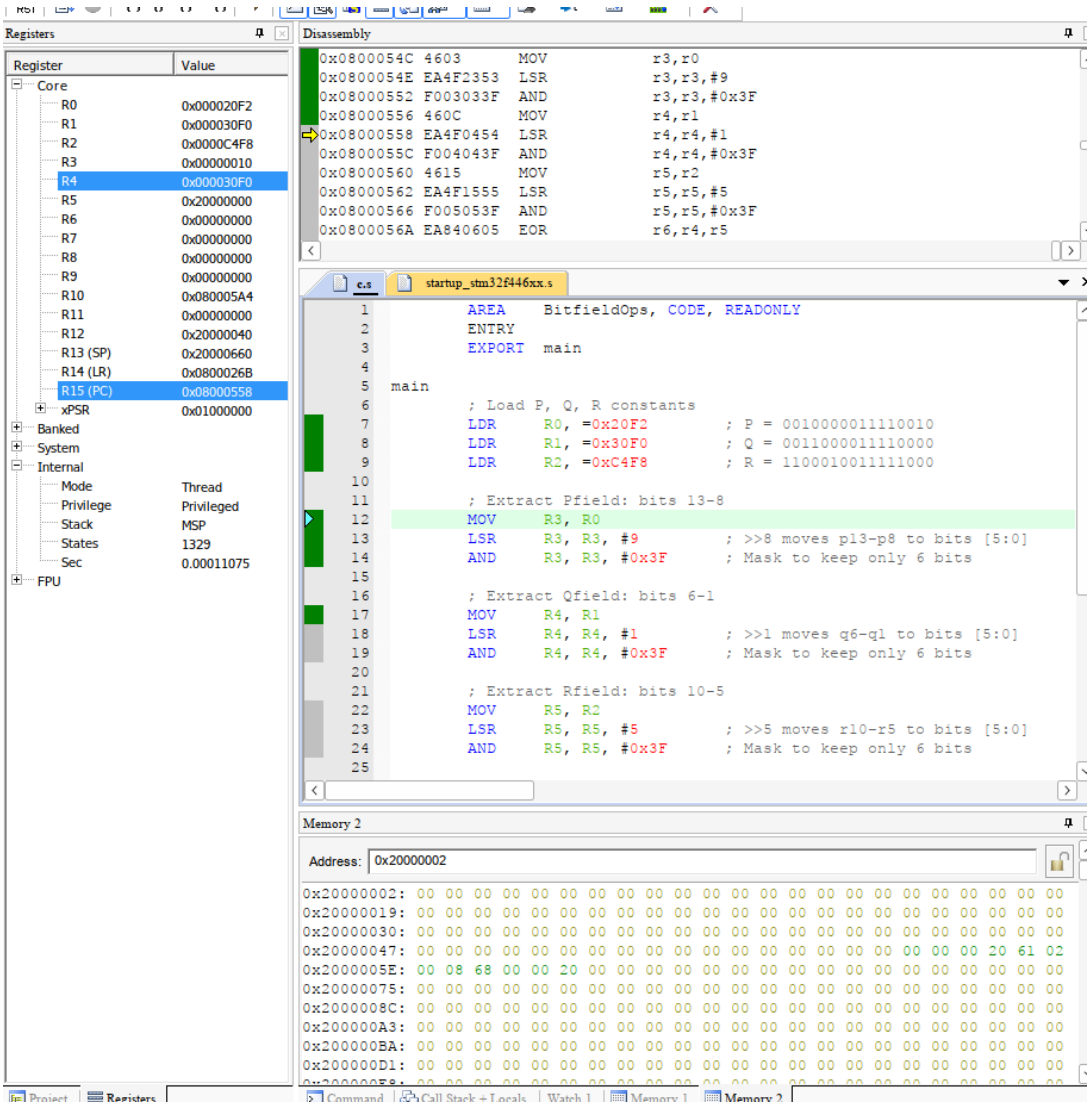

Figure 4: After executing

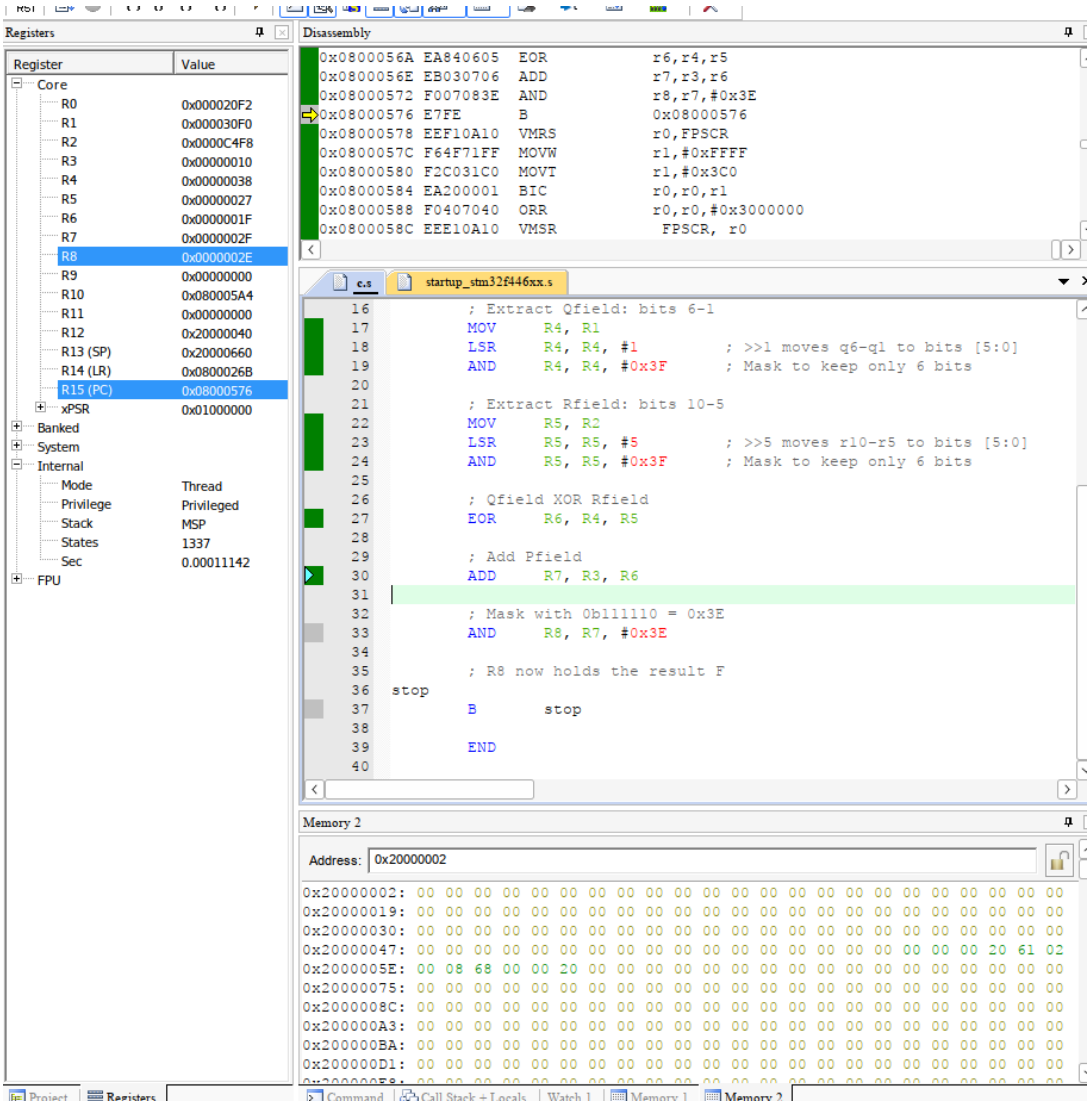

Figure 5: After executing

1.3 Task 3

Write an assembly language to find the one's complement of a number. Sample : Input: Number: C123

1.3.1 Screenshot that shows the state of the system after the code has been loaded.

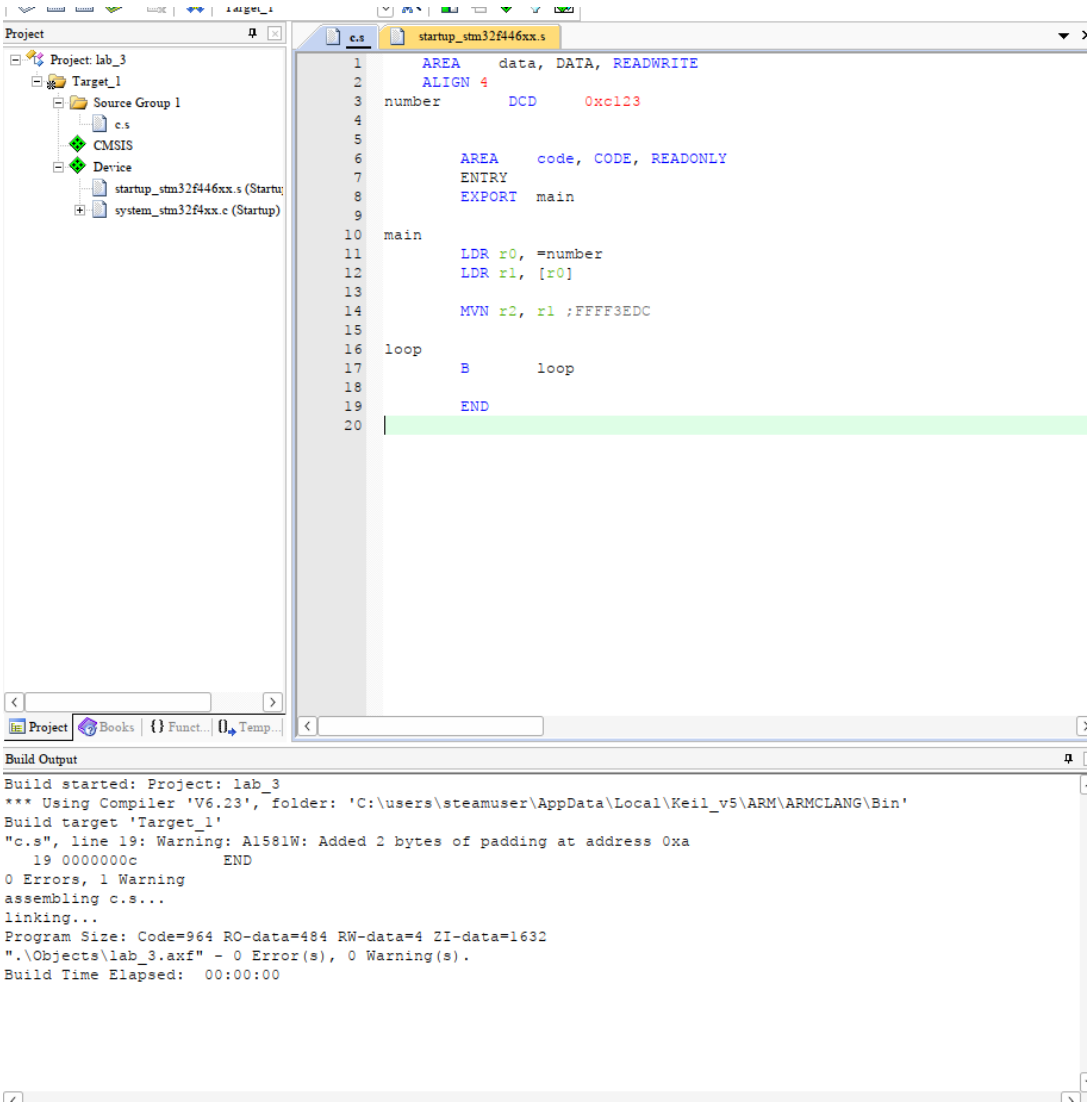

Figure 6: After build and debugging

1.3.2 Screenshot that shows the situation after the code has been executed.

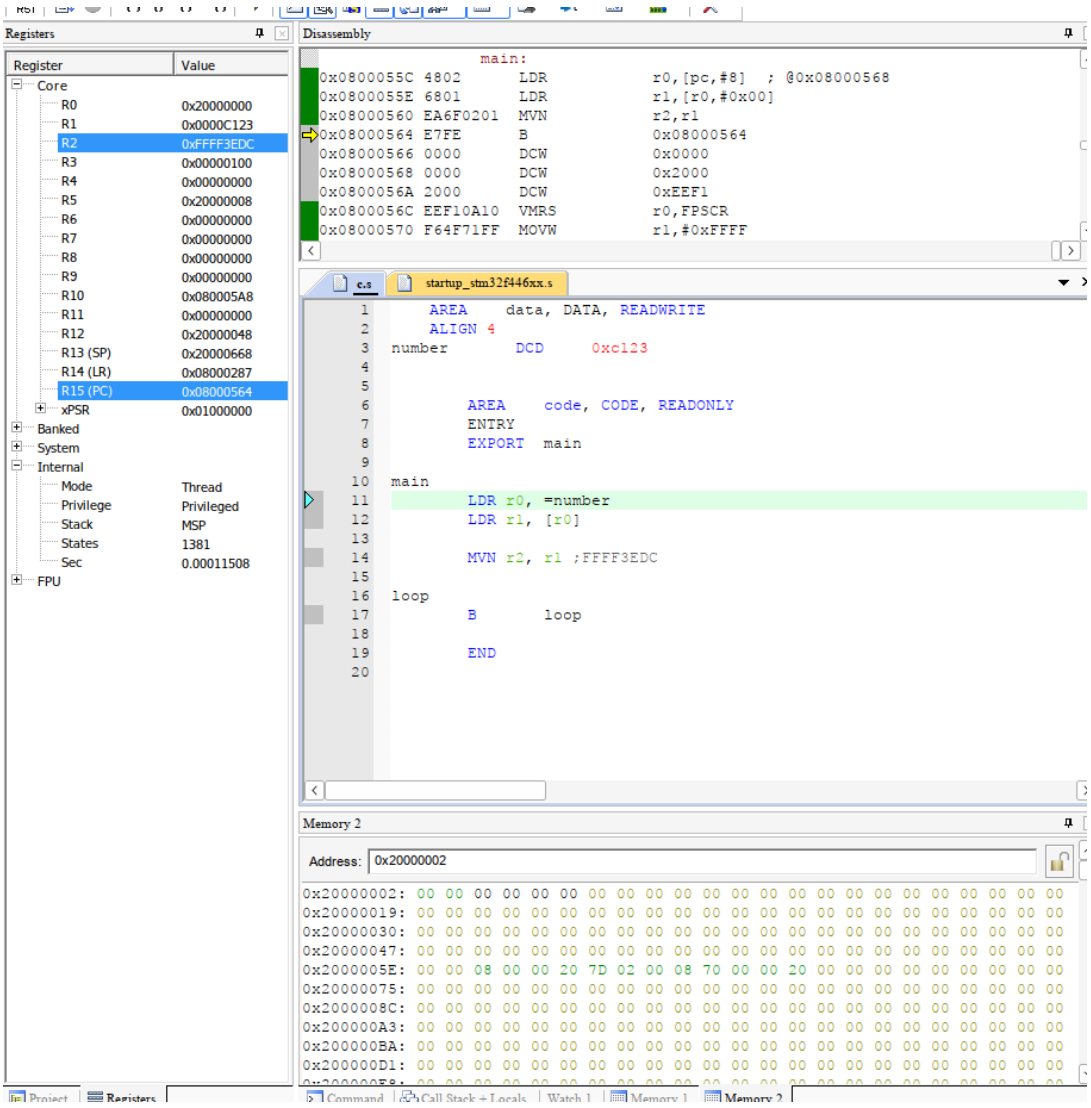

Figure 7: After executing

1.4 Task 4

Write an assembly language program to disassemble a byte into its high and low order nibbles. Divide the least significant byte of the 8-bit variable Value into two 4-bit nibbles and store one nibble in each byte of the 16-bit variable Result. The low-order four bits of the byte will be stored in the low-order four bits of the least significant byte of Result. The high-order four bits of the byte will be stored in the low-order four bits of the most significant byte of Result. Sample : Input: Number: 5F Output: Result: 050F

1.4.1 Screenshot that shows the state of the system after the code has been loaded.

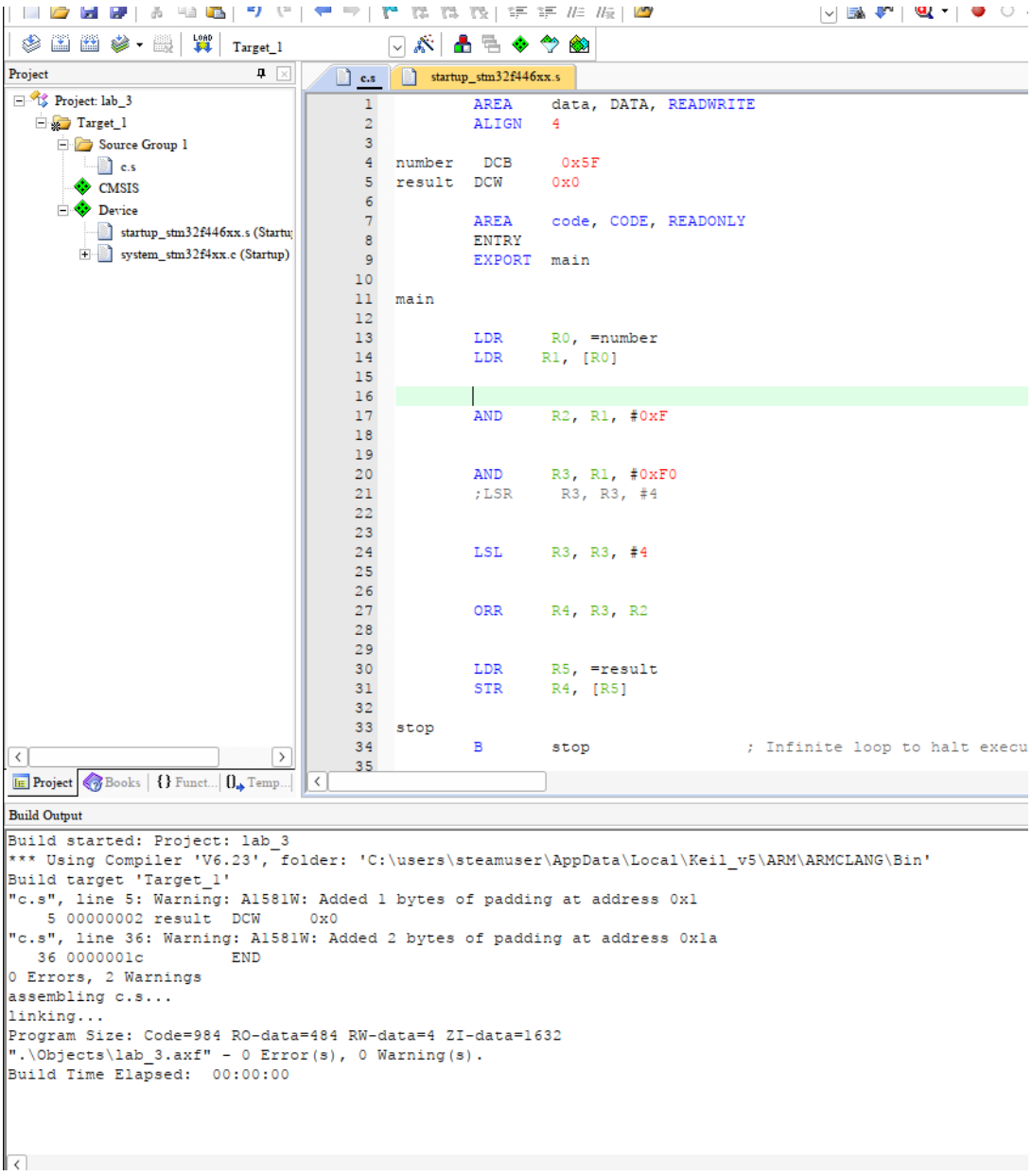

Figure 8: After build and debugging

1.4.2 Screenshot that shows the situation after the code has been executed.

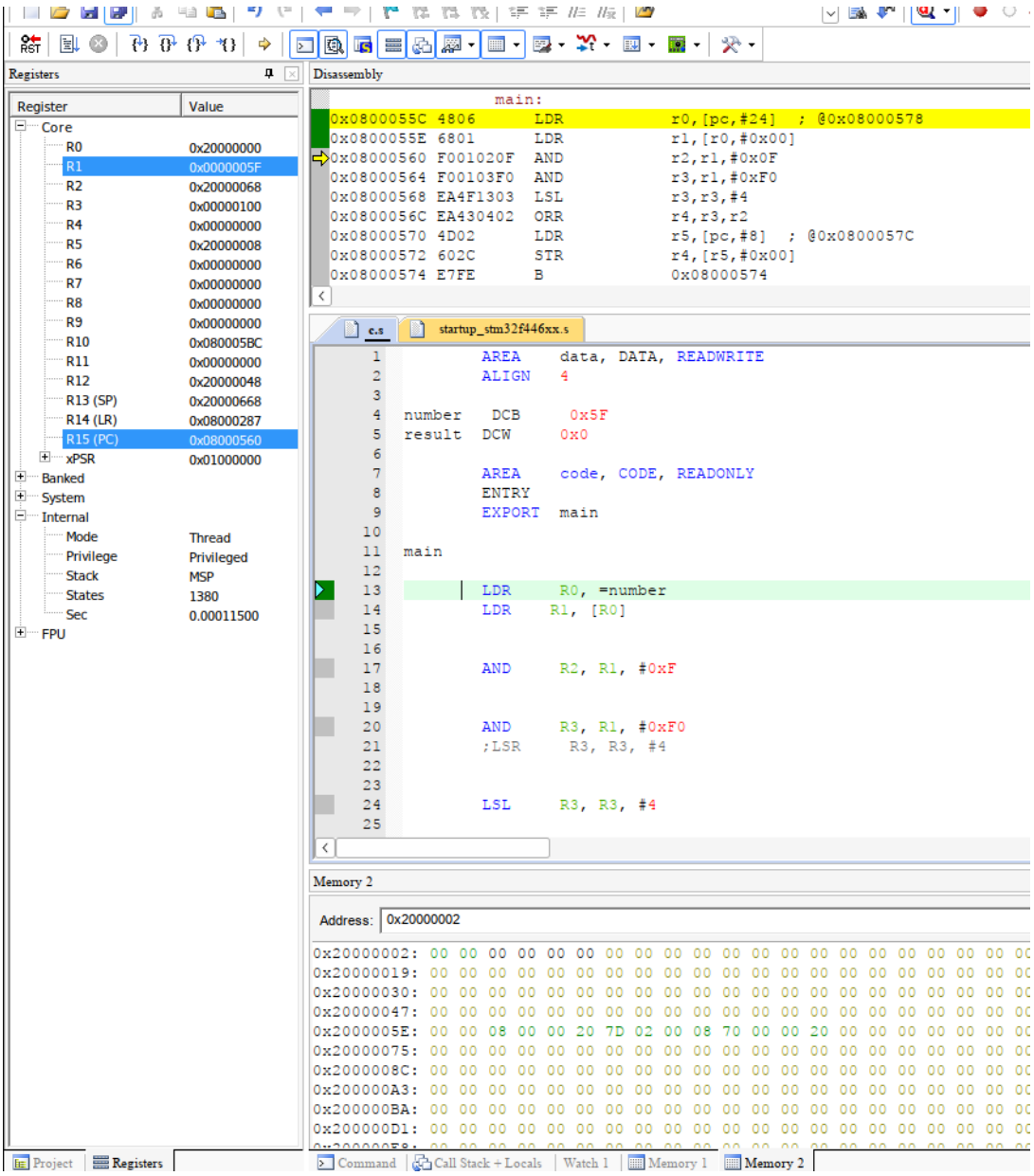

Figure 9: After executing

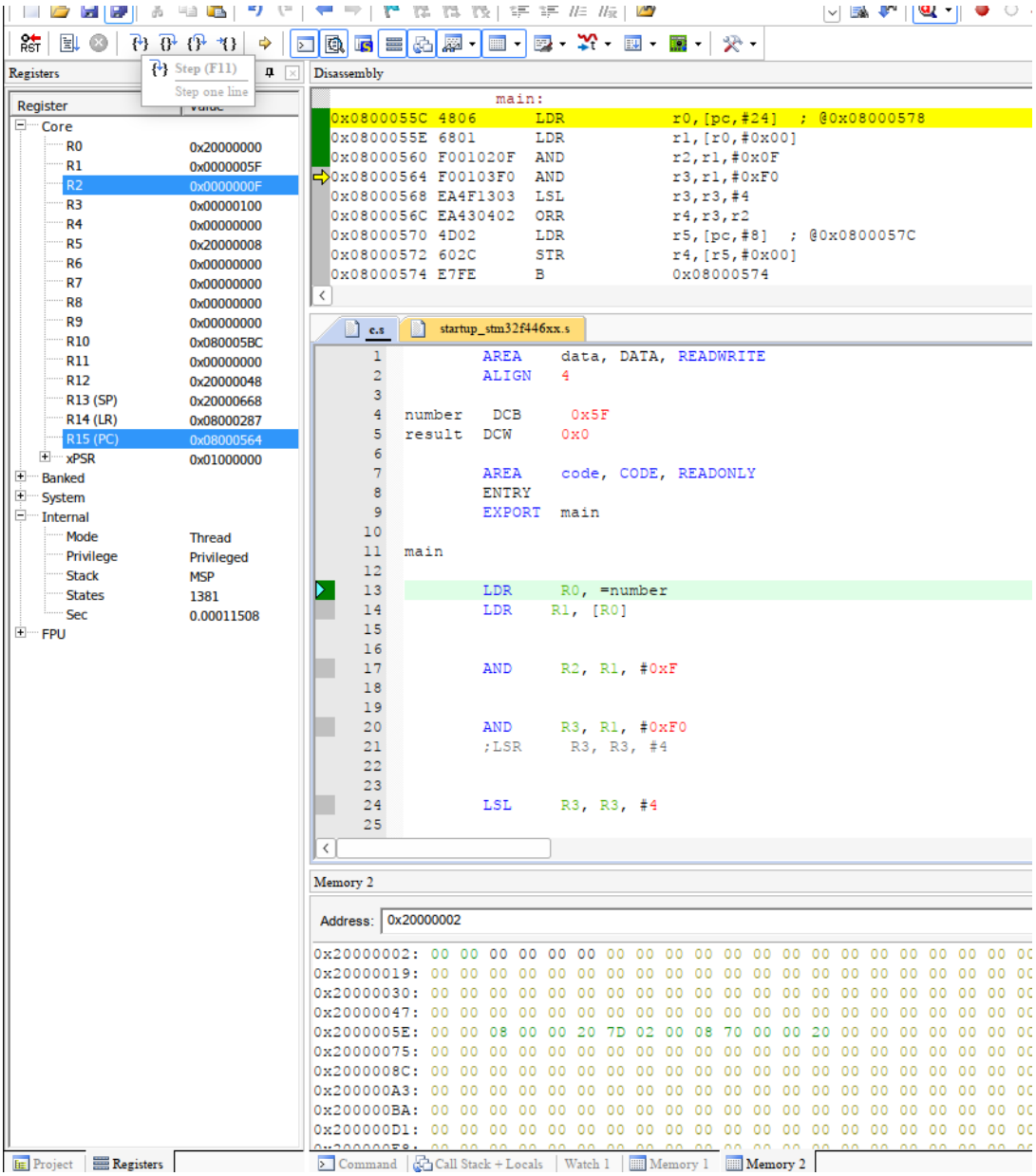

Figure 10: After executing

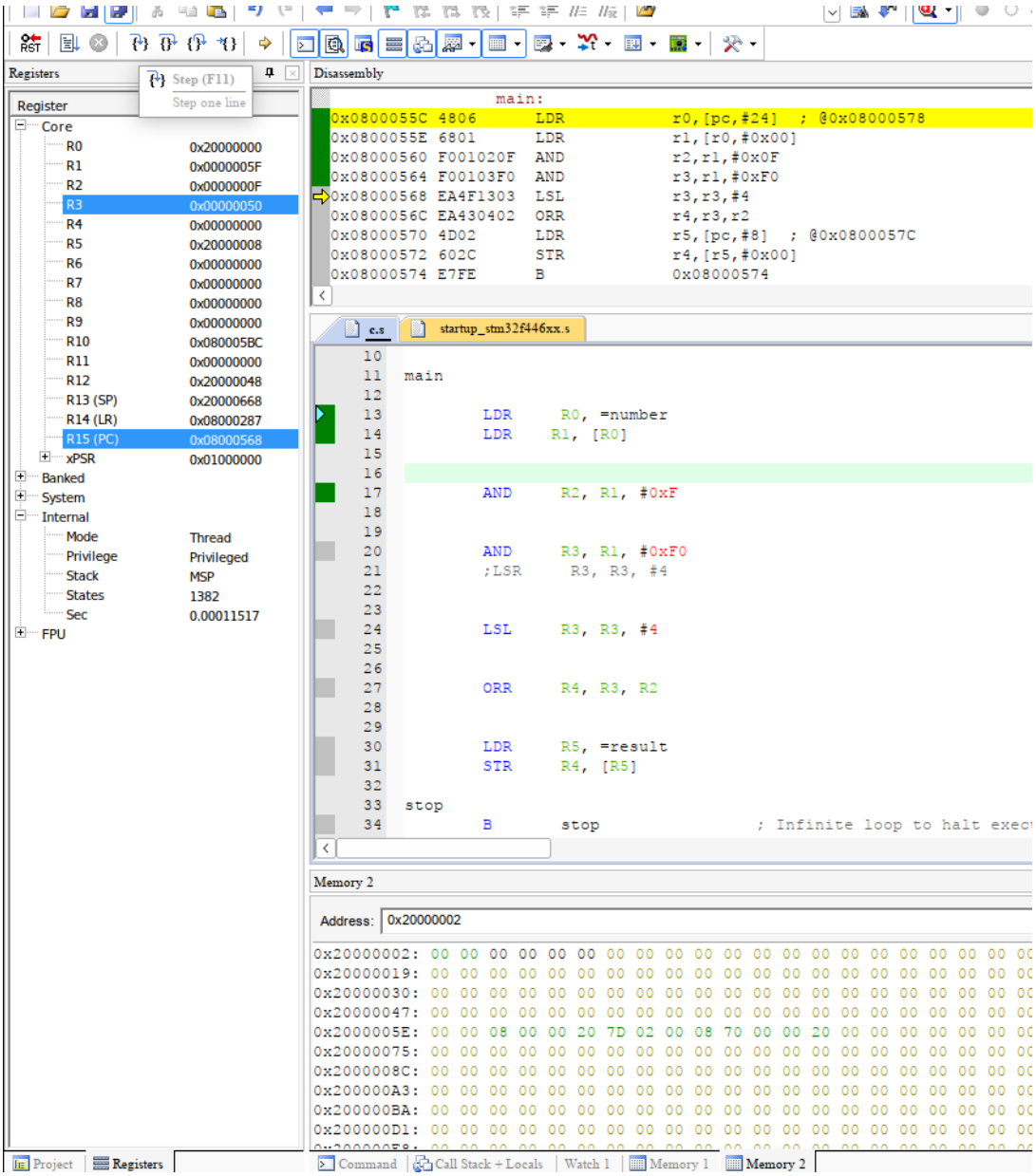

Figure 11: After executing

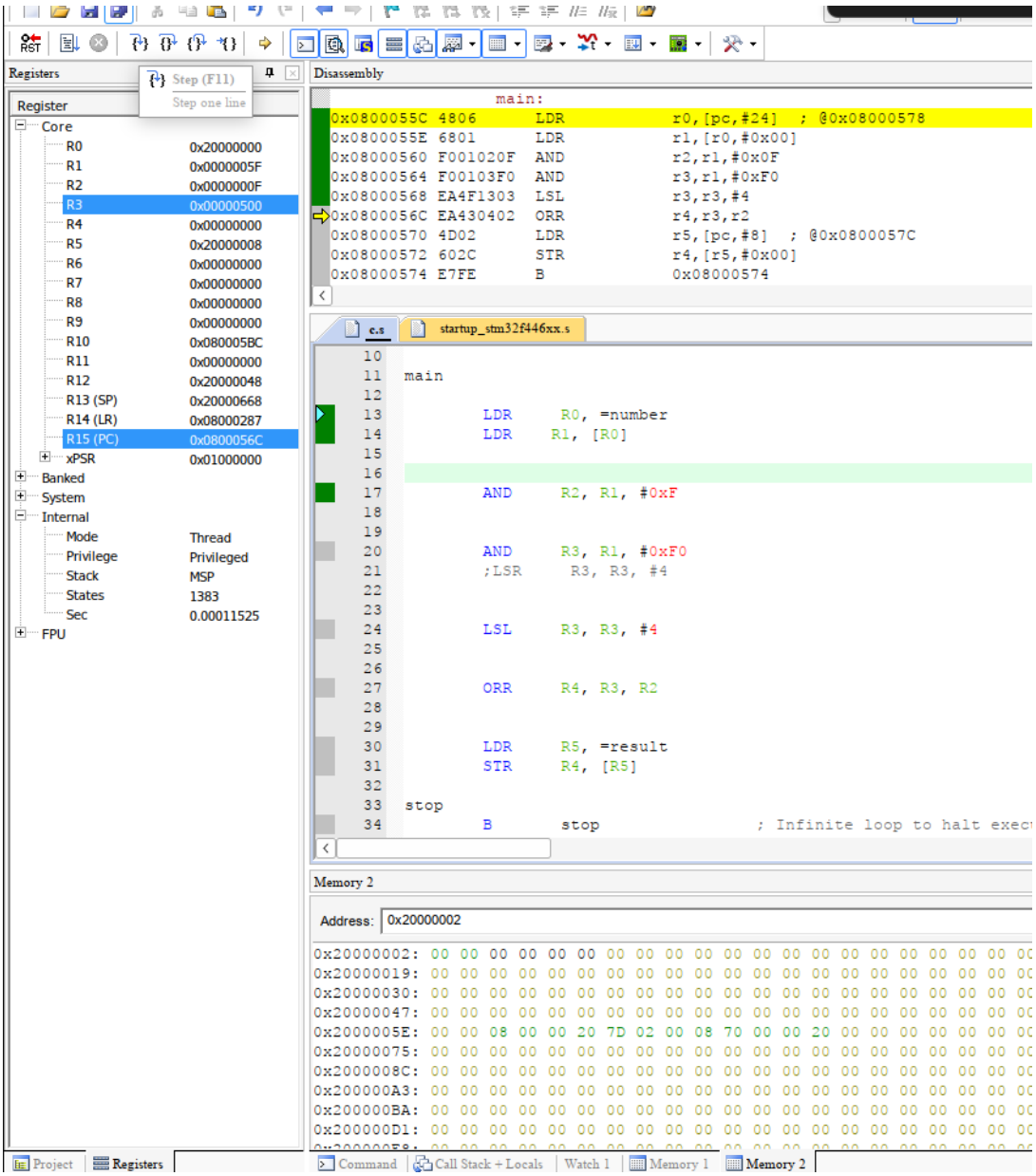

Figure 12: After executing

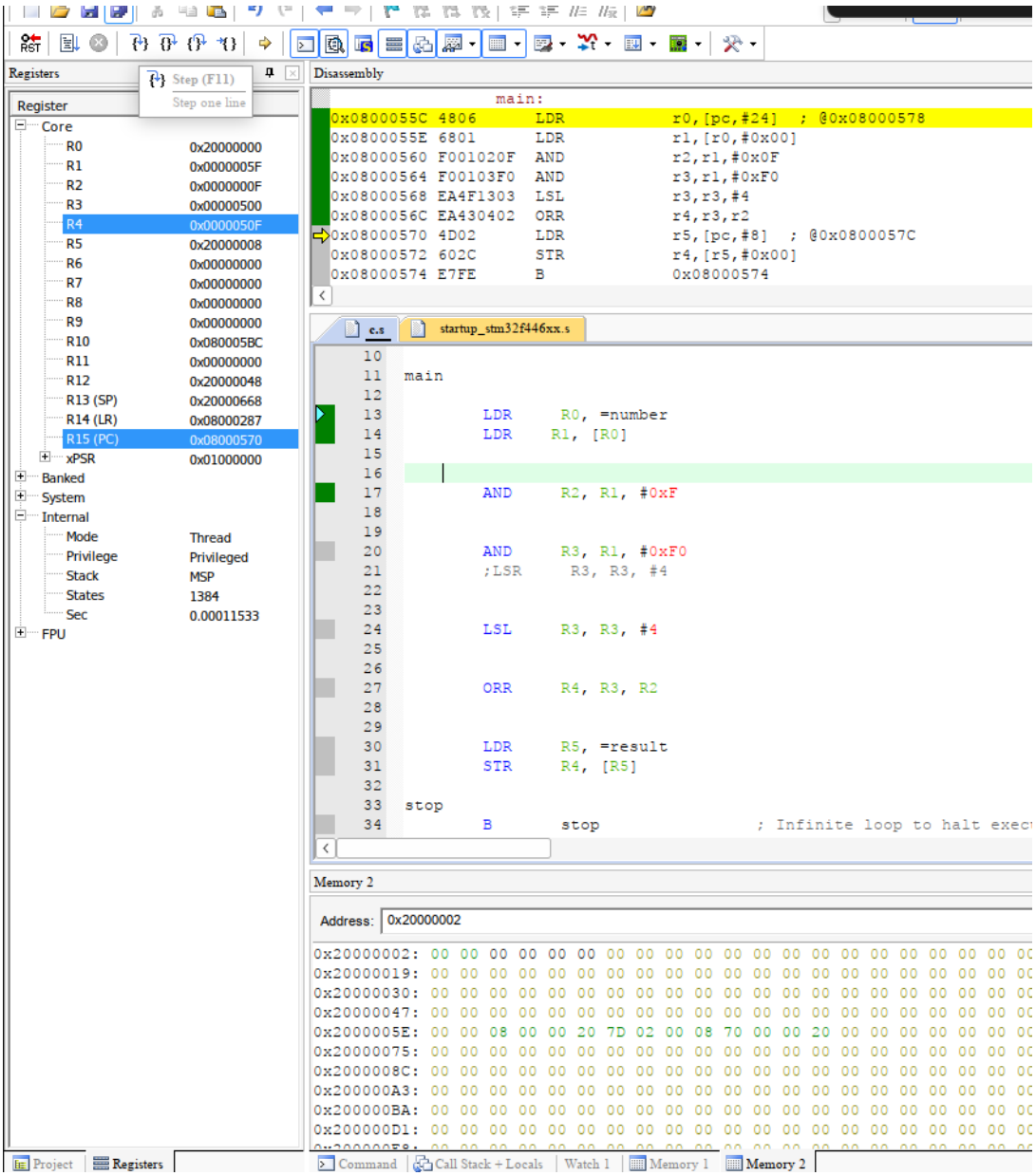

Figure 13: After executing

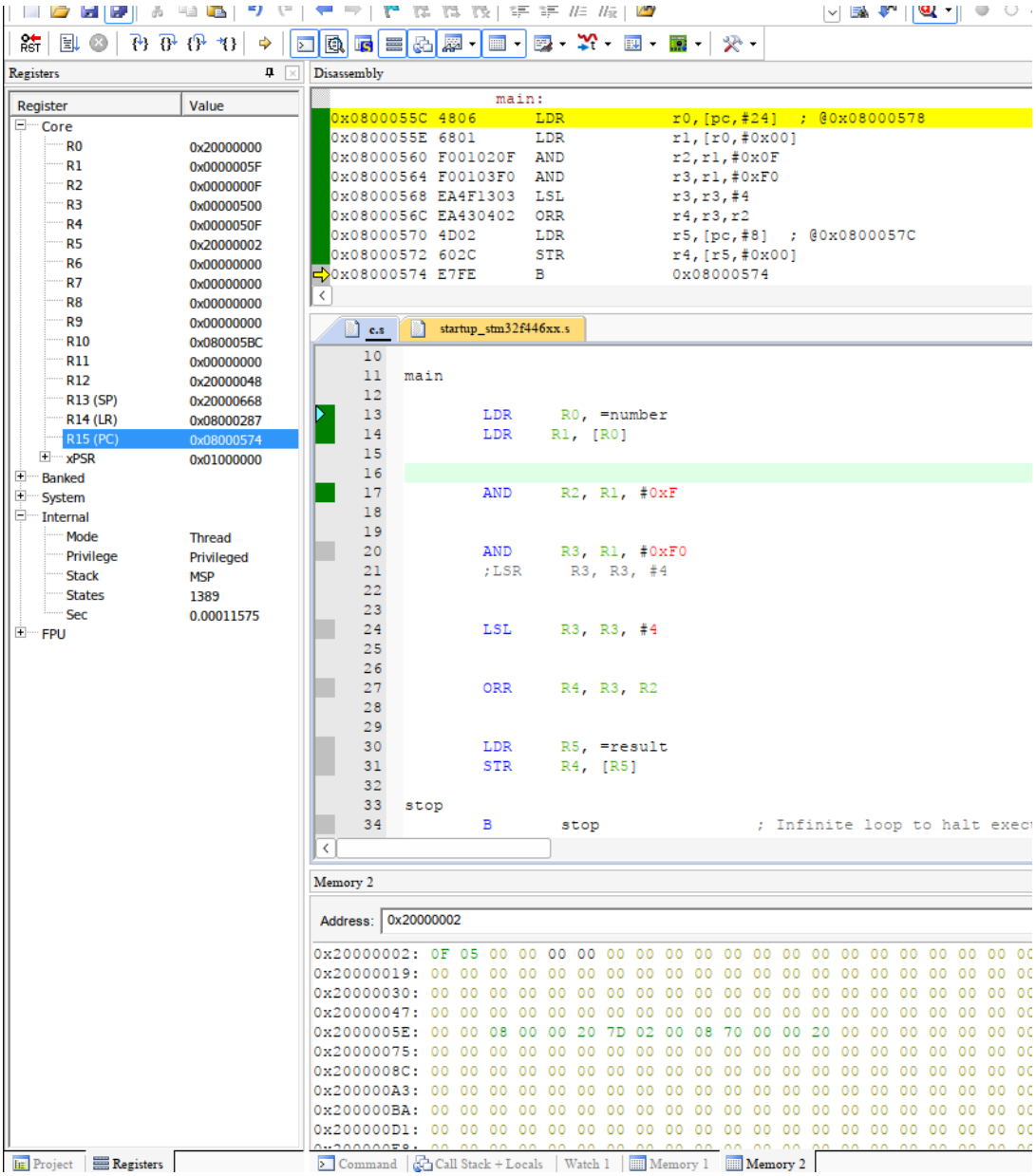

Figure 14: After executing