Name : Punyaja Mishra

Student Id : 0660001

A

Assume a color display using 10 bits for each of the primary colors (red, green, blue) per pixel and a frame size of **3440x1440 (widescreen)**

**a.** What is the minimum size in bytes of the frame buffer to store a frame?

Answer :

3 colors, each using 10 bits

Thus, one pixel uses 3X10 bits = 30 bits  
3 colors and each color uses 1 byte. Also the frame consists of   
3440X1440=4,953,600 pixels  
Therefore, minimum size of frame is  
30X4953600 = 148,608,000 bits

=18,576,000 bytes

**b.** How long would it take, at a minimum, for the frame to be sent over a **400**

Mbit/s network? Answer :   
For the frame to be sent at 400 Mbit/s  
speed : 400X10^6 bits = 4X10^8bit/s  
time = 148608000 bits / (10^8X4)  
 = 37152000/(10^3X10^5)  
 = 0.37152 s 🡺This much time.

**c.** What bandwidth do you need (at a minimum) to transmit this data at **144** frames per second?  
Answer :  
10 bits for each color per pixel  
3440X1440 frame size  
transmit data at 144 frames per second  
10X3440X1440X144 = 7133184000 bits/second  
 = 7133.184 MB/second 🡺The minimum bandwidth

# B

**1**Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 4.4GHz clock rate and a CPI of 1.5. P2 has a 3.3GHzclock rate and a CPI of 1.0. P3 has a 4.1 GHz clock rate and has a CPI of 2.2.

**a.** Which processor has the highest performance expressed in instructions per secondAnswer :  
P1 : 4.4GHz/1.5 = 2.93X10^9 instructions per sec  
P2 : 43.3GHz/1.0 = 3.3X10^9 instructions per sec  
P3 : 4.1GHz/2.2 = 1.86X10^9 instructions per sec  
Therefore, P2 has the highest performance

**b.** If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.   
Answer :

Cycles  
P1 : 4.4GHzX10 = 4.4X10^10 cycles  
P2 : 3.3GHzX10 = 3.3X10^10 cycles  
P3 : 4.1GHzX10 = 4.1X10^10 cycles

Number of Instructions  
P1 : 4.4GHz\*10/1.5 = 2.93X10^10 instructions per sec  
P2 : 43.3GHz\*10/1.0 = 3.3X10^10 instructions per sec  
P3 : 4.1GHz\*10/2.2 = 1.86X10^10 instructions per sec

**c.** We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?  
Answer :  
Execution time = (number of instructions X CPI)/Clock Rate

So, if we want to reduce execution time by 30%, CPI increase by 20%, we have

Execution time X 0.7 = (No. of instructions X CPI X 1.2)/ new clock rate

New clock rate = clock rate X (1.2/0.7)  
 = 1.71 X clock rate

P1 : 4.4GHzX1.71 = 7.524 GHz  
P2 : 3.3GHzX1.71 = 5.643 GHz  
P3 : 4.1GHzX1.71 = 7.011 GHz

# C

Section 1.10 cites as a pitfall the utilization of a subset of the performance equation as a performance metric. To illustrate this, consider the following two processors. P1 has a clock rate of 4 GHz, average CPI of 0.9, and requires the execution of 5.0E9 instructions. P2 has a clock rate of 3 GHz, an average CPI of 0.75, and requires the execution of 1.0E9 instructions.  
**1.12.1** One usual fallacy is to consider the computer with the largest clock rate as having the largest performance. Check if this is true for P1 and P2  
Answer :

|  |  |  |  |
| --- | --- | --- | --- |
| Processor | Clock Rate | CPI | Execution of instruction |
| P1 | 4GHz | 0.9 | 5X10^9 |
| P2 | 3GHz | 0.75 | 1.0E9X10^9 |

PP1 /PP2 = ETP2/ETP1 = { [1 X 10^9 X 0.75] / [3 X 10^9] } / { 5 X 10^9 X 0.9/ [4 X 10^9]}

= (0.75 X 1/ 5 X 0.9) X (4/3)

= 3 / 13.5  
 = 0.22  
There fore, P1 has less performance than P2 hence it’s not true for P1 and P2.

**1.12.2** Another fallacy is to consider that the processor executing the largest number of instructions will need a larger CPU time. Considering that processor P1 is executing a sequence of 1.0E9 instructions and that the CPI of processors P1 and P2 do not change, determine the number of instructions that P2 can execute in the same time that P1 needs to execute 1.0E9 instructions.  
Answer :  
ETP1 = (ICP1 X CPIP1)/CRPI   
 = [1 X 10^9 instructions X 0.9 cycles/instructions] / [4 X 10^9 cycles / sec]

= 0.25 second

ICP2 = [ETP2 X CRP2]]/[CPIP2]

= [0.225 X 3 X 10^9] / 0.75

= 0.9 X 10^9 instructions

Therefore, the number of instructions needed by P2 to execute in same time as P1.

**1.12.3** A common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors and consider that the processor with the largest MIPS has the largest performance.

Check if this is true for P1 and P2.  
Answer :

MIPS = clock rate X 10^-6 / CPI

MIPS(P1) = 4 X 10^9 X 10^-6 / 0.9 = 4.44 X 10^3

MIPS(P2) = 3 X 10^9 X 10^-6 / 0.75 = 4 X 10^3

MIPS(P1) > MIPS(P2) but  
performance(P1) < performance(P2)

Therefore, not true for P1 and P2

**1.12.4** Another common performance figure is MFLOPS (millions of floating-point operations per second), defined as MFLOPS = No. FP operations / (execution time × 1E6) but this figure has the same problems as MIPS. Assume that 40% of the instructions executed on both P1 and P2 are floating-point instructions. Find the MFLOPS figures for the programs.  
Answer :  
MFLOPS = number of FP operations X 10^-6 / ET

MFLOPS(P1) = 0.4 X 5 X 10^9 X 10^-6 / 1.125 = 1.78 X 10^3

MFLOPS(P2) = 0.4 X 10^9 X 10^-6 / 0.25 = 1.6 X 10^3

MFLOPS(P1) > MFLOPS(P2) but  
performance(P1) < performance(P2)

Therefore, not true for P1 and P2

**1.12.Sri/Alaadin -** Explain in ~100-200 words what this question is trying to illustrate, and what metric(s) you should use to compare CPU performance given there are obvious problems with clock rate, IPC, IPS etc.

Answer : The question is trying to illustrate that there are few fallacies consistent when it comes to performance metric. Since clock rate denotes the CPU speed, It is believed that the larger the clock rate of the processor means larger the performance of that processor but in the question clearly that wasn’t the case. Thus, clock rate alone isn’t only the factor in determining the performance of the CPU. It depends on other factors like CPI and execution of instruction. Same way, MIPS or MFLOPS alone can’t either be used to determine the performance of the processor since they are calculated using clock rate over CPI and hence not independent.

Best way to determine the speed of CPU is using clock rate(but not only clock rate), multi processors that can run independently or co-operatively, and front side bus and CPU together.

# D

**1.15 Sri/Alaadin Note: Do this problem using a spreadsheet** When a program is adapted to run on multiple processors in a multiprocessor system, the execution time on each processor is comprised of computing time and the overhead time required for locked critical sections and/or to send data from one processor to another.

Assume a program requires t = 100 s of execution time on one processor. When run *p* processors, each processor requires t/p s, as well as an additional 4 s of overhead, irrespective of the number of processors. Compute the per-processor execution time for 2, 4, 6, 8, 12, 16, 24, 28, 32, 56, 64, 224, and 448 processors. For each case, list the corresponding speedup relative to a single processor and the ratio between actual speedup versus ideal speedup (speedup if there was no overhead).

Answer : In the Excel Spread sheet attached with the submission in the zip file.

# E

**2.7**  Show how the value**s** 0x134 433C **AND** -23002020 **(-15EFBA4)**Answer : Conversion of -23002020 to hex is -15EFBA4

**0X134 433C**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte Address** | **0** | **1** | **2** | **3** |  |
| **Hex** | **C** | **13** | **33** | **44** | **Big Endian** |
|  |  |  |  |  |  |
| **Hex** | **44** | **33** | **13** | **C** | **Low Endian** |

**-15EFBA4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Byte Address** | **0** | **1** | **2** | **3** |  |
| **Hex** | **EF** | **BA** | **-15** | **4** | **Big Endian** |
|  |  |  |  |  |  |
| **Hex** | **4** | **-15** | **BA** | **EF** | **Low Endian** |

# F

**2.8** Translate 0x134 433C into decimal. Answer :  
To convert into decimal, we multiply each digit by 16 to the power of the place the digit is at, starting 0 from the right.  
C=12  
(1X16^6)+ (3X16^5)+ (4X16^4)+ (4X16^3)+ (3X16^2)+ (3X16^1)+ (CX16^0) = 20,202,300

# G

Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 700 million arithmetic instructions, 200 million load/store instructions, 80 million branch instructions.

1. Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, and the cost of increasing the clock cycle time by only 10%. Is this a good design choice? Why?  
   Answer :  
   CPU Time = clock cycles X clock cycle time  
   Now, number of clock cycles in the beginning is (in millions) :  
   Number of instructions X CPI   
   = 700 X 1 + 200 X 10 + 80 X 3  
   = 2940   
   Therefore, the CPU time in the beginning is   
   CPU Time A = 2940 X 10^6 X clock cycle time A  
   After decreasing the number of arithmetic instructions by 25% , we have 0.75X700=525 million arithmetic instructions.   
   The number of clock cycles is now (in millions)   
   525 X 1 + 200 X 10 + 80 X 3  
   = 2765  
   So, the CPU time now is   
   CPU time B = 2765 X 10^6 X clock cycle time B  
   = 2765 X 10^6 X 1.1 X clock cycle time A   
   = 3041.5 X clock cycle time A X 10^6  
   So, CPU time increases and hence proving it is not good so not design choice
2. Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times?  
   Answer :  
   We find a way to double performance of arithmetic instructions. So, CPI becomes ½ = 0.5  
   700 X 0.5 + 200 X 10 + 80 X 3 = 2590  
   So, CPU time now is,  
   CPU time C = 2590 X 10^6 X clock cycle time A  
   since, we do not change clock cycle time. Since,  
   CPU time C / CPU time A   
   = 2590/2940  
   =0.88095  
   Now, CPI of arithmetic instructions become 0.1. Thus the number of clock cycles is  
   700 X 0.1 + 200 X 10 + 80 X 3  
   =2310  
   So, CPU time now is  
   CPU time D = 2310 X 10^6 X clock cycle time A  
   CPU time D/CPU time A = 2310/2940 = 0.78571

# H: Sri/Alaadin General Theory Question (2.5 marks total)

Describe a way for an end user to compare CPU’s and decide which one to buy. Some possible things to consider: Performance for certain workloads, value, cost, likely future performance or whatever else you think might be important.

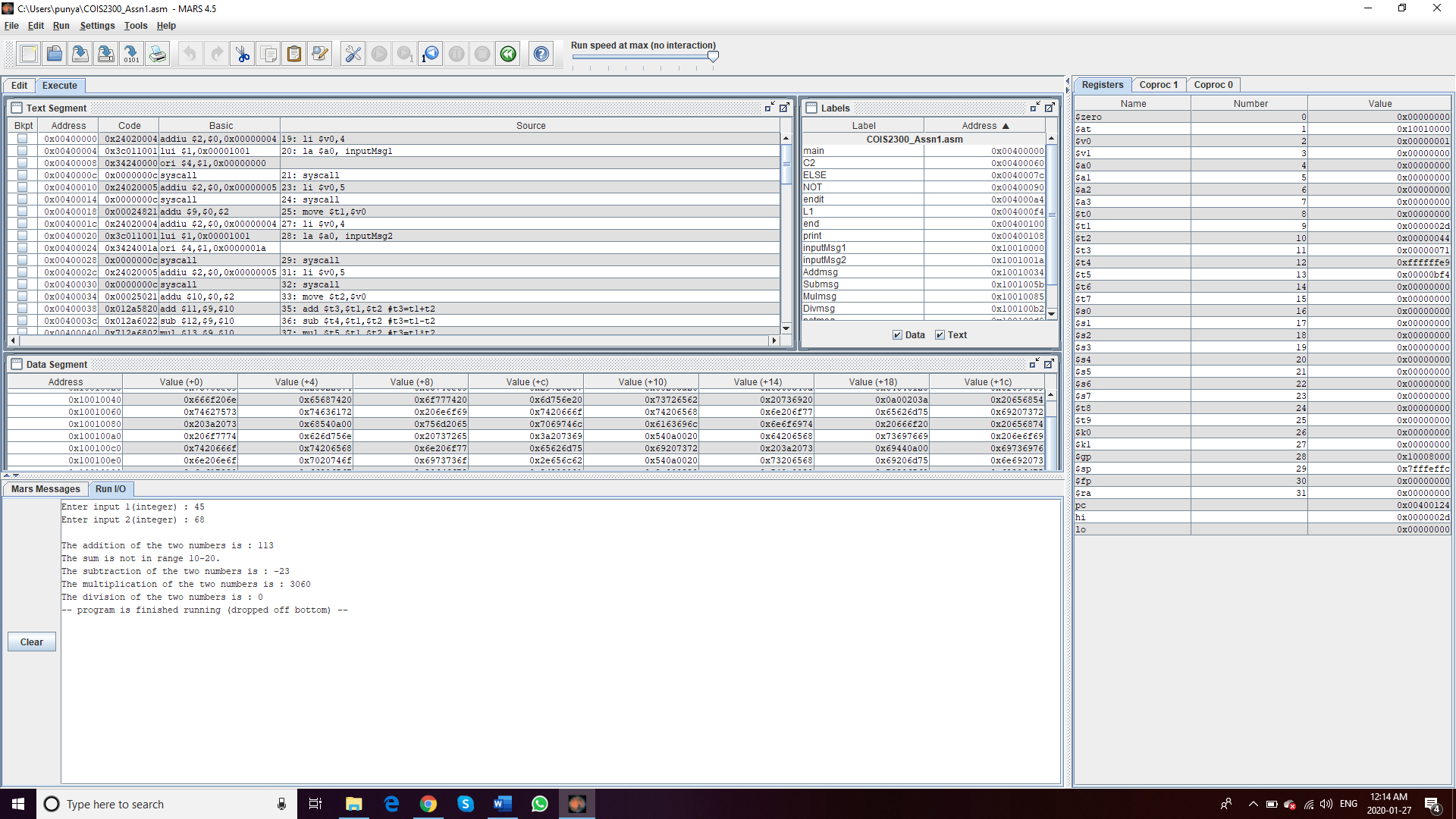
Answer : The CPU matters a lot when we consider the upgrading the existing system. Higher clock speeds and core counts can make a major difference in overall performance. This decision is made as to which one has the best processor that is available. We check how much cores there is and the cost as well is an important to consider. For mainstream users, current generation parts is a choice to make, like AMD Ryzen 3000 or Intel 9th Generation Core and Intel is doing better on the 1080p gaming on some titles and AMD handling tasks like video editing faster. But as long as the reason of buying CPU is not only gaming AMD delivers more cores and general performance plus 4.0 on its latest chips at better value. Clock speed however is more important than the core number as higher clock speeds translate to snappier performance in simple however, more cores is better for time-consuming workloads faster. The latest generation is the best option most of the time, even though the money and cost would be lot. A CPU, latest gen or not, should not have a weak storage, RAM and graphics. It does increase the budget but weak storage with a good core CPU is a bad decision just to save small money. While buying CPU, we can know about it buy it’s model number that mentions the generation in the first digit of the four number(ex- the 8 in core i7-8400). The rest of the numbers just mark various models in the line with higher generally being better with more cores and/or intel chips are “K” at the end of an Intel chip means it’s unlocked for overlocking. A processor’s on-board cache is used to speed up the access to the data and instructions between your CPU and RAM.

**TESTING DOCUMENTATION**

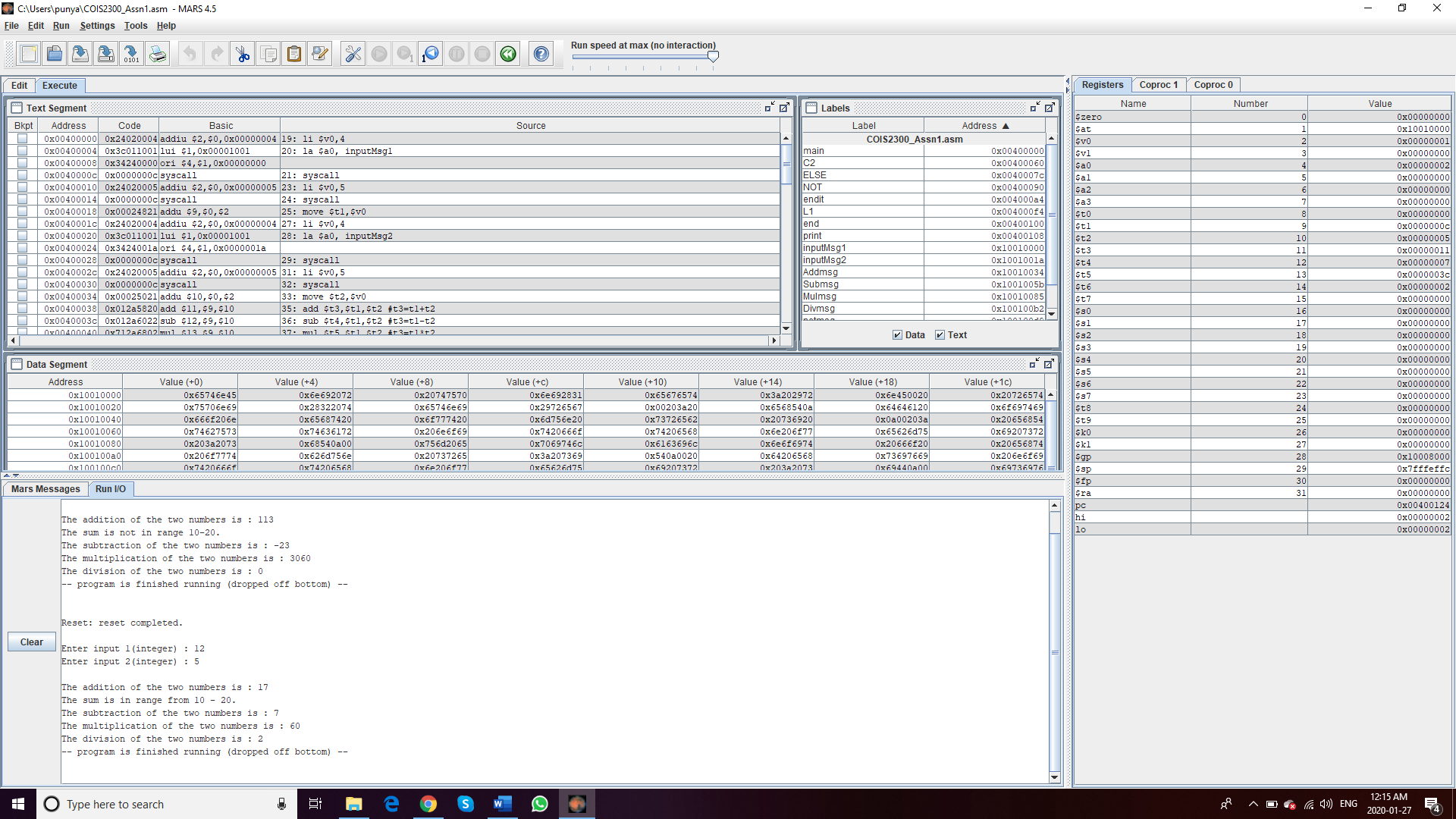
# I: Programming Problems

**Program 1:** Write a MIPS program that asks the user for two number inputs (we’ll call them *a* and *b*), calculate and print to screen: , if *b* is not 0, also calculate . Then check if a + b is in the range 10 … 20. Stick to integers, but allow positive or negative values (don’t check input types, and don’t test edge cases like very large integers)

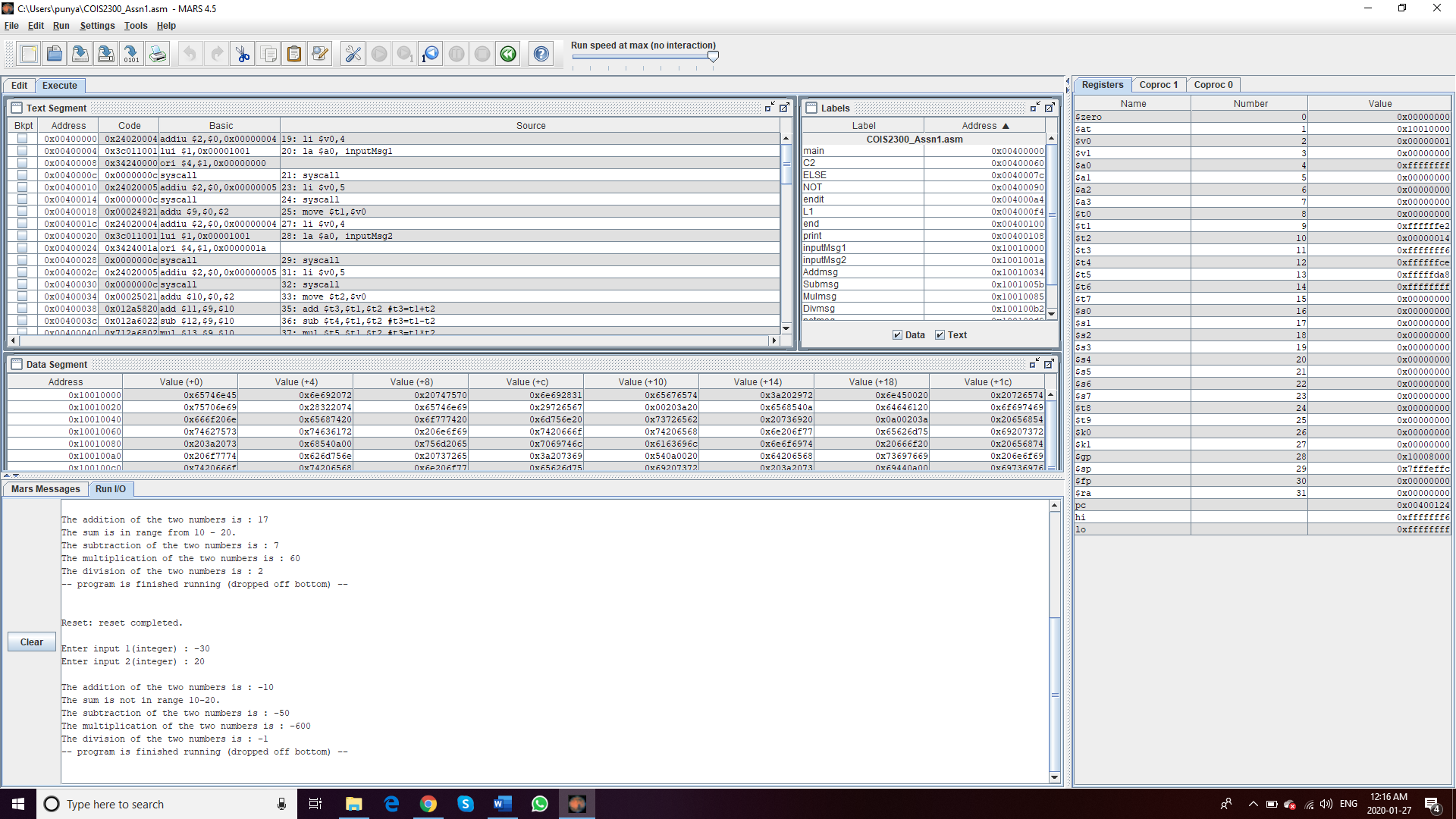
**Test 1 :** Checking for any two positive integers, if we get the correct addition, subtraction multiplication and division output. Their sum is not in the range if 10-20 **Input : 45, 68  
Output :**



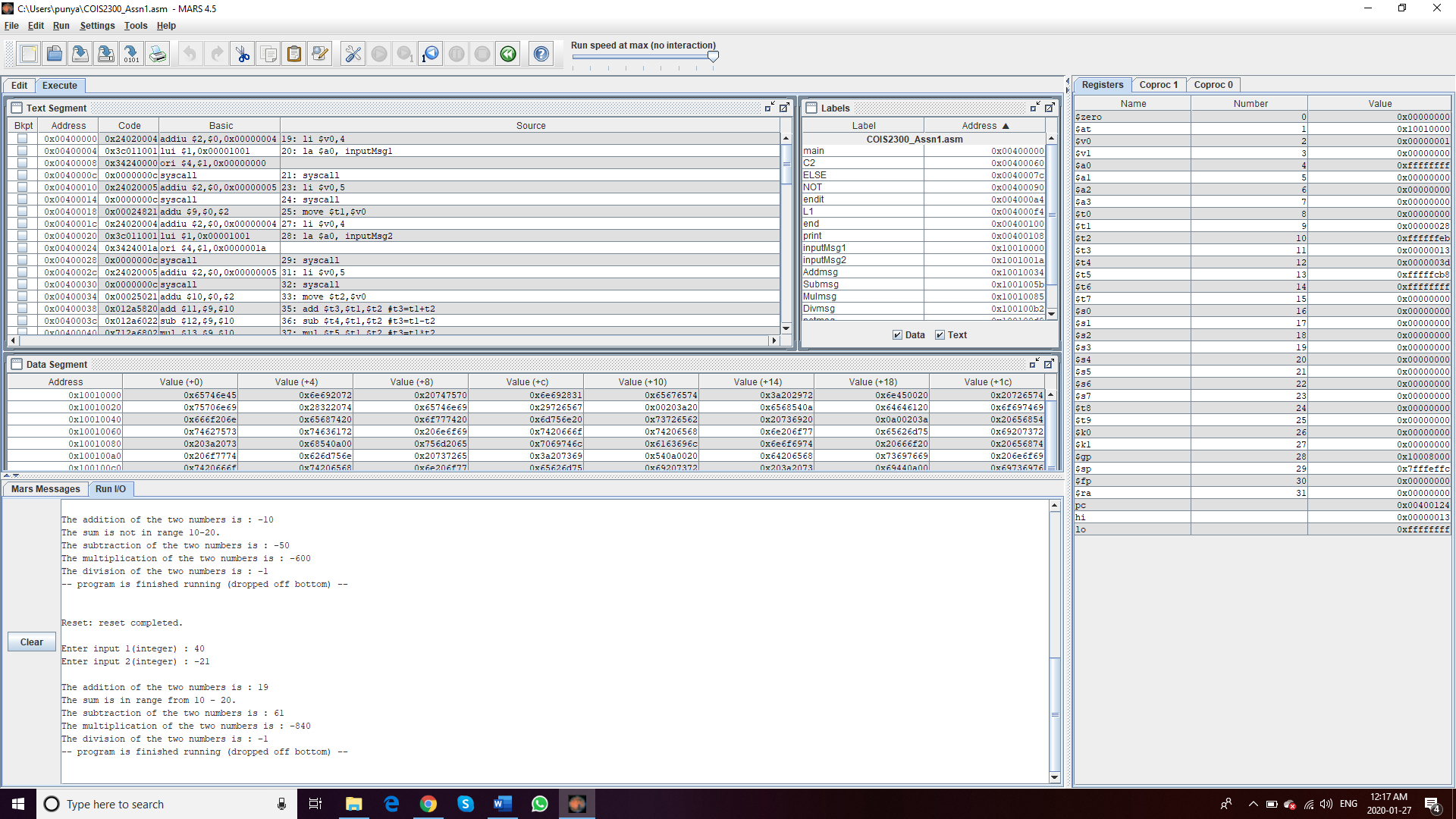
**Test 2:** Checking for two positive integers when their sum is in range of 10-20, to check if we get the correct output for all operands and the correct statement about the range. **Input :** 12, 5 **Output:**



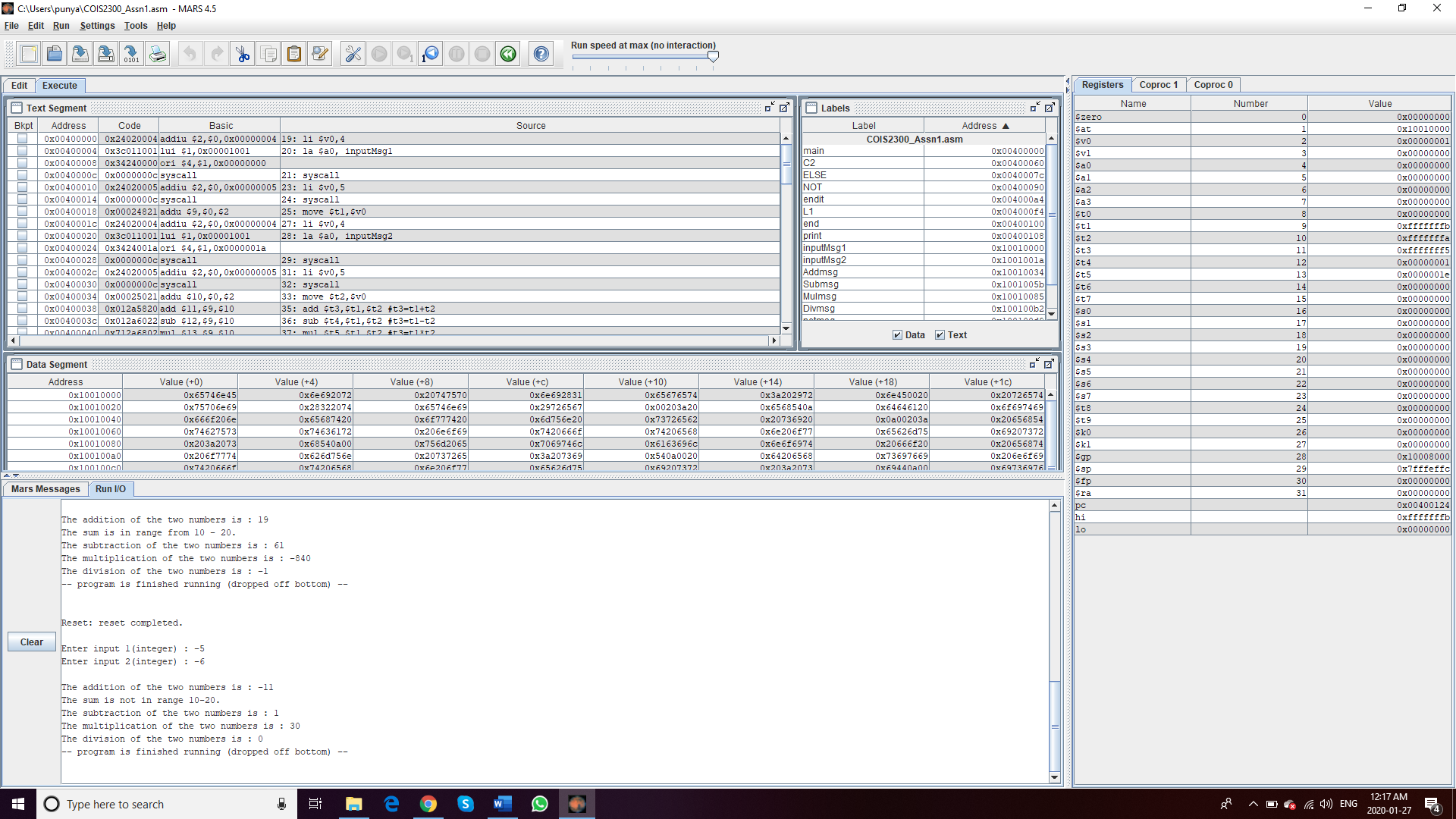
**Test 3:**  Entering the first integer negative **Input : -30, 20  
Output :**

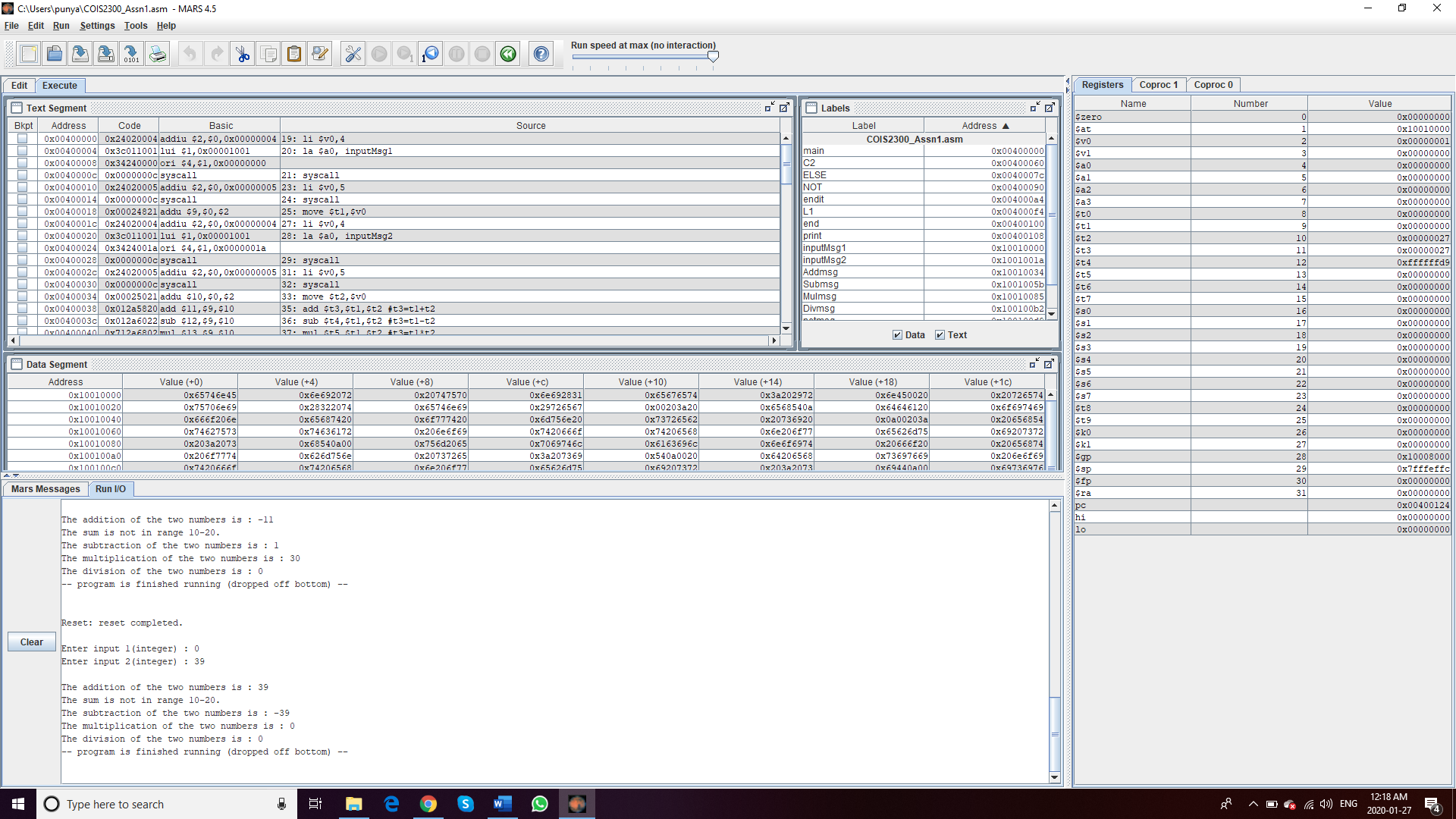


**Test 4:**  Putting second integer negative **Input : 40,-21  
Output :**

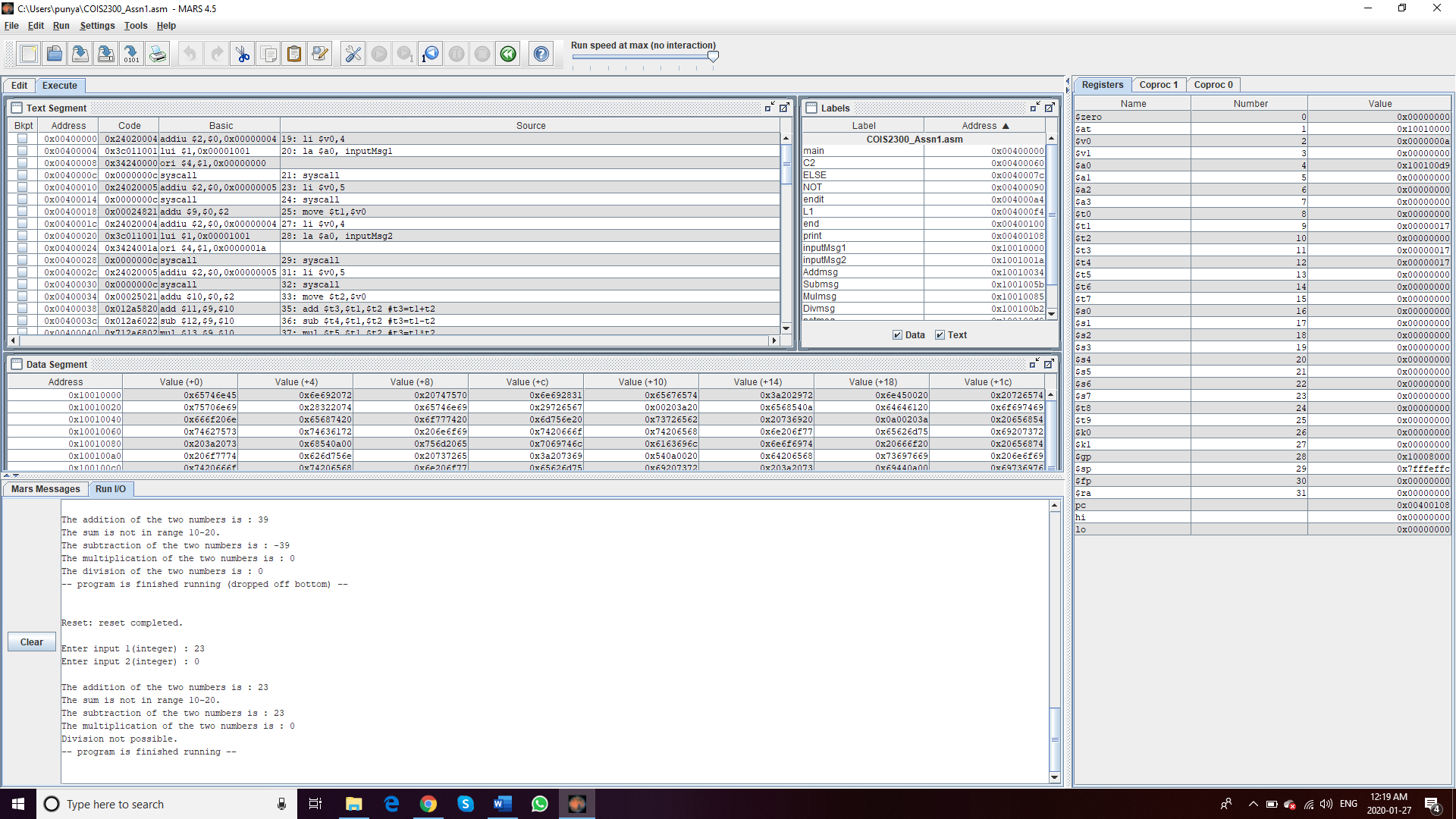


**Test 5:**  Both integers negative  
**Input : -5,-6  
Output :**



**Test 6:** First integer 0 **Input: 0, 39  
Output:**

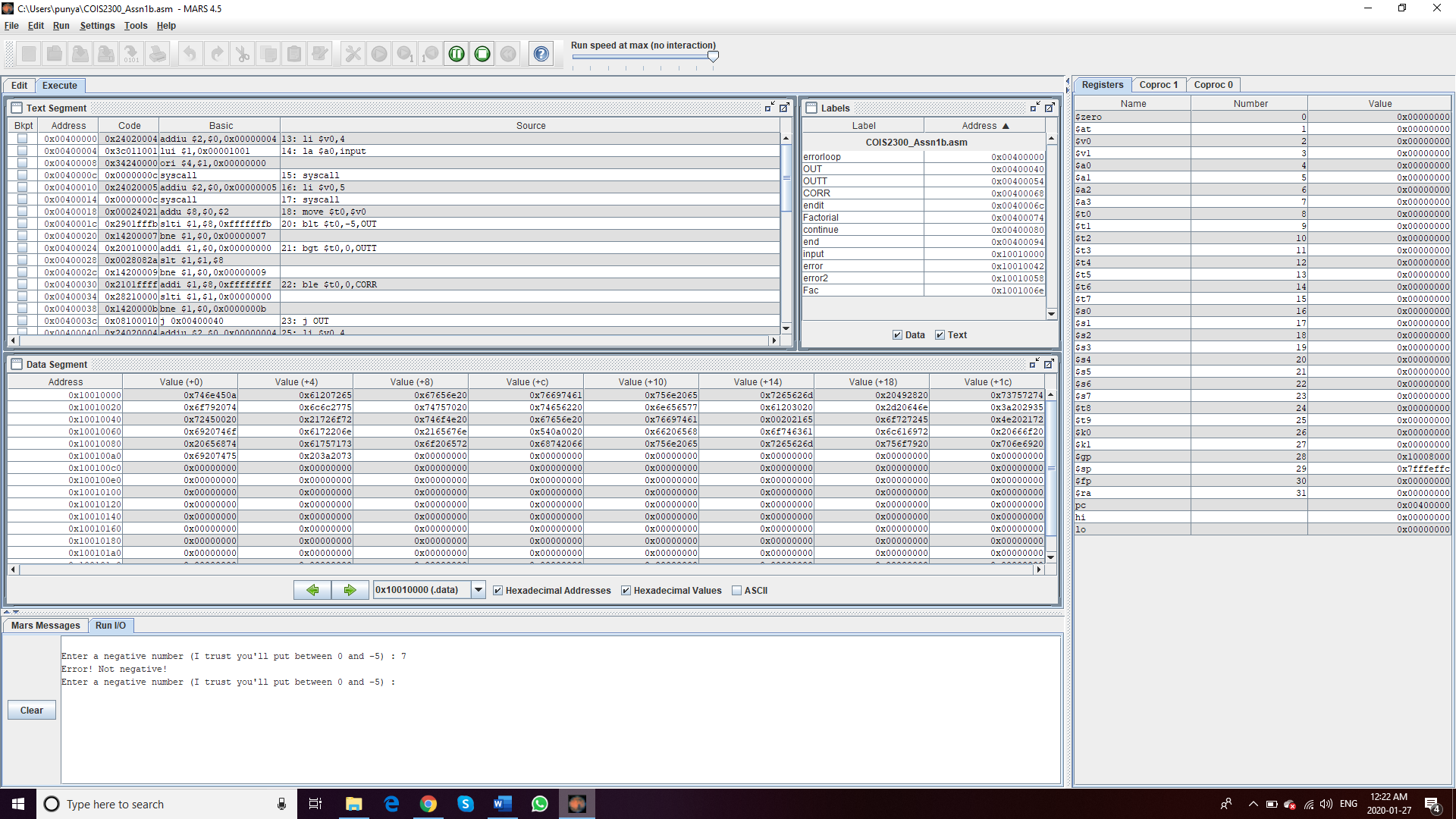
**Test 7:** Second integer 0 **Input : 23,0  
Output:**



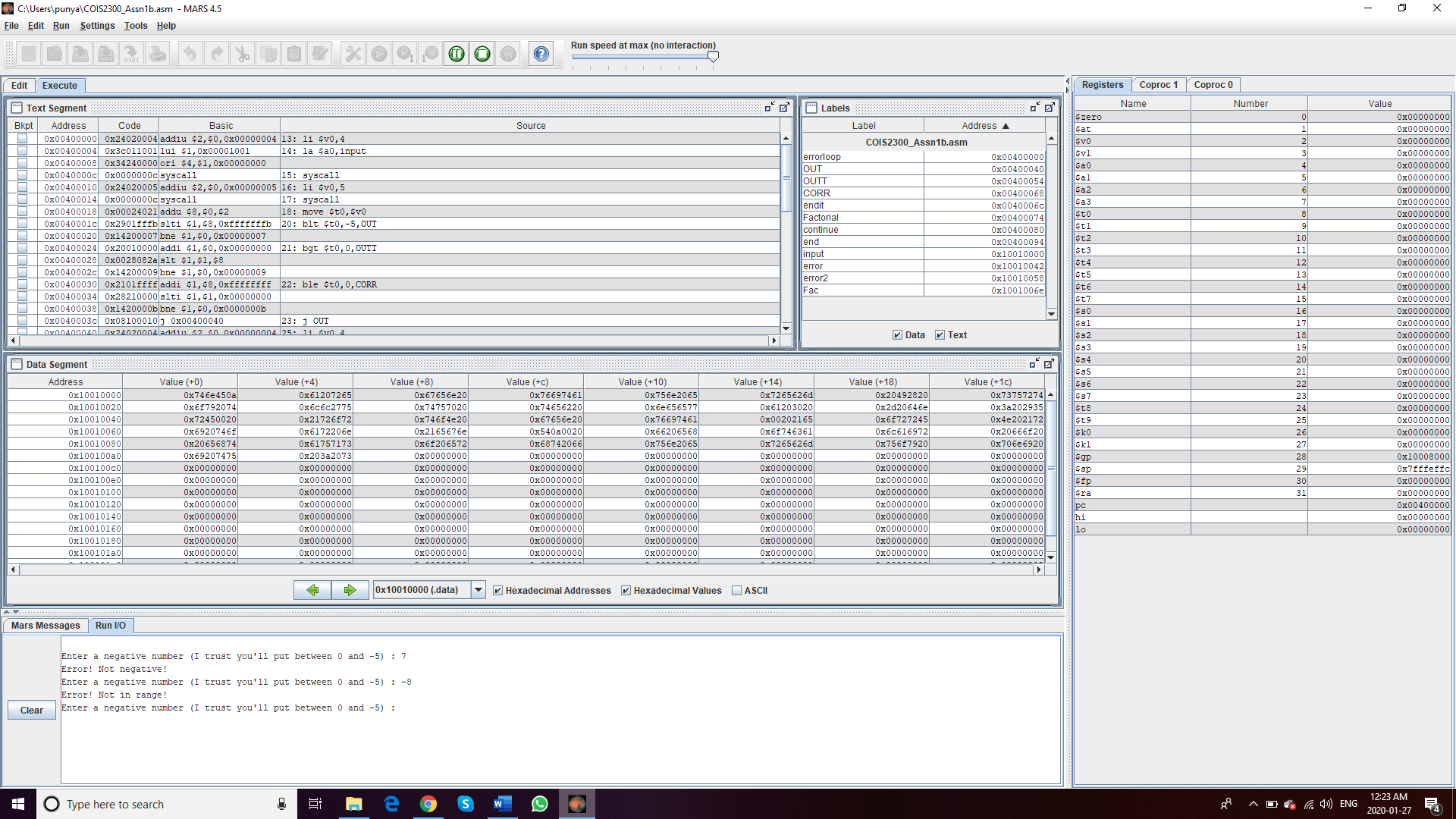
**All the possible tests conducted, and program runs successfully.**

**Program 2:** Write a MIPS program that will ask the user for a negative integer (we will call this *n* ), check if it is negative, if not throw an error. Iteratively calculate the factorial of n squared (). Assume n is between 0 and -5, there are some issues with factorial calculations with big numbers which we will discuss in class… don’t sign yourself up for those). Recursion is a problem for future assignments, so do this iteratively.

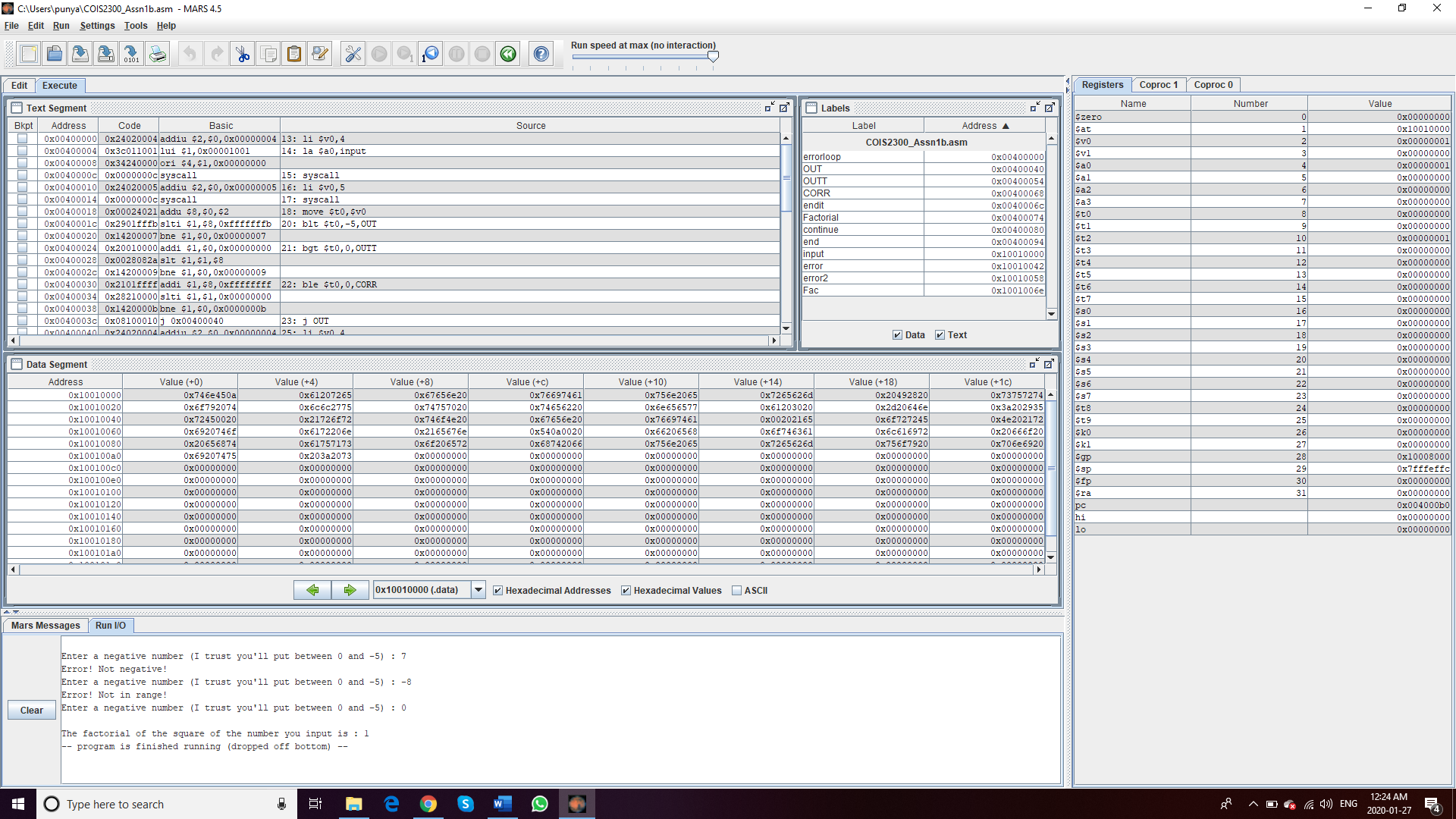
**Test 1:** Since we have to enter a number between 0 and -5, checking the output for input greater than 0 **Input: 7  
Output:**



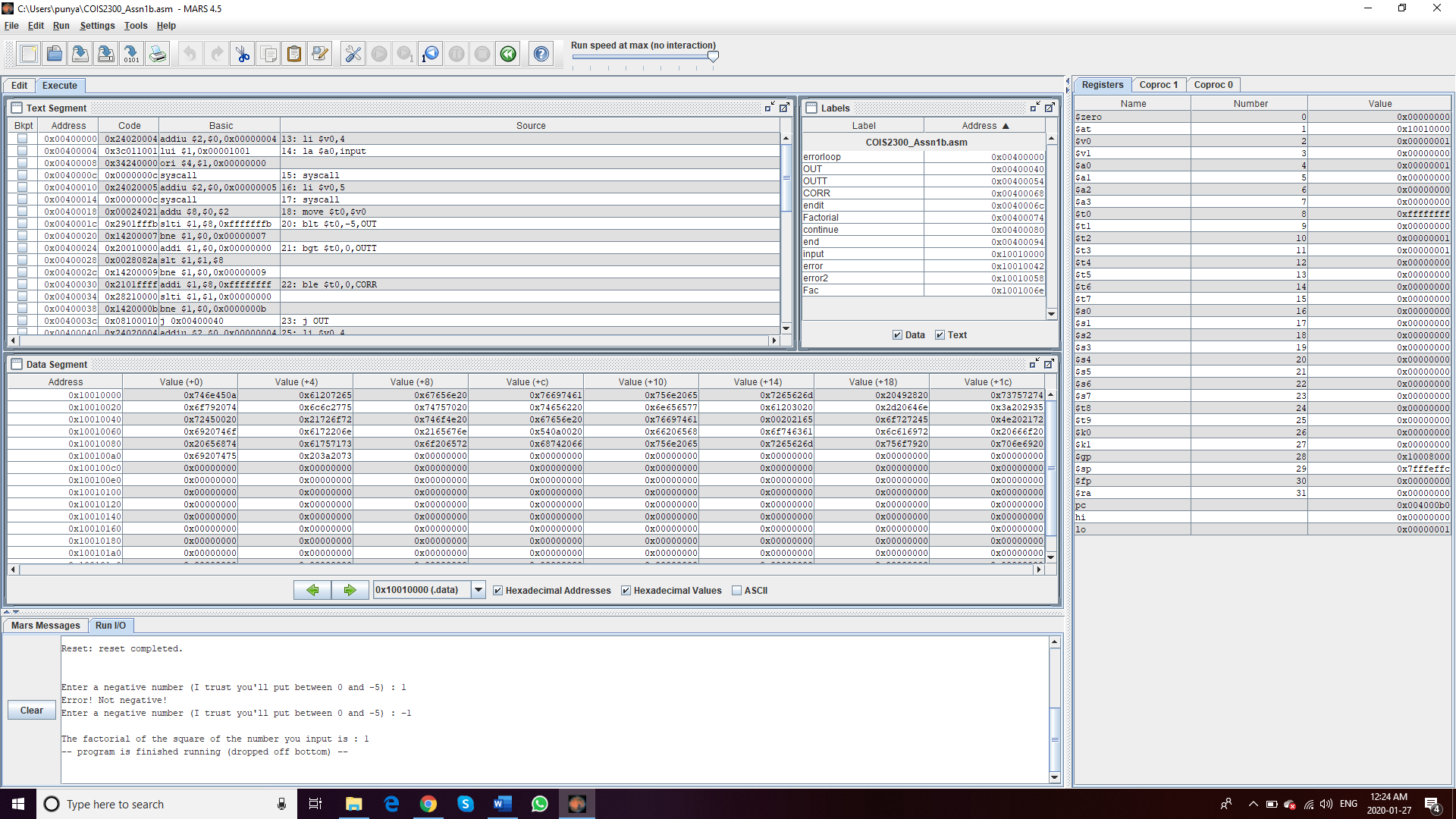
**Test 2:** Input a number lower than -5  
**Input : -8  
Output :**



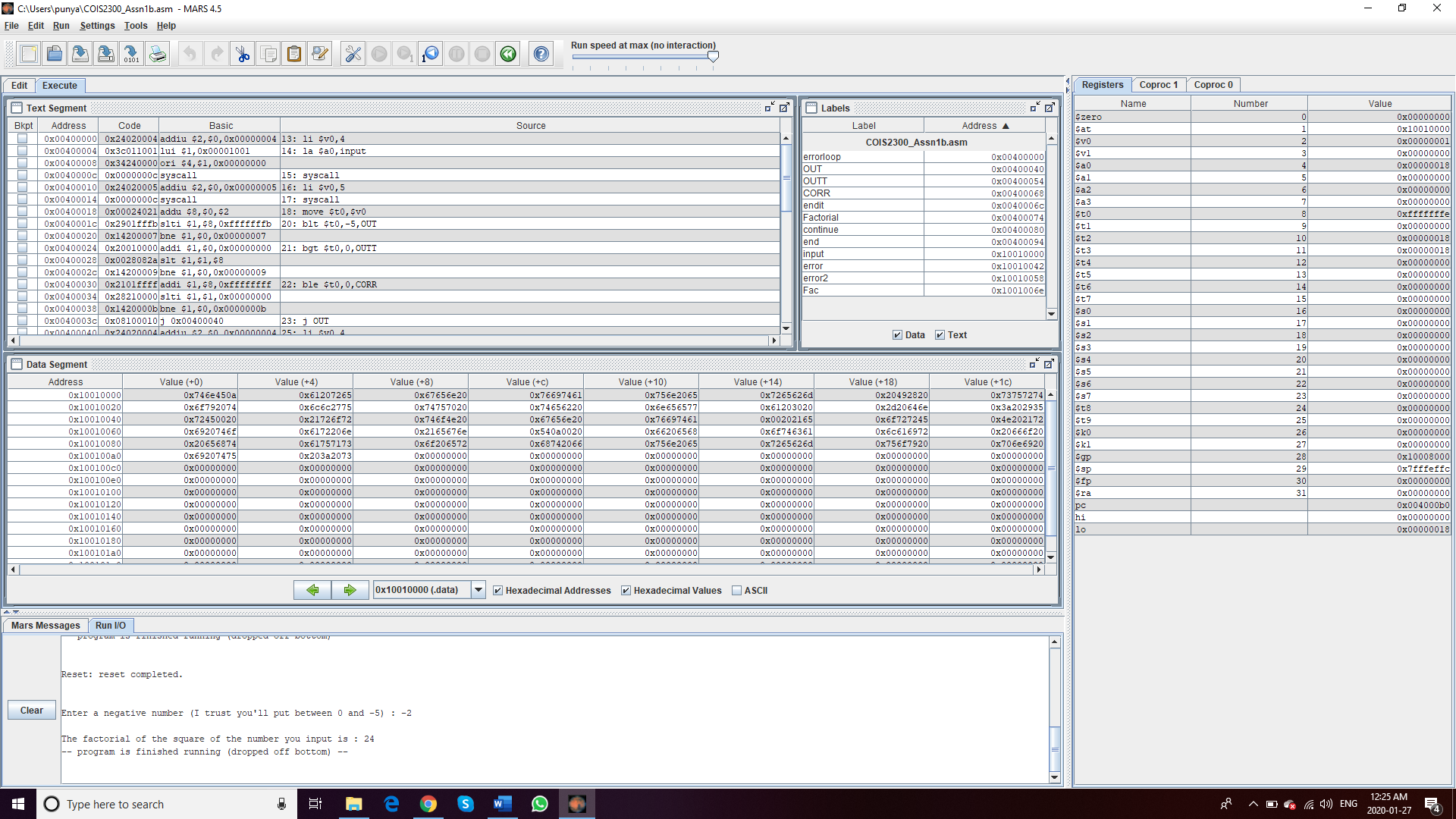
**Test 3 :** Entering 0  
**Input: 0  
Output :**



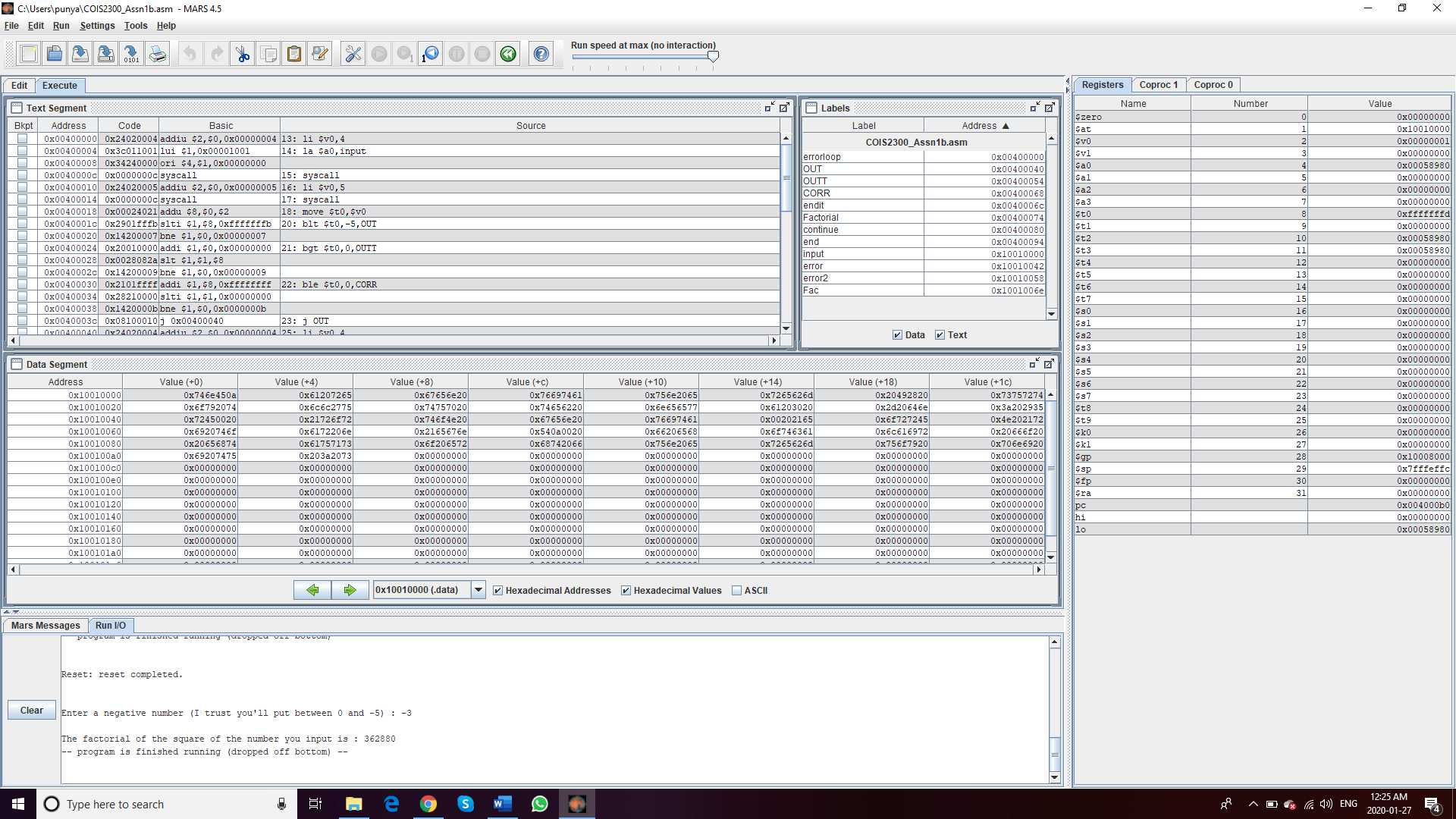
**Test 4 :** Entering -1  
**Input: -1  
Output :**



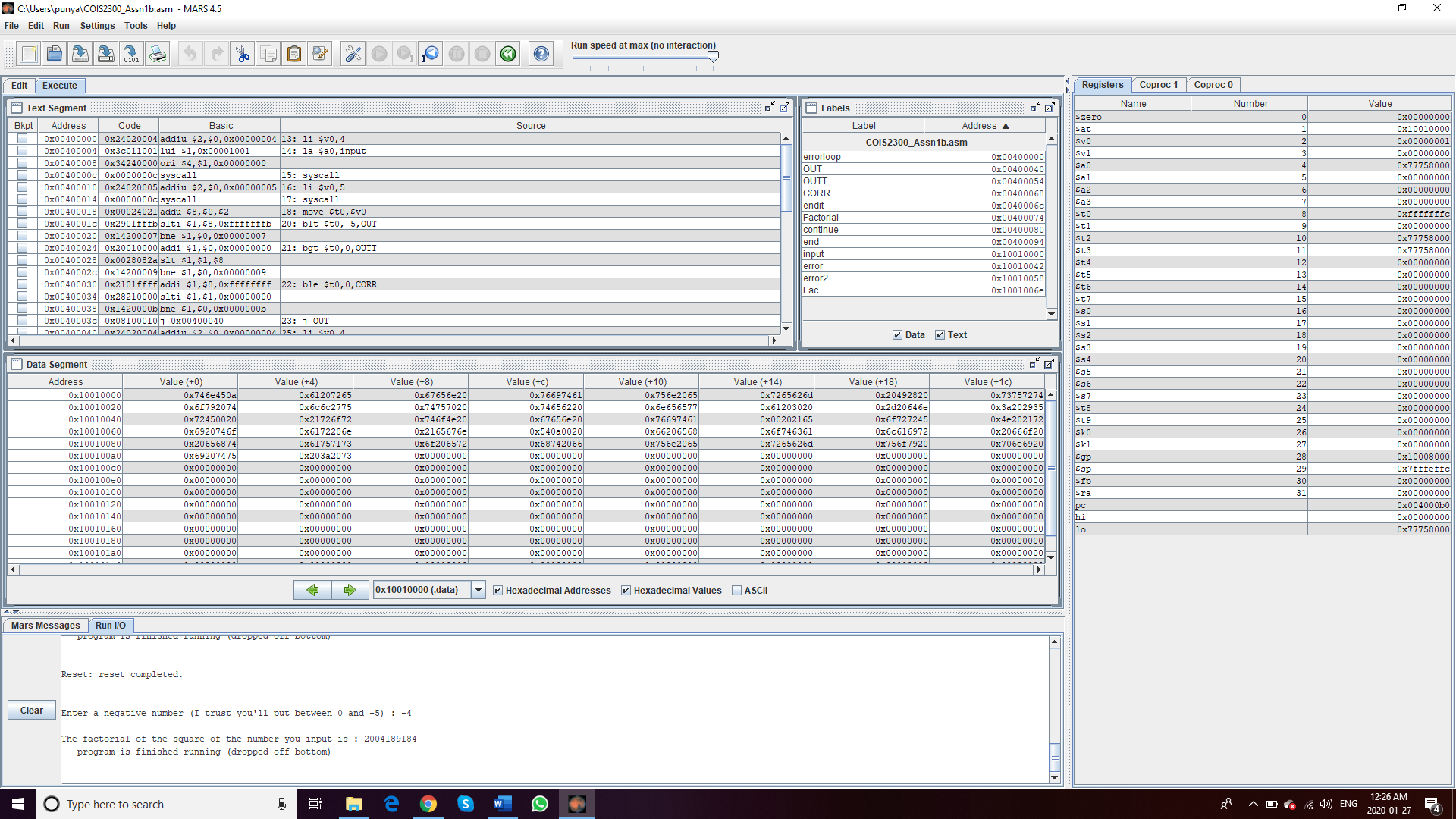
**Test 5 :** Entering -2  
**Input: -2  
Output :**



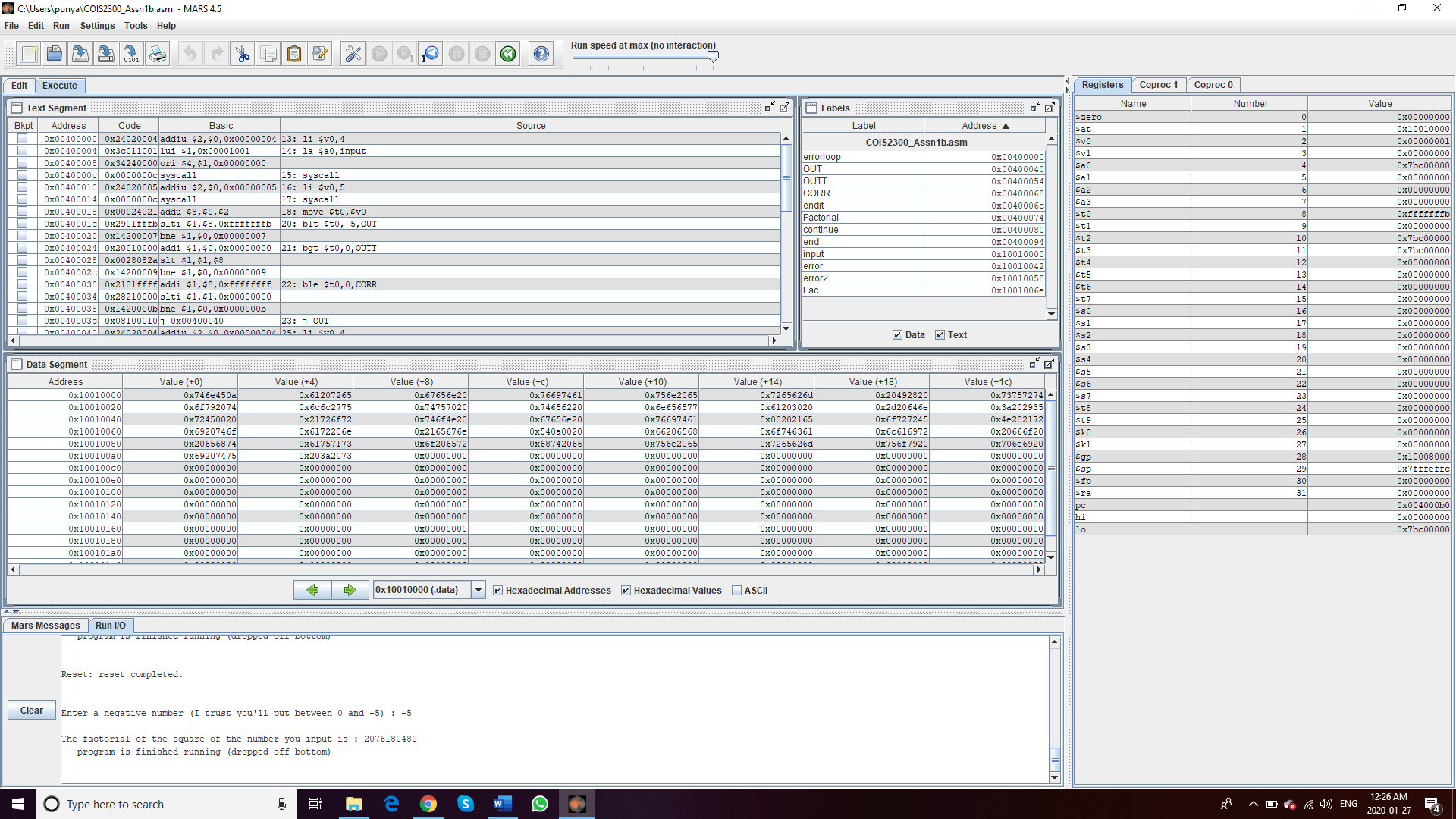
**Test 6 :** Entering -3  
**Input: -3  
Output :**



**Test 7 :** Entering -4  
**Input: -4  
Output :**



**Test 8 :** Entering -5  
**Input: -5  
Output :**



**All the possible tests conducted, and we can see that the program works correctly. However, for input -4 and -5 the results are wrong, that is the program can’t calculate factorial for higher numbers as we were expecting.**