Fall Semester 2021-2022

Microprocessor and Interfacing

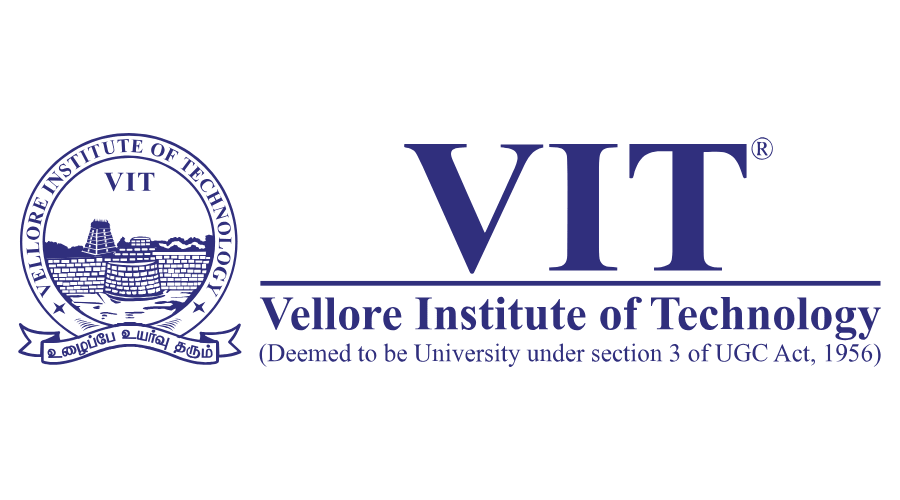
Lab Report

Digital Assignment-6

Experiment No: 7,8,9 Task No: 6

Course Code: CSE2006

Slot: L7+L8



Submitted By: Alokam Nikhitha

Reg. Numb: 19BCE2555

Submitted To: Dr. Abdul Majed KK

**Experiment -7**

**Aim:**

**1)Convert Binary number corresponding to 0109H to BCD number.**

**2) Convert BCD number corresponding to 27H to Binary number.**

**Tools Required:**

**8086 Emulator**

**1) Convert Binary number corresponding to 0109H to BCD number.**

**ALGORITHM**

**Step I              :**Initialize the data segment.

**Step II             :**Initialize BX = 0000 H and DH = 00H.

**Step III           :**    Load the number in AX.

**Step IV           :**Compare number with 10000 decimal. If below goto step VII else goto step V.

**Step V             :**Subtract 10,000 decimal from AX and add 1 decimal to DH

**Step VI           :**Jump to step IV.

**Step VII          :**Compare number in AX with 1000, if below goto step X else goto step VIII.

**Step VIII        :**Subtract 1000 decimal from AX and add 1000 decimal to BX.

**Step IX           :**Jump to step VII.

**Step X             :**Compare the number in AX with 100 decimal if below goto step XIII

**Step XI           :**Subtract 100 decimal from AX and add 100 decimal to BX.

**Step XII         :**Jump to step X

**Step XIII        :**Compare number in AX with 10. If below goto step XVI

**Step XIV        :**Subtract 10 decimal from AX and add 10 decimal to BX..

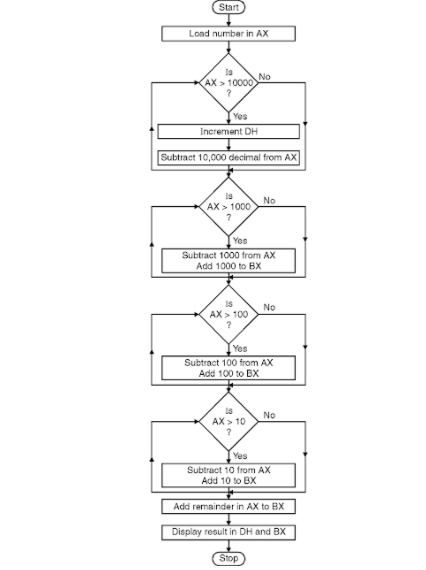
**Step XV          :**Jump to step XIII.

**Step XVI        :**Add remainder in AX with result in BX.

**Step XVII      :**Display the result in DH and BX.

**Step XVIII     :**Stop.

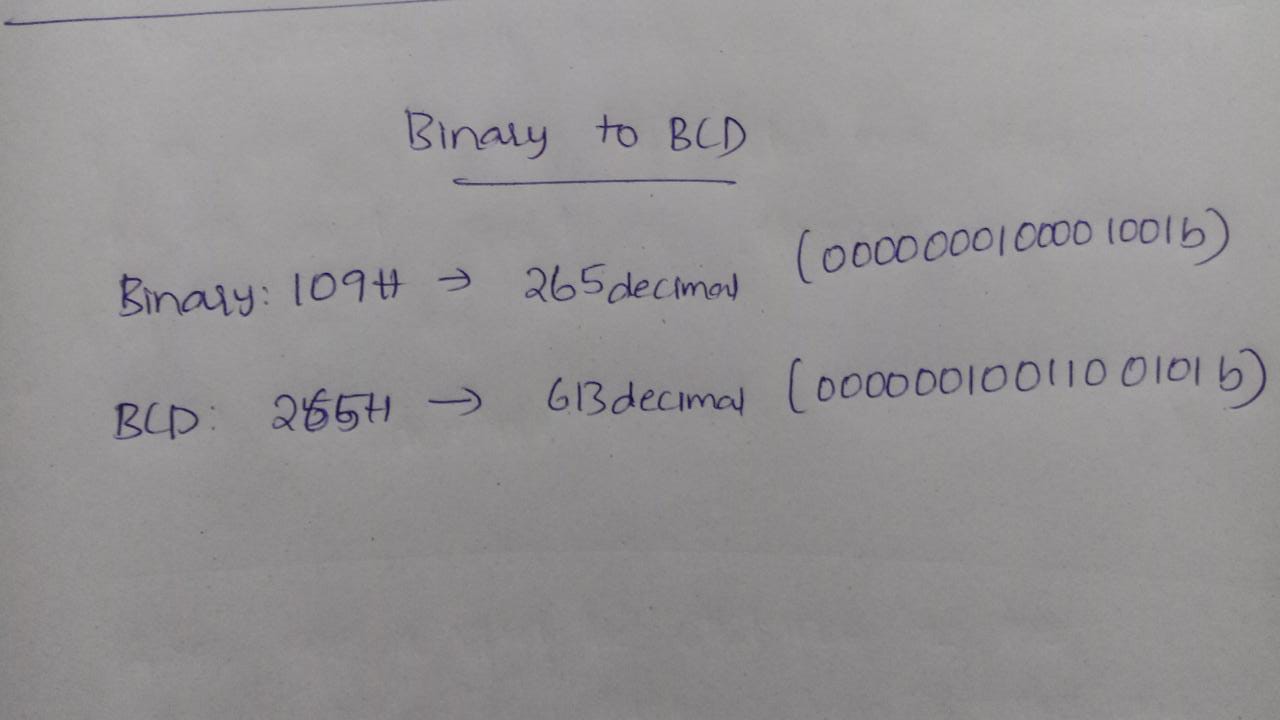
**Flow Chart:**

****

**Design and Calculations:**

Initialize the data segment. Initialize BX = 0000 H and DH = 00H. Load the number in AX. Compare number with 10000 decimal. Subtract 10,000 decimal from AX and add 1 decimal to DH.Jump to step IV. Compare number in AX with 1000, if below goto step Subtract 1000 decimal from AX and add 1000 decimal to BX. Jump to step VII. Compare the number in AX with 100 decimal.Subtract 100 decimal from AX and add 100 decimal to BX. number in AX with 10.Subtract 10 decimal from AX and add 10 decimal to BX. Add remainder in AX with result in BX. Display the result in DH and BX.

Calculations:

****

**Program Code:**

**data segment**

**bin dw 0109h**

**bcd dw ?**

**data ends**

**code segment**

**assume cs :code, ds :data**

**start :**

**mov ax , data**

**mov ds ,ax**

**mov al ,bin**

**mov ah, 00h**

**mov cl, 64h**

**div cl**

**mov bh,al**

**mov al, ah**

**mov ah,00h**

**mov cl,0ah**

**div cl**

**mov cl,04h**

**rol al,cl**

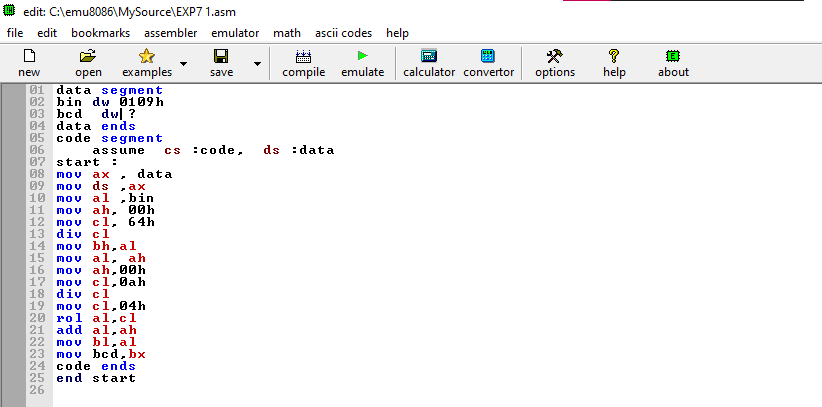
**add al,ah**

**mov bl,al**

**mov bcd,bx**

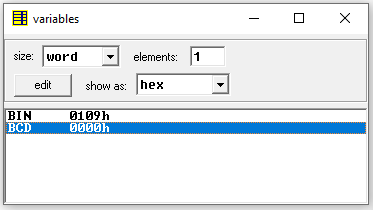
**code ends**

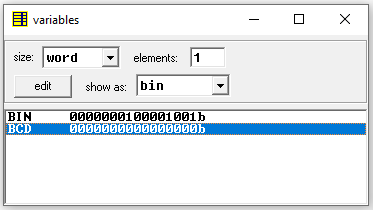
**end start**

****

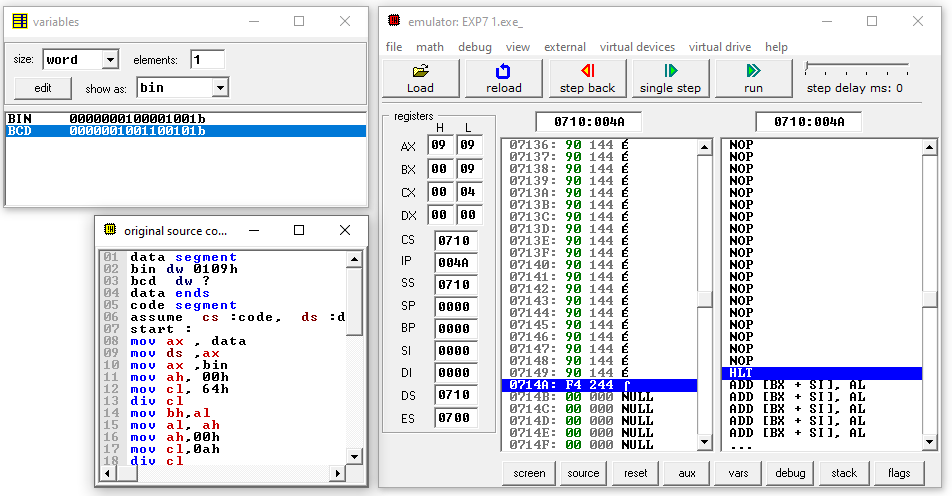
**Output:**

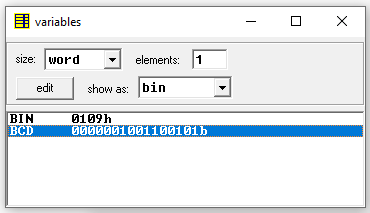
**Before Execution:**

****

****

**After Execution:**

****

****

****

**Result and Inference:**

**The BCD form of Binary number is obtained in Variables.**

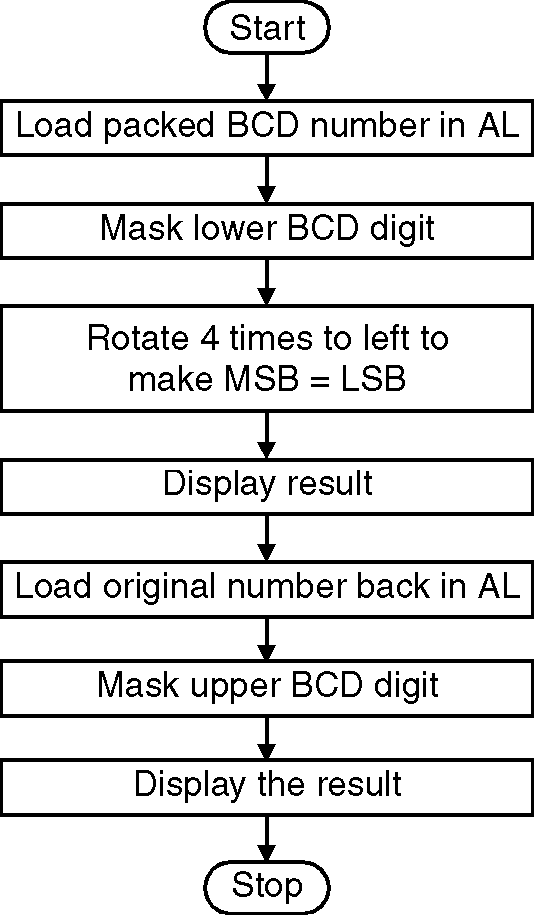
**Bin : 0109h (0000000100001001b) to BCD: 0265h(0000001001100101b)**

**Question 2) Convert BCD number corresponding to 27H to Binary number.**

**ALGORITHM**

* Assign value 27H to BCD in Datset
* Move BCD into AH(Accumulator)
* Move the contents of [SI] in BL.
* Use AND instruction to calculate AND between 0F and contents of BL.
* Move the contents of [SI] in AL.
* Use AND instruction to calculate AND between F0 and contents of AL.
* Move 04 in CL.
* Use ROR instruction on AL.
* Move 0A in DL.
* Use MUL instruction to multiply AL with DL.
* Use ADD instruction to add AL with BL.
* Move the contents of AL in [DI].
* Halt the program.

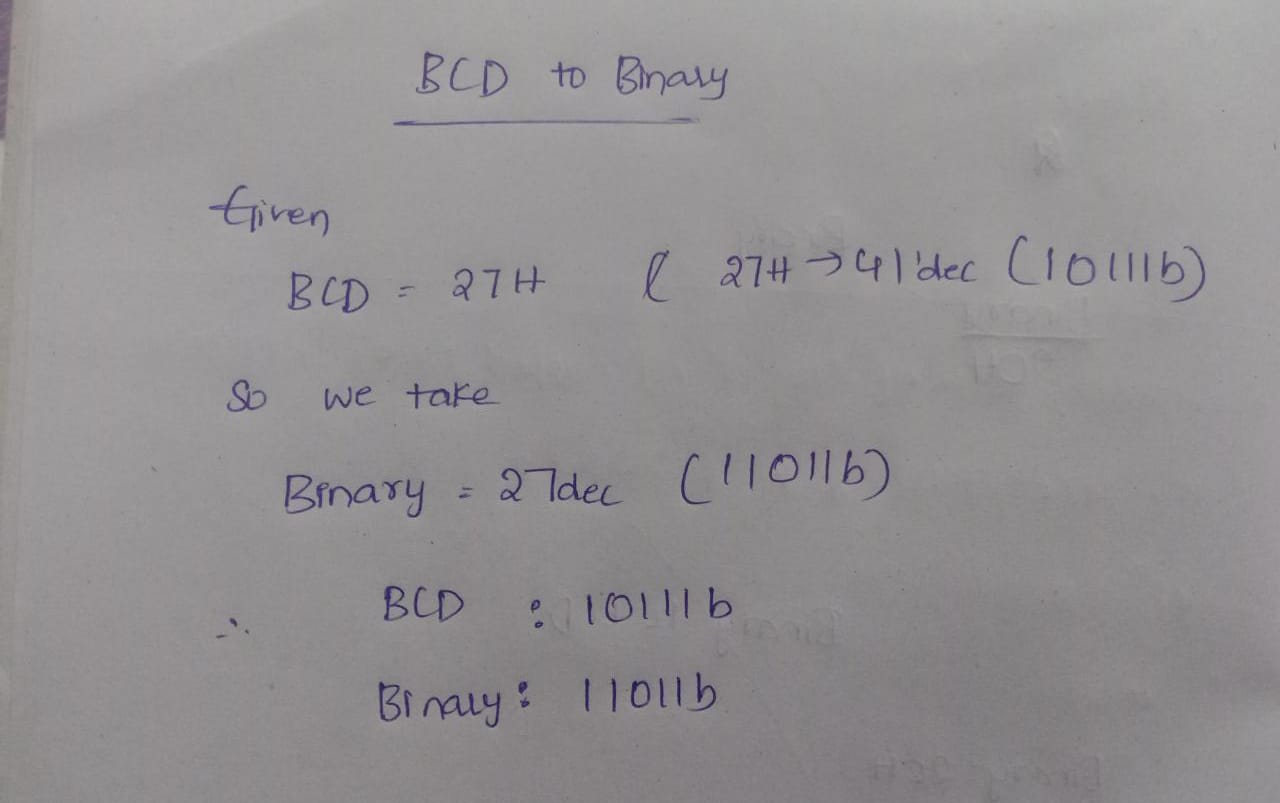
**Flow Chart:**



**Design and Calculations:**

Assign value 27H to BCD in Datset.Move the contents of [SI] in BL.Use AND instruction to calculate AND between 0F and contents of BL.Move the contents of [SI] in AL.Use AND instruction to calculate AND between F0 and contents of AL.Move 04 in CL.Use ROR instruction on AL.Move 0A in DL.Use MUL instruction to multiply AL with DL.Use ADD instruction to add AL with BL.Move the contents of AL in [DI].Halt the program.\

Calculation:



**Program Code:**

**DATA\_SEG SEGMENT**

**BCD DB 27H ; STORAGE FOR A BCD VALUE**

**BIN DB ? ; STORAGE FOR BINARY VALUE**

**DATA\_SEG ENDS**

**CODE\_SEG SEGMENT**

**ASSUME CS:CODE\_SEG,DS:DATA\_SEG**

**START:**

**MOV AX,DATA\_SEG**

**MOV DS,AX**

**MOV AH,BCD**

**MOV BH,AH**

**AND BH,0FH**

**AND AH,0F0H**

**ROR AH,04**

**MOV CL,10**

**MOV AL,AH**

**AND AX,00FFH**

**MUL CL**

**ADD AL,BH**

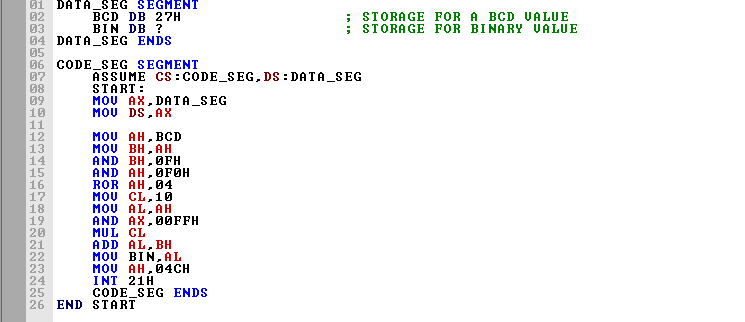
**MOV BIN,AL**

**MOV AH,04CH**

**INT 21H**

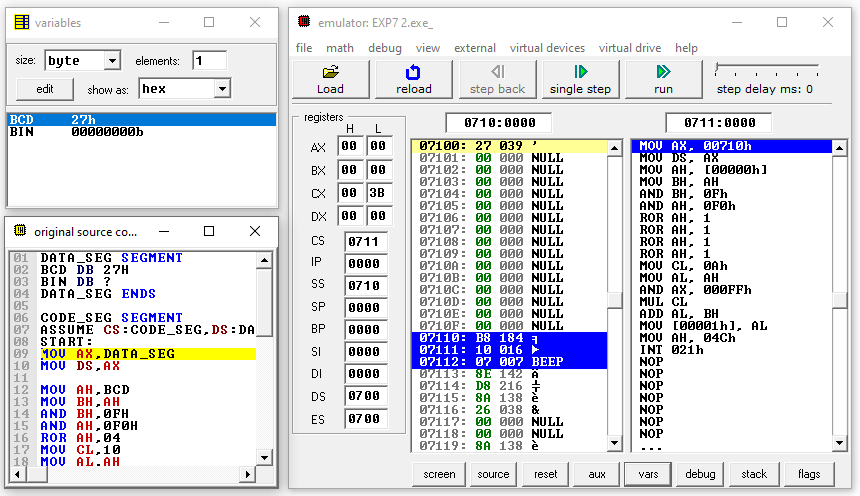
**CODE\_SEG ENDS**

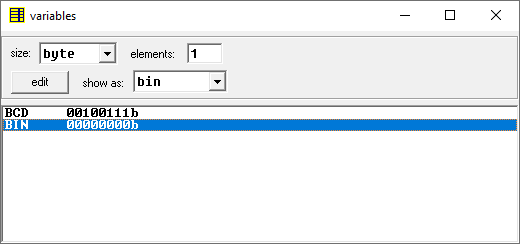
**END START**

****

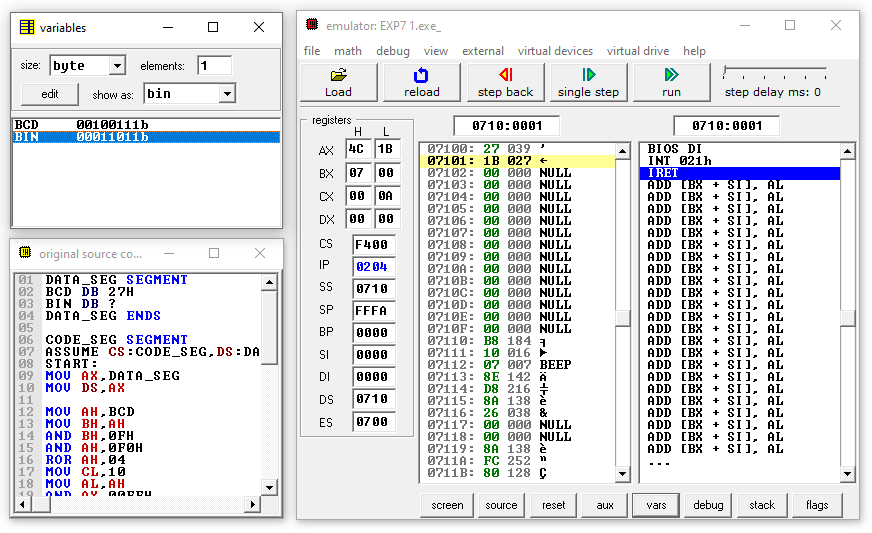
**Output:**

**Before Execution:**

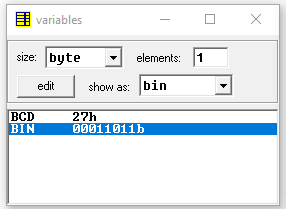
****

****

**After Execution:**

****

****

****

**Result and Inference:**

**On execution we got the Binary of the given BCD(00100111b) 27H as 00011011b which is 1BH and the answer is stored in BIN variable.**

**Experiment -8.1**

**Aim:**

1. **Write an ALP program to sort the numbers in ascending / descending order**
2. **Write an ALP to find square and cube of an 8 bit number**
3. **Write an ALP to check if the given number is even or odd.**

**Tools Required:**

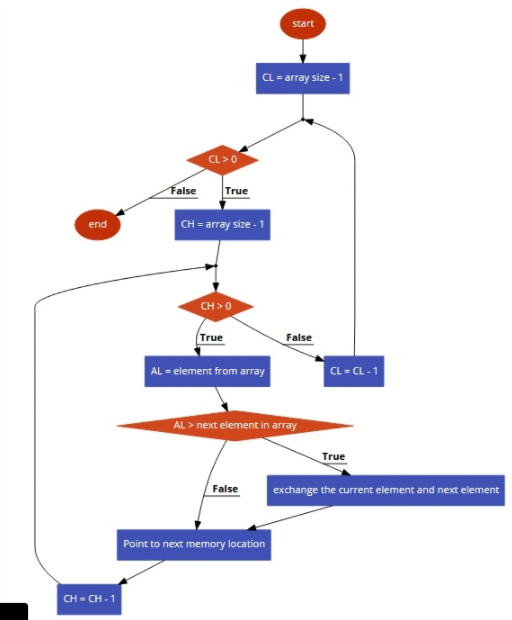
**8086 Emulator**

1. **Write an ALP program to sort the numbers in ascending / descending order**

**ALGORITHM:**

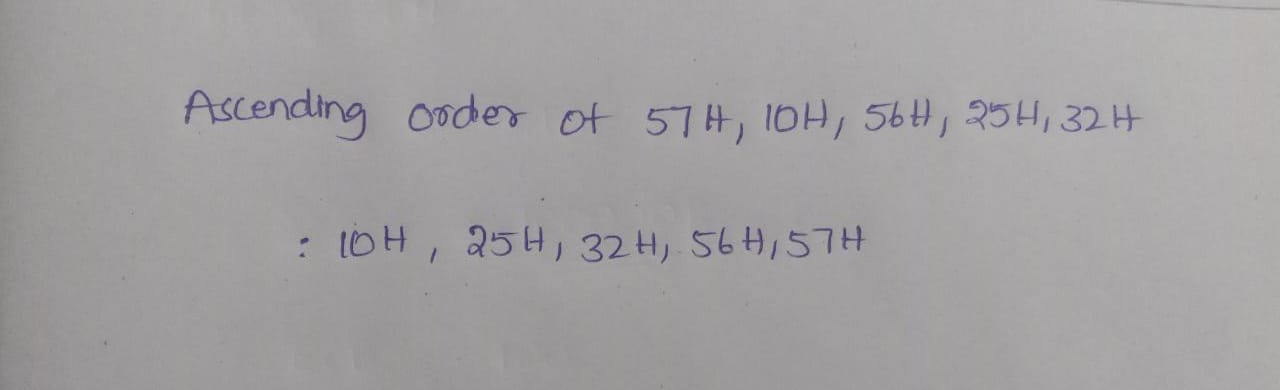
1. Load data from offset 0 to register CL (for count).
2. Travel from starting memory location to last and compare two numbers if first number is greater than second number then swap them.
3. First pass fix the position for last number.
4. Decrease the count by 1.
5. Again travel from starting memory location to (last-1, by help of count) and compare two numbers if first number is greater than second number then swap them.
6. Second pass fix the position for last two numbers.
7. Repeated.

**FLOWCHART:**

****

**DESIGN AND CALCULATIONS:**

Load data from offset to register CL (for count).Travel from starting memory location to last and compare two numbers if first number is greater than second number then swap them.First pass fix the position for last number.Decrease the count by 1.Again travel from starting memory location to (last-1, by help of count) and compare two numbers if first number is greater than second number then swap them. Second pass fix the position for last two numbers Repeated.

****

**PROGRAM CODE:**

**DATA SEGMENT**

**STRING1 DB 57H,10H,56H,25H,32H**

**DATA ENDS**

**CODE SEGMENT**

**ASSUME CS:CODE,DS:DATA**

**START: MOV AX,DATA**

**MOV DS,AX**

**MOV CH,04H**

**UP2: MOV CL,04H**

**LEA SI,STRING1**

**UP1: MOV AL,[SI]**

**MOV BL,[SI+1]**

**CMP AL,BL**

**JC DOWN**

**MOV DL,[SI+1]**

**XCHG [SI],DL**

**MOV [SI+1],DL**

**DOWN: INC SI**

**DEC CL**

**JNZ UP1**

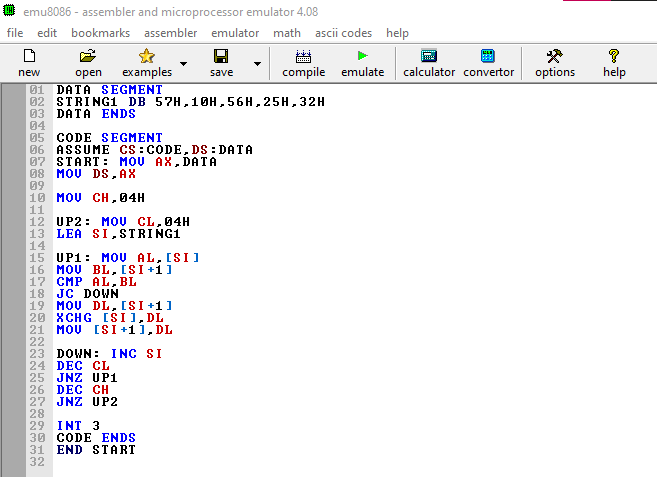
**DEC CH**

**JNZ UP2**

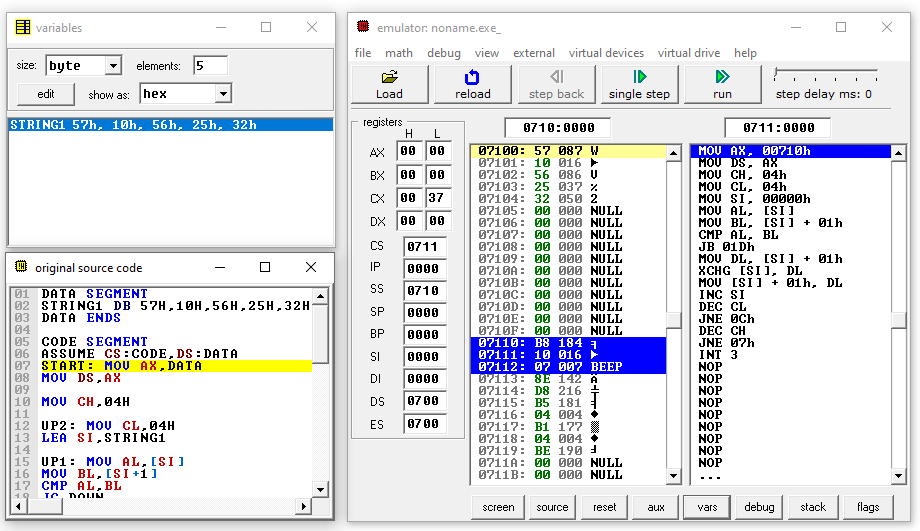
**INT 3**

**CODE ENDS**

**END START**

****

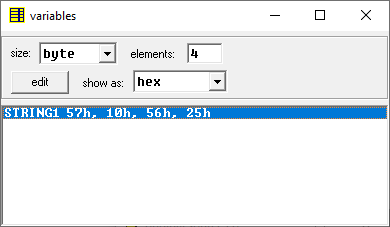
**Before Execution:**

****

**Memory Location**

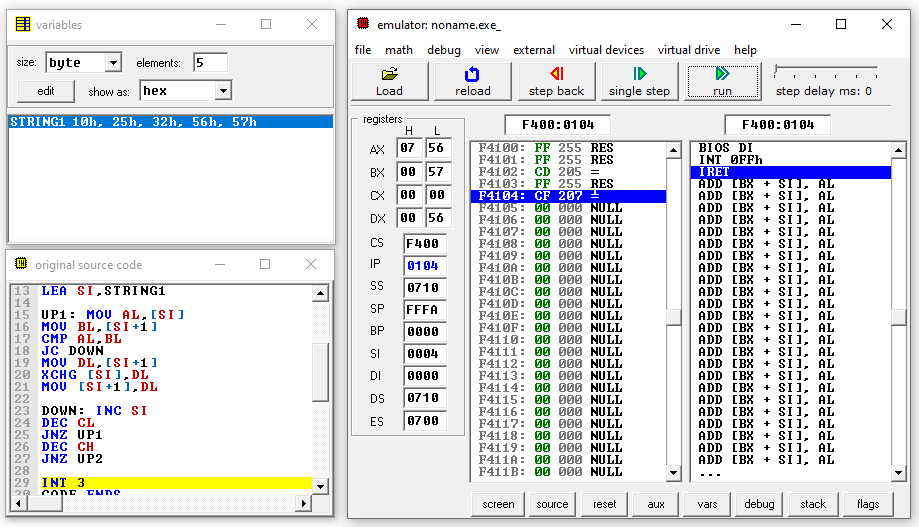
****

**Variables**

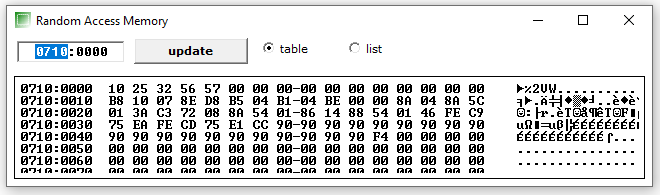
****

**OUTPUT:**

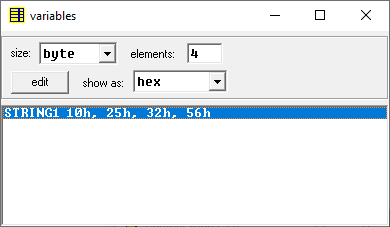
**After Execution:**

****

**Memory Location**

****

**Variables**

****

**RESULTS & INFERENCE**

**The sorted Array is stored in String1 which can be found in variables and the sorted values are present in location 0710:0000**

**2. Write an ALP to find square and cube of an 8 bit number**

**ALGORITHM**

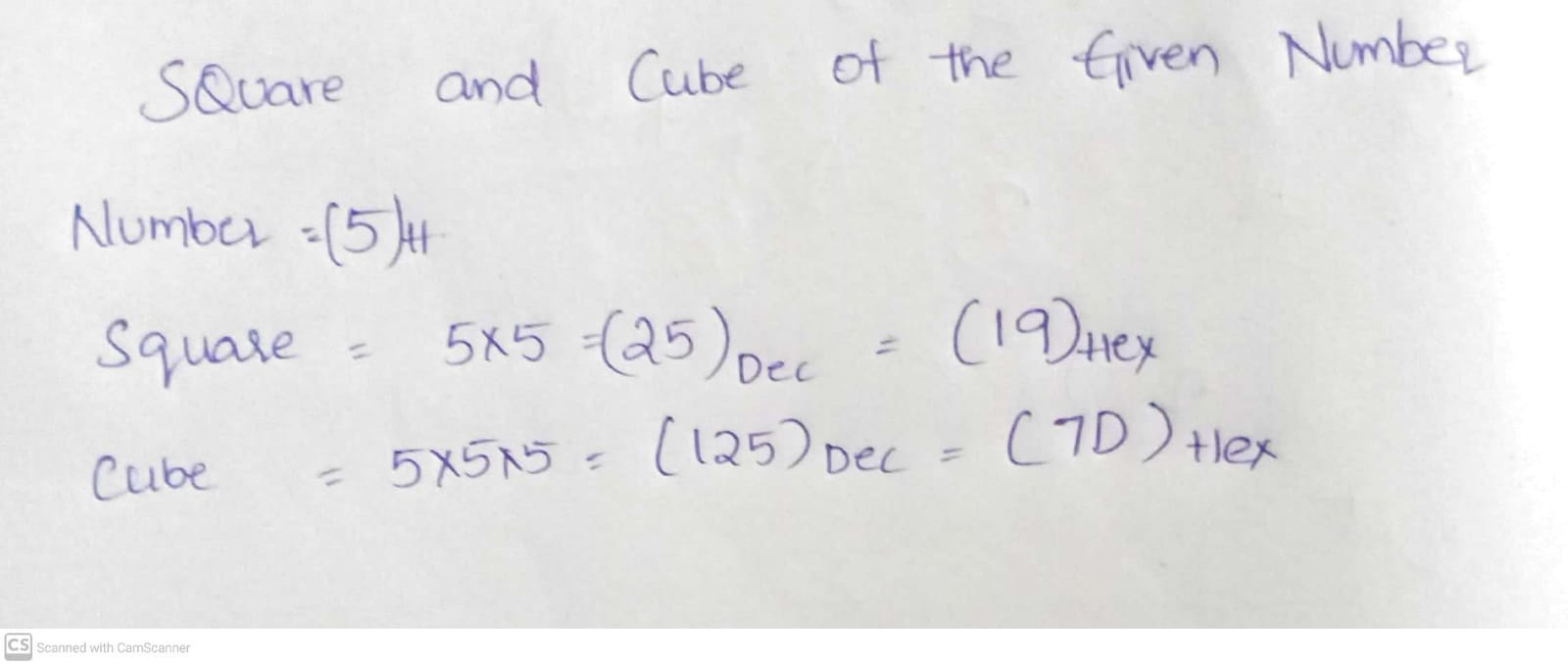
* **Input the Number N whose Square and Cube is to be found in the data Segment and assign the value to N**
* **Input the SQUARE and CUBE variables with no value**
* **Move the Value of N into the Accumulator**
* **Multiply N to the value in Accumulator . We get the square of the Number**
* **Move the value in Accumulator to Base Register and SQUARE variable**
* **Multiply the Number again with the Accumulator. We get the Cube of the given Number.**
* **Move the value obtained in Accumulator into CUBE variable**
* **The Square and Cube of the Number given is stored in SQUARE and CUBE variables respectively.**

**FLOWCHART**

****

**Design and Calculations:**

**Input the Number N whose Square and Cube is to be found in the data Segment and assign the value to N.Input the SQUARE and CUBE variables with no value.Move the Value of N into the Accumulator.Multiply N to the value in Accumulator . We get the square of the Number.Move the value in Accumulator to Base Register and SQUARE variable.Multiply the Number again with the Accumulator. We get the Cube of the given Number.Move the value obtained in Accumulator into CUBE variable.The Square and Cube of the Number given is stored in SQUARE and CUBE variables respectively.**

****

**Program Code:**

**DATA SEGMENT**

**A DW 5H**

**SQUARE DW ?**

**CUBE DW ?**

**DATA ENDS**

**CODE SEGMENT**

**ASSUME DS:DATA,CS:CODE**

**START:**

**MOV AX,DATA**

**MOV DS,AX**

**MOV AX,A**

**MUL A**

**MOV SQUARE, AX**

**MOV BX, AX**

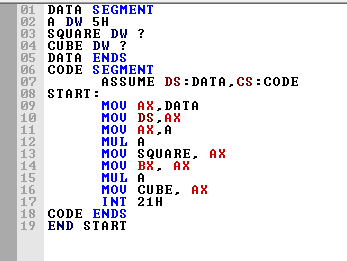
**MUL A**

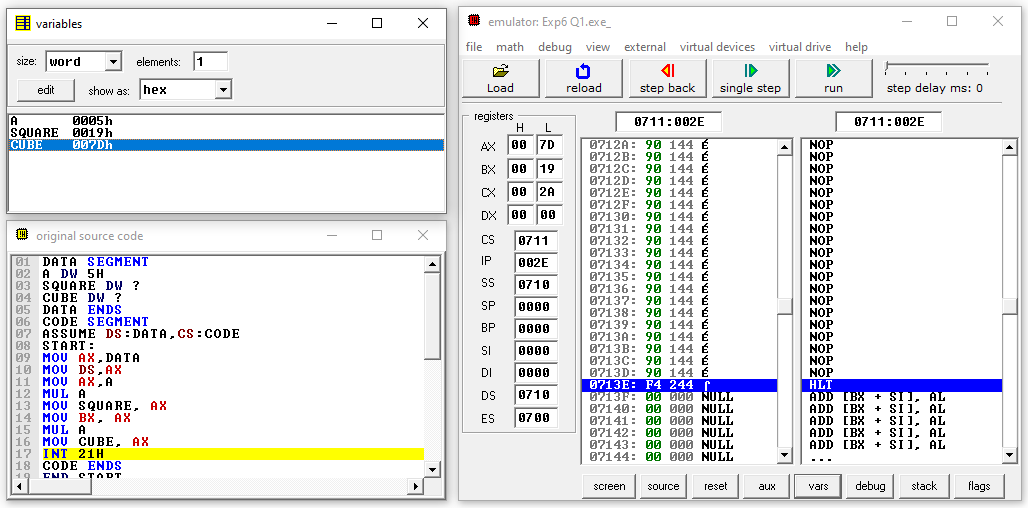
**MOV CUBE, AX**

**INT 21H**

**CODE ENDS**

**END START**

****

**Output: **

**Result and Inference:**

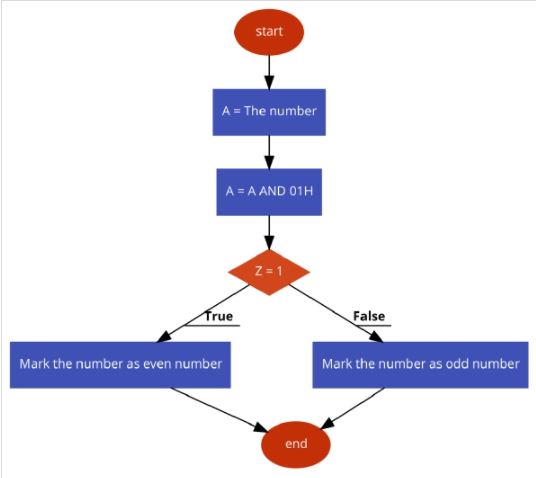
**The value in the Accumulator is the Cube of the number given and Square of the number is stored in Counter Register. We can see that SQUARE(0019H) and CUBE (007DH) variables are filled with values of square andd cube of the number(0005H) given**

**3. Write an ALP to check if the given number is even or odd.**

**ALGORITHM:**

* Load the content of the input from screen in accumulator A.
* Perform AND operation with 01 in value of accumulator A by the help of ANI instruction.
* Check if zero flag is set, i.e if ZF = 1 then store 22 in accumulator A otherwise store 11 in A.
* Store the value of A in memory location
* If Number is Even print msg1 or else print message 2.

**FLOWCHART:**

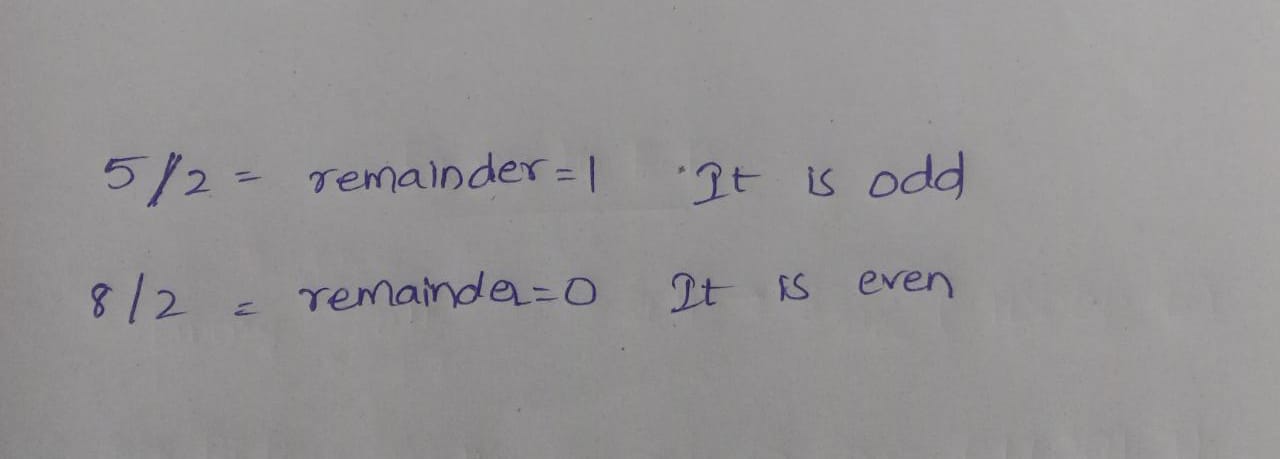
****

**DESIGN AND CALCULATIONS:**

Load the content of the input from screen in accumulator A.

Perform AND operation with 01 in value of accumulator A by the help of ANI instruction.Check if zero flag is set, i.e if ZF = 1 then store 22 in accumulator A otherwise store 11 in A.

Store the value of A in memory location If Number is Even print msg1 or else print message 2.

****

**PROGRAM CODE:**

**DATA SEGMENT**

**MSG1 DB 10,13,’ENTER NUMBER HERE :- $’**

**MSG2 DB 10,13,’ENTERED VALUE IS EVEN$’**

**MSG3 DB 10,13,’ENTERED VALUE IS ODD$’**

**DATA ENDS**

**DISPLAY MACRO MSG**

**MOV AH,9**

**LEA DX,MSG**

**INT 21H**

**ENDM**

**CODE SEGMENT**

**ASSUME CS:CODE,DS:DATA**

**START:**

**MOV AX,DATA**

**MOV DS,AX**

**DISPLAY MSG1**

**MOV AH,1**

**INT 21H**

**MOV AH,0**

**CHECK: MOV DL,2**

**DIV DL**

**CMP AH,0**

**JNE ODD**

**EVEN:**

**DISPLAY MSG2**

**JMP DONE**

**ODD:**

**DISPLAY MSG3**

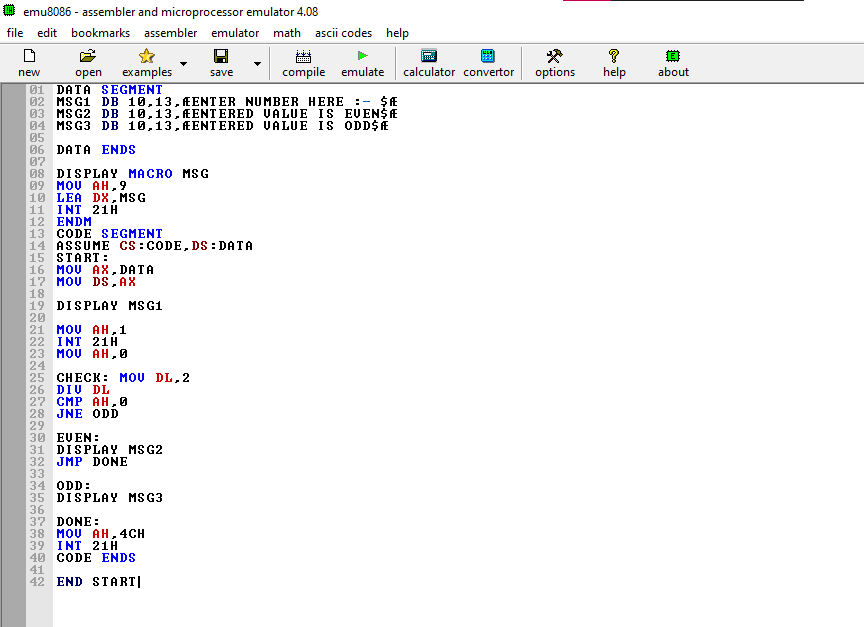
**DONE:**

**MOV AH,4CH**

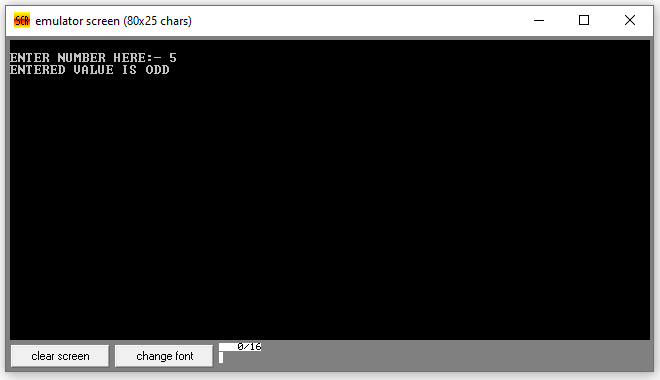
**INT 21H**

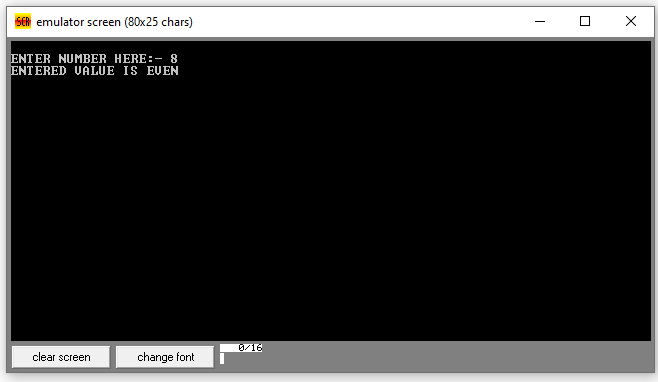
**CODE ENDS**

**END START**

****

**OUTPUT:**

****

****

**RESULTS & INFERENCE**

**When input is given as 5, since it is odd the second message is displayed in the screen. And when input is given as 8 since it is even number message 2 is displayed.**

**Experiment -9**

**Stepper motor**

**Types & Its Working**

A stepper motor is an electromechanical device it converts electrical power into mechanical power. Also, it is a brushless, synchronous electric motor that can divide a full rotation into an expansive number of steps. The motor’s position can be controlled accurately without any feedback mechanism, as long as the motor is carefully sized to the application. Stepper motors are similar to switched [reluctance motors.](https://www.elprocus.com/what-is-a-reluctance-motor-working-its-applications/) The stepper motor uses the theory of operation for magnets to make the motor shaft turn a precise distance when a pulse of electricity is provided. The stator has eight poles, and the rotor has six poles. The rotor will require 24 pulses of electricity to move the 24 steps to make one complete revolution. Another way to say this is that the rotor will move precisely 15° for each pulse of electricity that the motor receives.

## **Construction & Working Principle**

The **construction of a stepper motor** is fairly related to a[DC motor](https://www.elprocus.com/dc-motor-basics-types-application/). It includes a permanent magnet like Rotor which is in the middle & it will turn once force acts on it. This rotor is enclosed through a no. of the stator which is wound through a magnetic coil all over it. The stator is arranged near to rotor so that magnetic fields within the stators can control the movement of the rotor.

The stepper motor can be controlled by energizing every stator one by one. So the stator will magnetize & works like an electromagnetic pole which uses repulsive energy on the rotor to move forward. The stator’s alternative magnetizing as well as demagnetizing will shift the rotor gradually &allows it to turn through great control.

The**stepper motor working principle** is Electro-Magnetism. It includes a rotor which is made with a permanent magnet whereas a stator is with electromagnets. Once the supply is provided to the winding of the stator then the magnetic field will be developed within the stator. Now rotor in the motor will start to move with the rotating magnetic field of the stator. So this is the fundamental working principle of this motor.

In this motor, there is a soft iron that is enclosed through the electromagnetic stators. The poles of the stator as well as the rotor don’t depend on the kind of stepper. Once the stators of this motor are energized then the rotor will rotate to line up itself with the stator otherwise turns to have the least gap through the stator. In this way, the stators are activated in a series to revolve the stepper motor.

#### **Driving Techniques**

**Stepper motor driving technique**s can be possible with some special circuits due to their complex design. There are several methods to drive this motor, some of them are discussed below by taking an example of a four-phase stepper motor.

#### Single Excitation Mode

The basic method of driving a stepper motor is a single excitation mode. It is an old method and not used much at present but one has to know about this technique. In this technique every phase otherwise stator next to each other will be triggered one by one alternatively with a special circuit. This will magnetize & demagnetize the stator to move the rotor forward.

### **Stepper Motor Circuit & Its Operation**

Stepper motors operate differently from [DC brush motors](https://www.elprocus.com/dc-motor-basics-types-application/), which rotate when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple toothed electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, for example, a microcontroller.

To make the motor shaft turn, first one electromagnet is given power, which makes the gear’s teeth magnetically attracted to the electromagnet’s teeth. At the point when the gear’s teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned ON and the first is turned OFF, the gear rotates slightly to align with the next one and from there the process is repeated. Each of those slight rotations is called a step, with an integer number of steps making a full rotation.

In that way, the motor can be turned by a precise. Stepper motor doesn’t rotate continuously, they rotate in steps. There are 4 coils with a 90o angle between each other fixed on the stator. The stepper motor connections are determined by the way the coils are interconnected. In a stepper motor, the coils are not connected. The motor has a 90o rotation step with the coils being energized in a cyclic order, determining the shaft rotation direction.

The working of this motor is shown by operating the switch. The coils are activated in series in 1-sec intervals. The shaft rotates 90o each time the next coil is activated. Its low-speed torque will vary directly with current.

### Types of Stepper Motor

There are three main types of stepper motors, they are:

* Permanent magnet stepper
* Hybrid synchronous stepper
* Variable reluctance stepper

#### **Permanent Magnet Stepper Motor**

Permanent magnet motors use a permanent magnet (PM) in the rotor and operate on the attraction or repulsion between the rotor PM and the stator electromagnets.

This is the most common type of stepper motor as compared with different types of stepper motors available in the market. This motor includes permanent magnets in the construction of the motor. This kind of motor is also known as tin-can/can-stack motor. The main benefit of this stepper motor is less manufacturing cost. For every revolution, it has 48-24 steps.

#### **Variable Reluctance Stepper Motor**

Variable reluctance (VR) motors have a plain iron rotor and operate based on the principle that minimum reluctance occurs with minimum gap, hence the rotor points are attracted toward the stator magnet poles.

The stepper motor like variable reluctance is the basic type of motor and it is used for the past many years. As the name suggests, the rotor’s angular position mainly depends on the magnetic circuit’s reluctance that can be formed among the teeth of the stator as well as a rotor.

#### **Hybrid Synchronous Stepper Motor**

Hybrid stepper motors are named because they use a combination of permanent magnet (PM) and variable reluctance (VR) techniques to achieve maximum power in small package sizes.

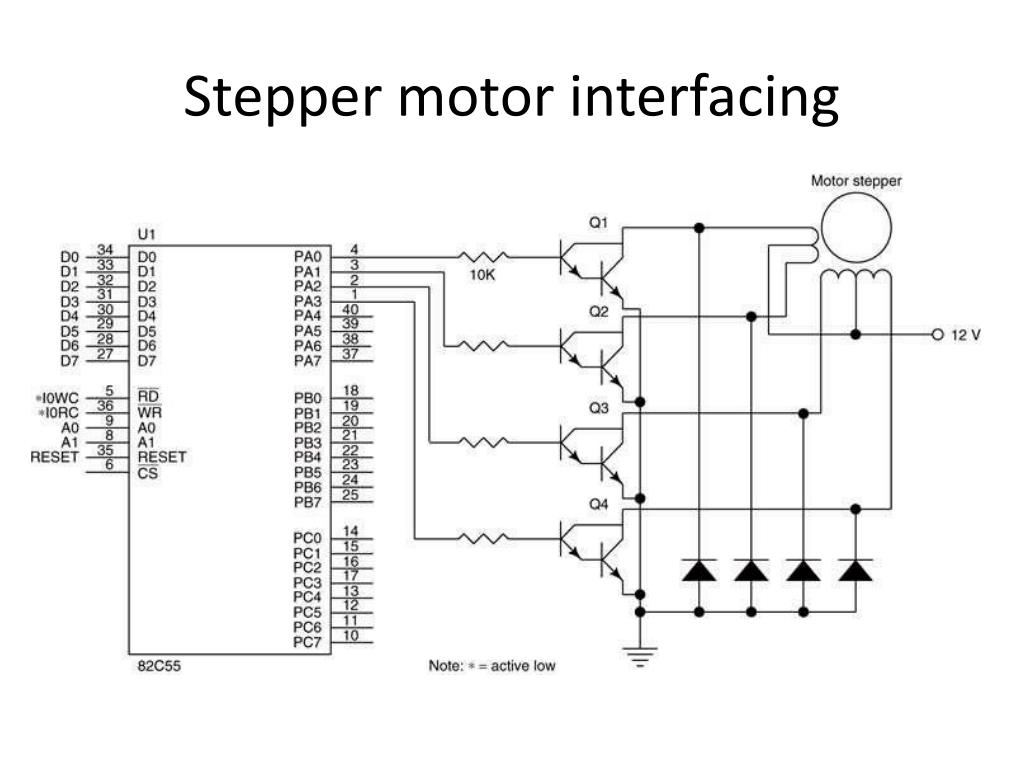
The most popular type of motor is the [hybrid stepper motor](https://www.elprocus.com/what-is-hybrid-stepper-motor-working-its-applications/) because it gives a good performance as compared with a permanent magnet rotor in terms of speed, step resolution, and holding torque. But, this type of stepper motor is expensive as compared with permanent magnet stepper motors. This motor combines the features of both the permanent magnet and variable reluctance stepper motors. These motors are used where less stepping angle is required like 1.5, 1.8 & 2.5 degrees.

### **Applications**

The **applications of stepper motor** include the following.

1. **Industrial Machines** – Stepper motors are used in automotive gauges and machine tooling automated production equipment.
2. **Security**– new surveillance products for the security industry.
3. **Medical** – Stepper motors are used inside medical scanners, samplers, and also found inside digital dental photography, fluid pumps, respirators, and blood analysis machinery.
4. **Consumer Electronics** – Stepper motors in cameras for automatic digital camera focus and zoom functions.

**PIN DIAGRAM:**



**1.** **Stepper Motor in Clock-wise Direction:**

**Aim:**

**To construct the Interfacing of Stepper Motor in clock wise**

**Tools Required:**

**8086 Emulator**

Apparatus:

1. ADS-SDA-86-STA kit

2. Stepper motor interface card

3. 1 Amp Power Supply

4. Stepper Motor

5. Adapter, Keyboard, Cables, Connecting Wires Etc...

**Procedure:**

1. Connect 8086 kit PC using RS232 cable.

2. Connect Power supply to 8086 kit

3. Connect 1Amp Power Supply to the Stepper Motor

4. Connect 8255 to CN4 of 8086 using 26 pin bus.

5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.

6. Change dip switch into 1 & 5on, once reset 8086 kit.

7. Go to file →Download hex file

8. Keep the DIP switch in 1 & 7 on (8086kit)

9. G-4000(on kit keyboard), now the stepper motor will be rotating in clockwise direction

**Program Code:**

**org 100h**

**#START=STEPPER\_MOTOR.EXE#**

**JMP START**

**DATACW DB 0000\_0011B**

**DB 0000\_0110B**

**DB 0000\_1100B**

**DB 0000\_1001B**

**START:**

**MOV BX, offset DATACW**

**MOV SI, 0h**

**NEXT\_STEP:**

**WAIT: IN AL,07H**

**TEST AL,10000000b**

**JZ WAIT**

**MOV AL, [BX][SI]**

**OUT 7, AL**

**INC SI**

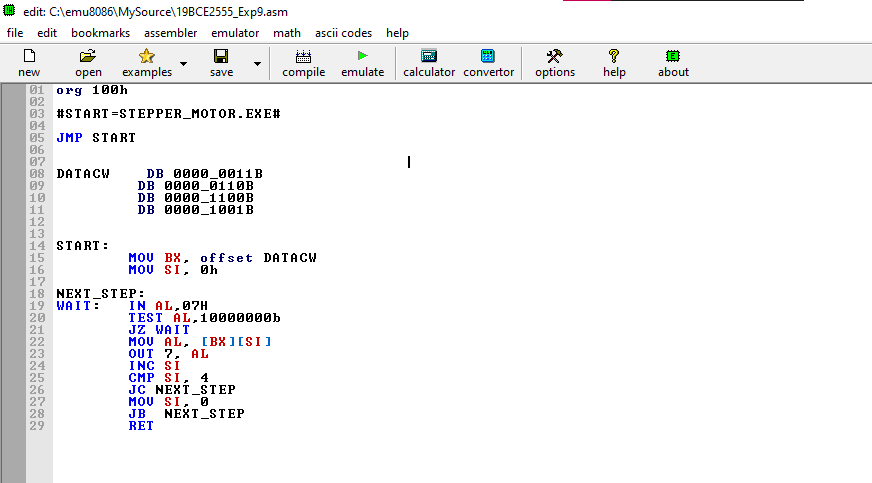
**CMP SI, 4**

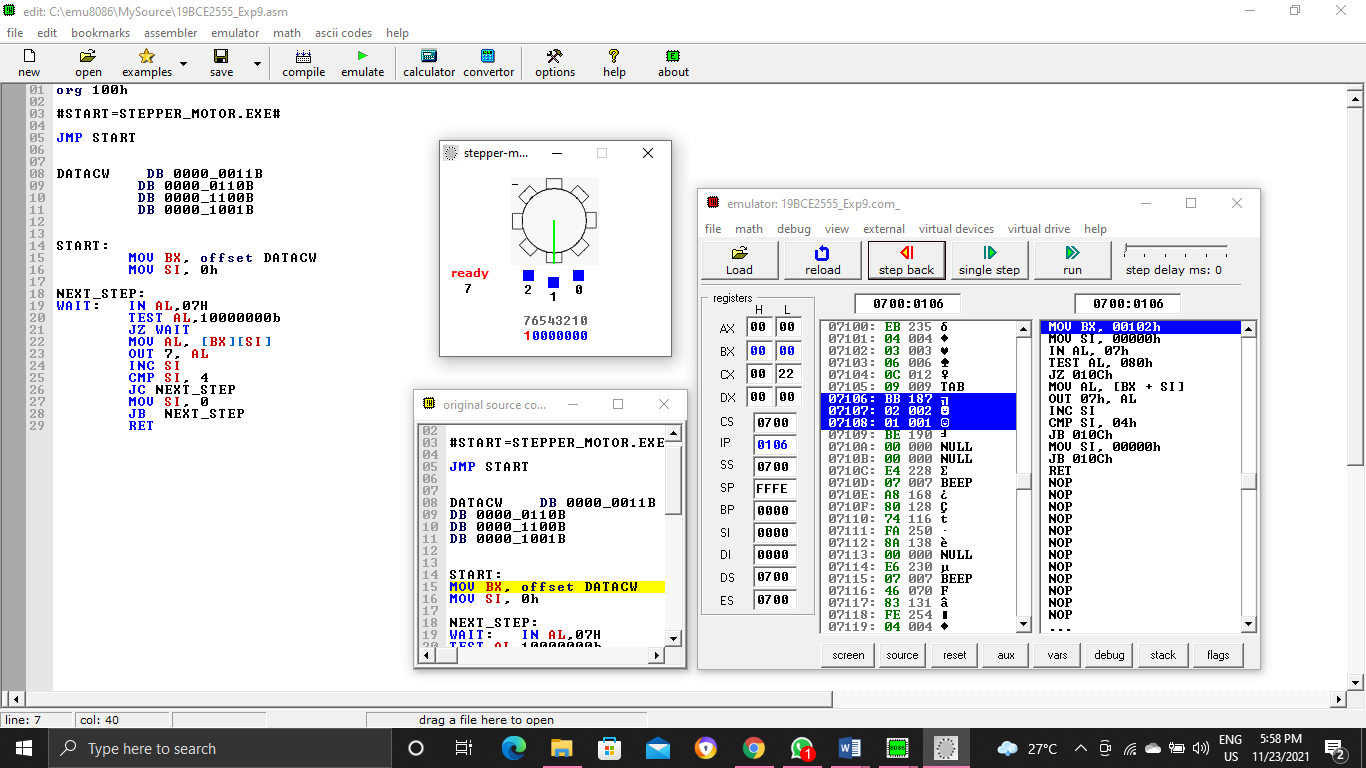
**JC NEXT\_STEP**

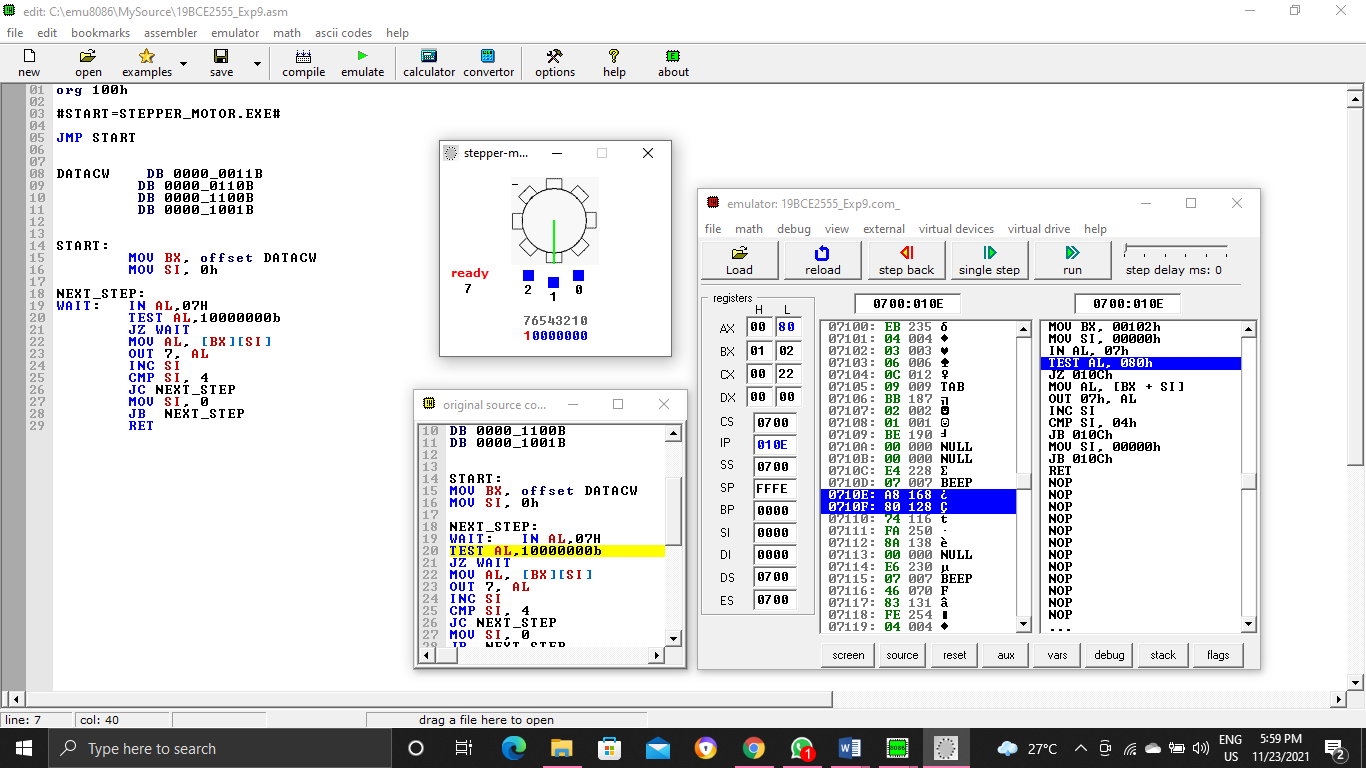
**MOV SI, 0**

**JB NEXT\_STEP**

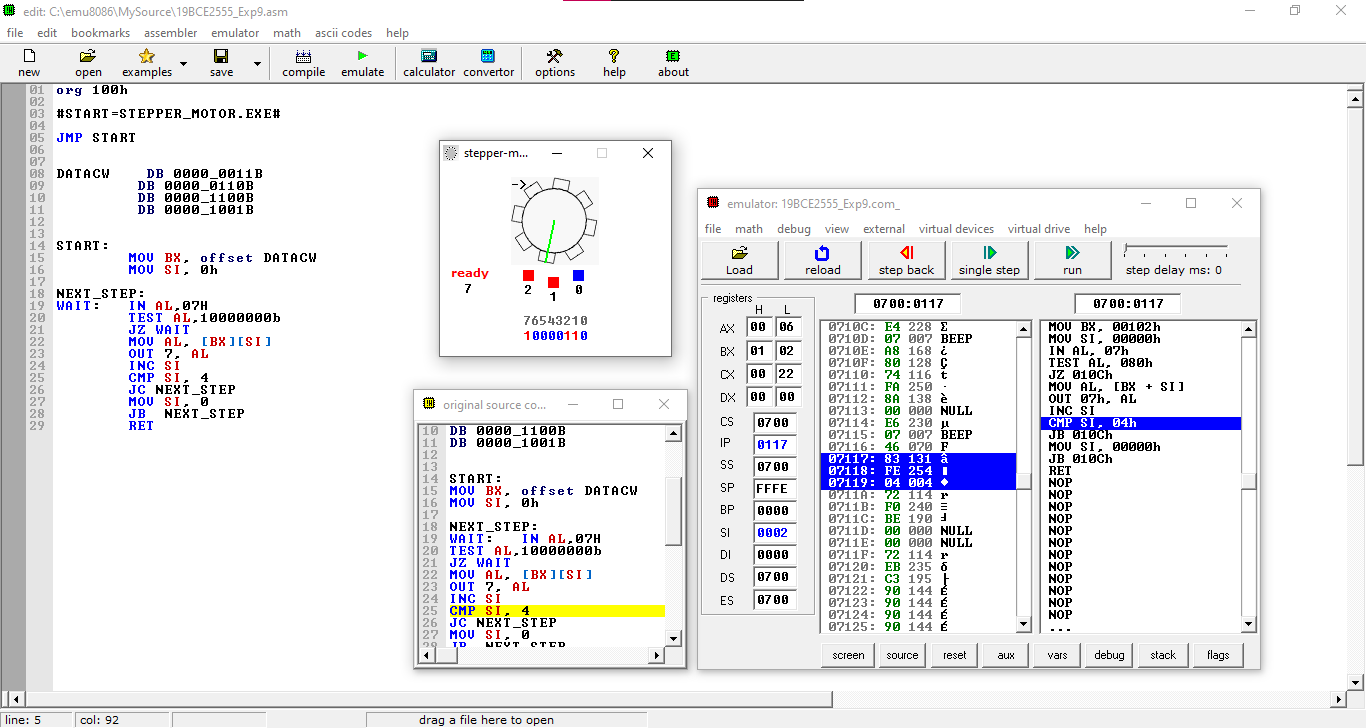
**RET**

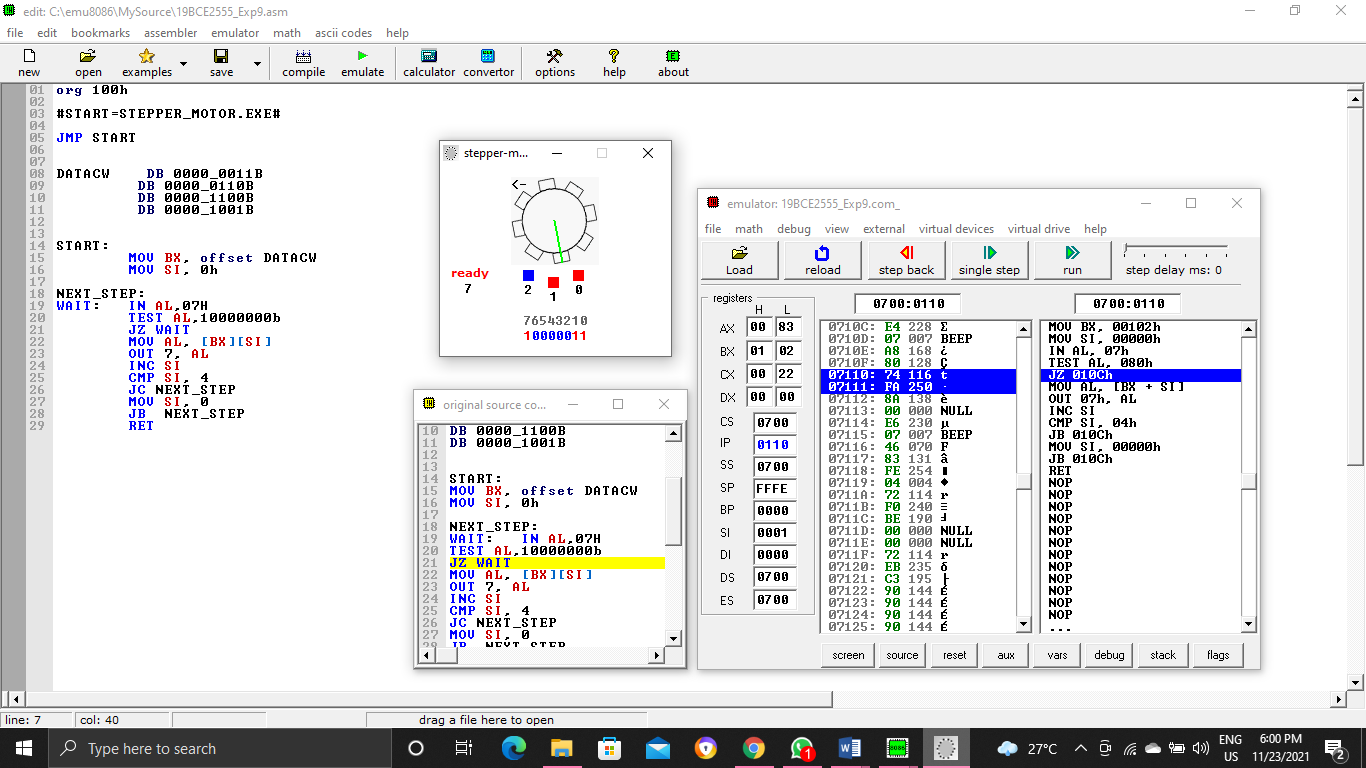
****

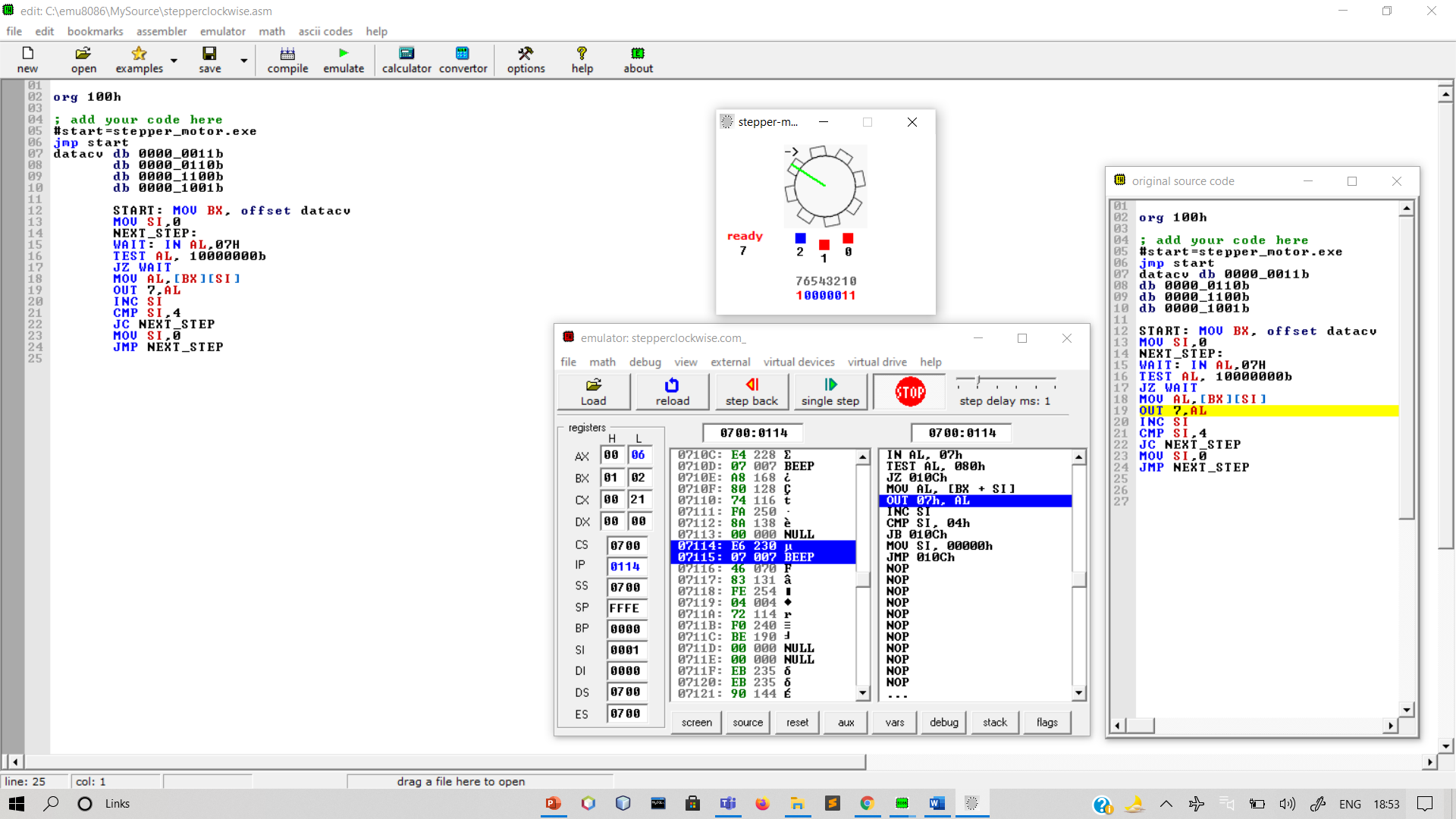




**Output:**







Results and Inference:

We can see that the Stepper Motor is running in Clock Wise Direction

**2.** **Stepper Motor in AntiClock-wise Direction:**

**Aim:**

**To construct the Interfacing of Stepper Motor in Anti clock wise Direction**

**Tools Required:**

**8086 Emulator**

**Apparatus:**

**1. ADS-SDA-86-STA kit**

**2. Stepper motor interface card**

**3. 1 Amp Power Supply.**

**4. Stepper Motor**

**5. Adapter, Keyboard, Cables, Connecting Wires Etc...**

**Procedure :**

**1. Connect 8086 kit PC using RS232 cable.**

**2. Connect Power supply to 8086 kit**

**3. Connect 1Amp Power Supply to the Stepper Motor**

**4. Connect 8255 to CN4 of 8086 using 26 pin bus.**

**5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.**

**6. Change dip switch into 1 & 5on, once reset 8086 kit.**

**7. Go to file →Download hex file**

**8. Keep the DIP switch in 1 & 7 on (8086kit)**

**9. G-4000(on kit keyboard), now the stepper motor will be rotating in clockwise direction**

**Program Code:**

**org 100h**

**#START=STEPPER\_MOTOR.EXE#**

**JMP START**

**DATACW DB 0011\_0011B**

**DB 0000\_1001B**

**DB 0000\_1100B**

**DB 0000\_0110B**

**START:**

**MOV BX, offset DATACW**

**MOV SI, 0h**

**NEXT\_STEP:**

**WAIT: IN AL,07H**

**TEST AL,10000000b**

**JZ WAIT**

**MOV AL, [BX][SI]**

**OUT 7, AL**

**INC SI**

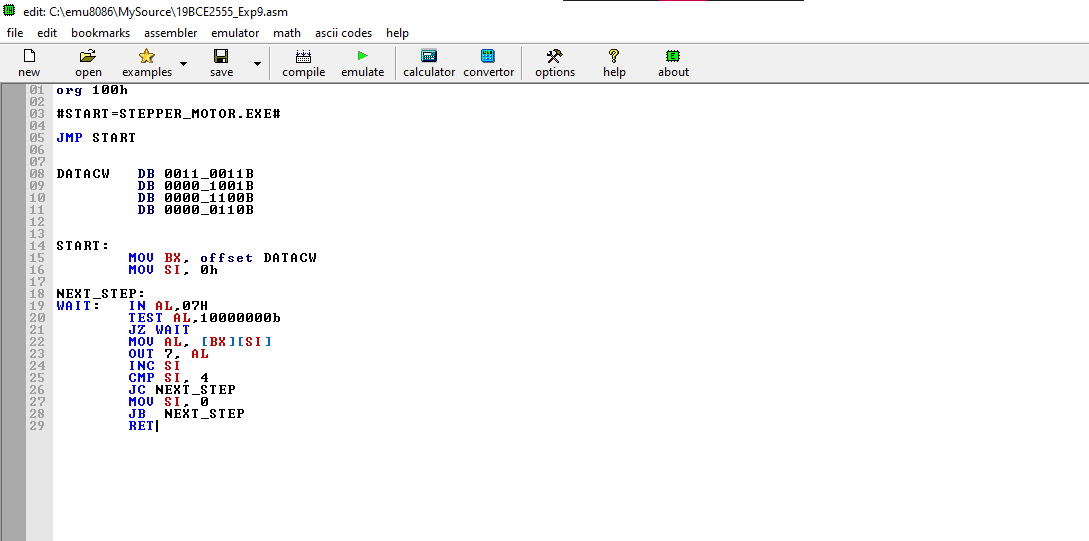
**CMP SI, 4**

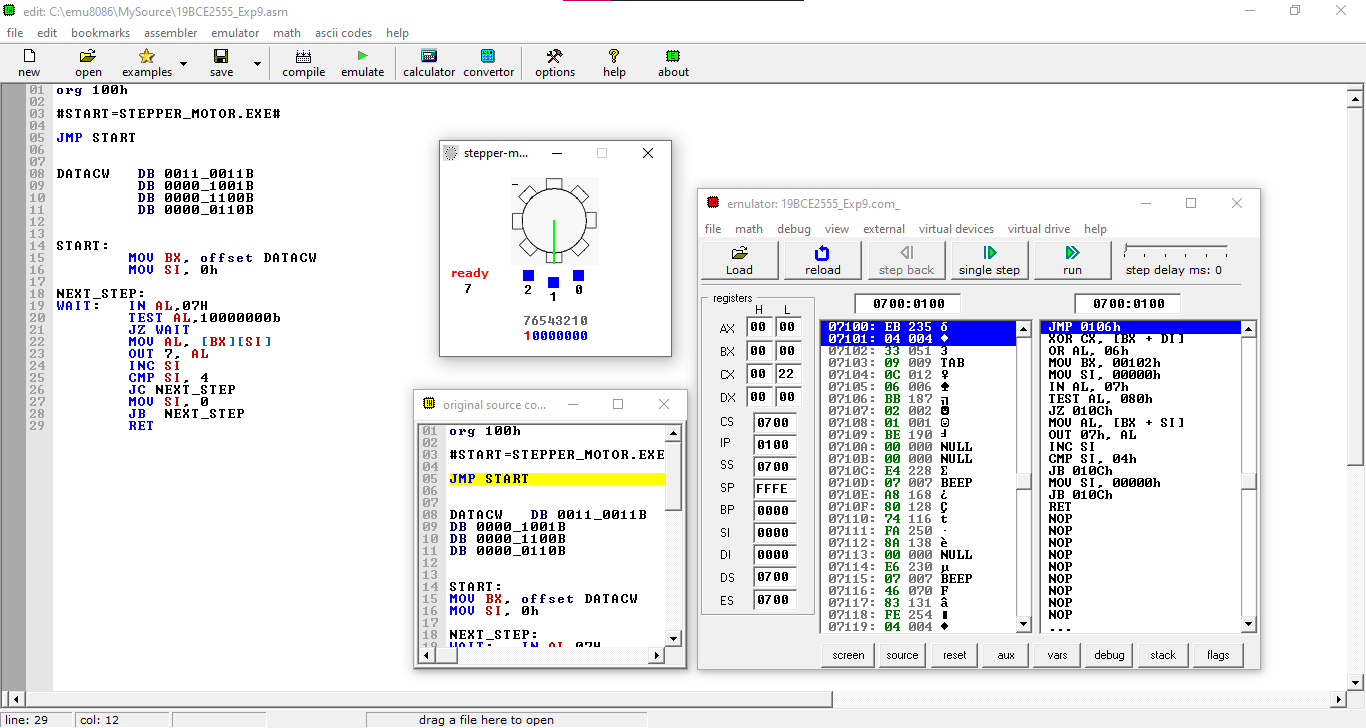
**JC NEXT\_STEP**

**MOV SI, 0**

**JB NEXT\_STEP**

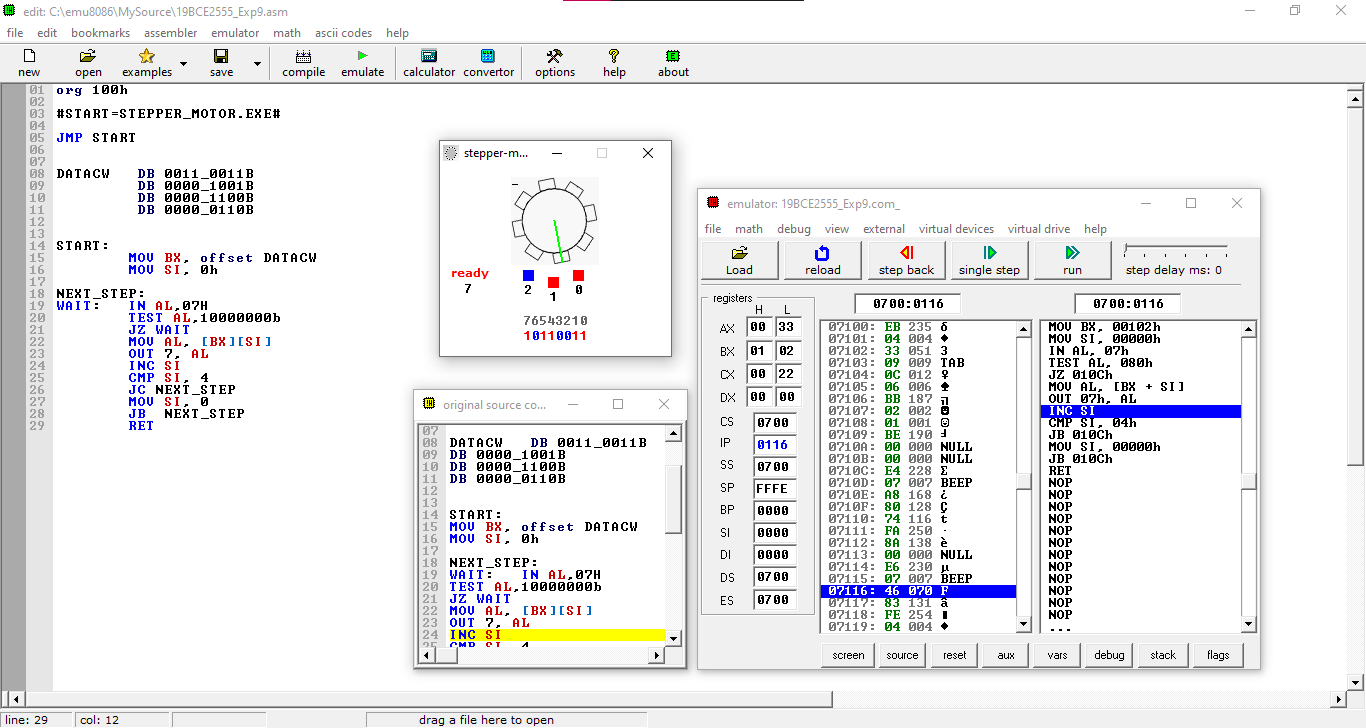
**RET**

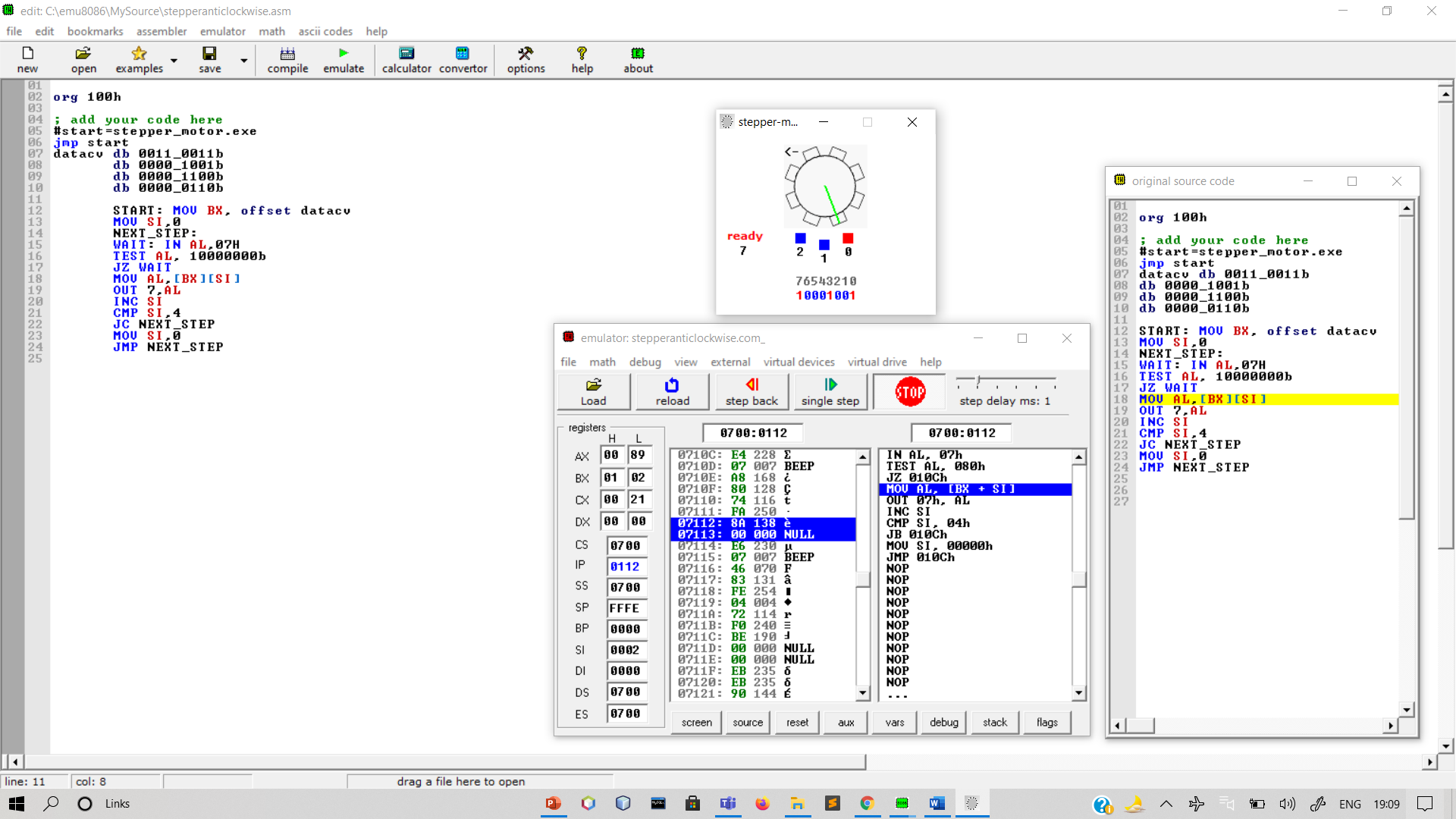
****

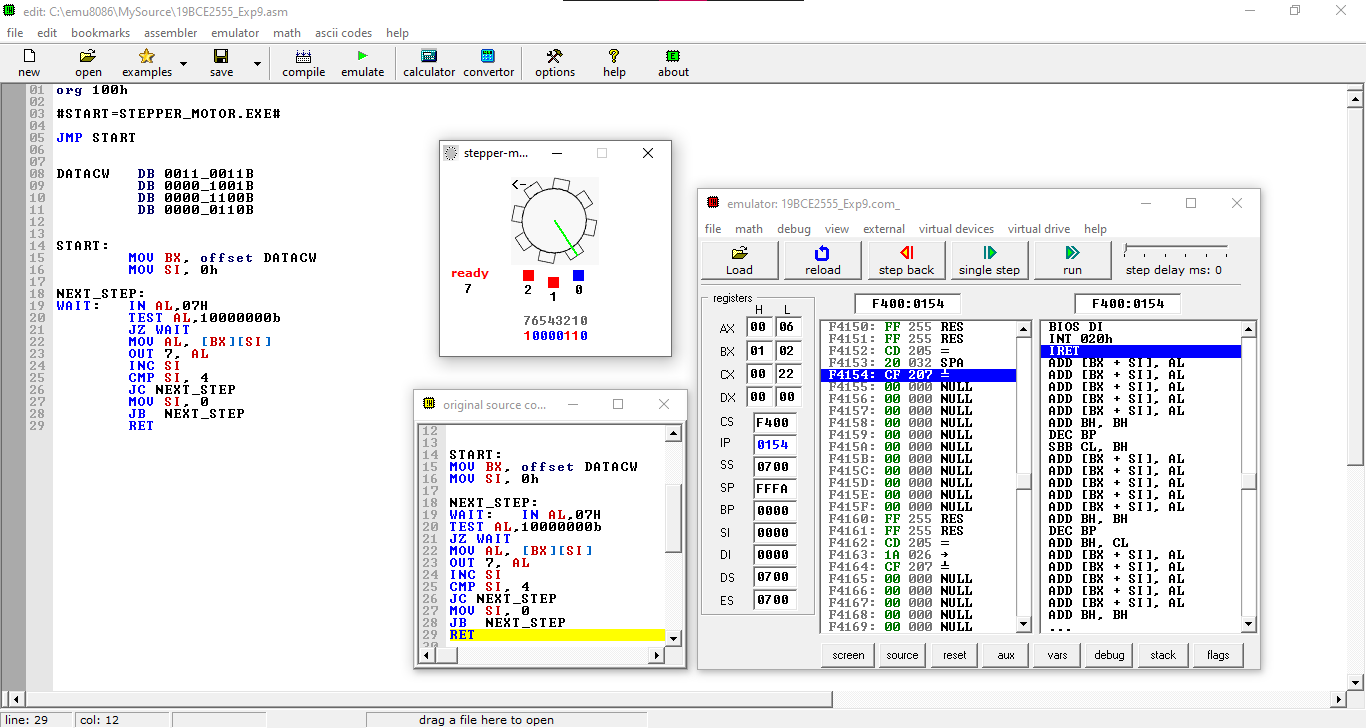
****

****

**Output:**







**Results and Inference:**

**We can see that the Stepper Motor is running in Anti Clock Wise Direction**

**3.** **Stepper Motor in Clock-wise Direction with a Random Delay:**

**Aim:**

**To construct the Interfacing of Stepper Motor in Clock wise Direction with a Random Delay.**

**Tools Required:**

**8086 Emulator**

**Program Code:**

**org 100h**

**#START=STEPPER\_MOTOR.EXE#**

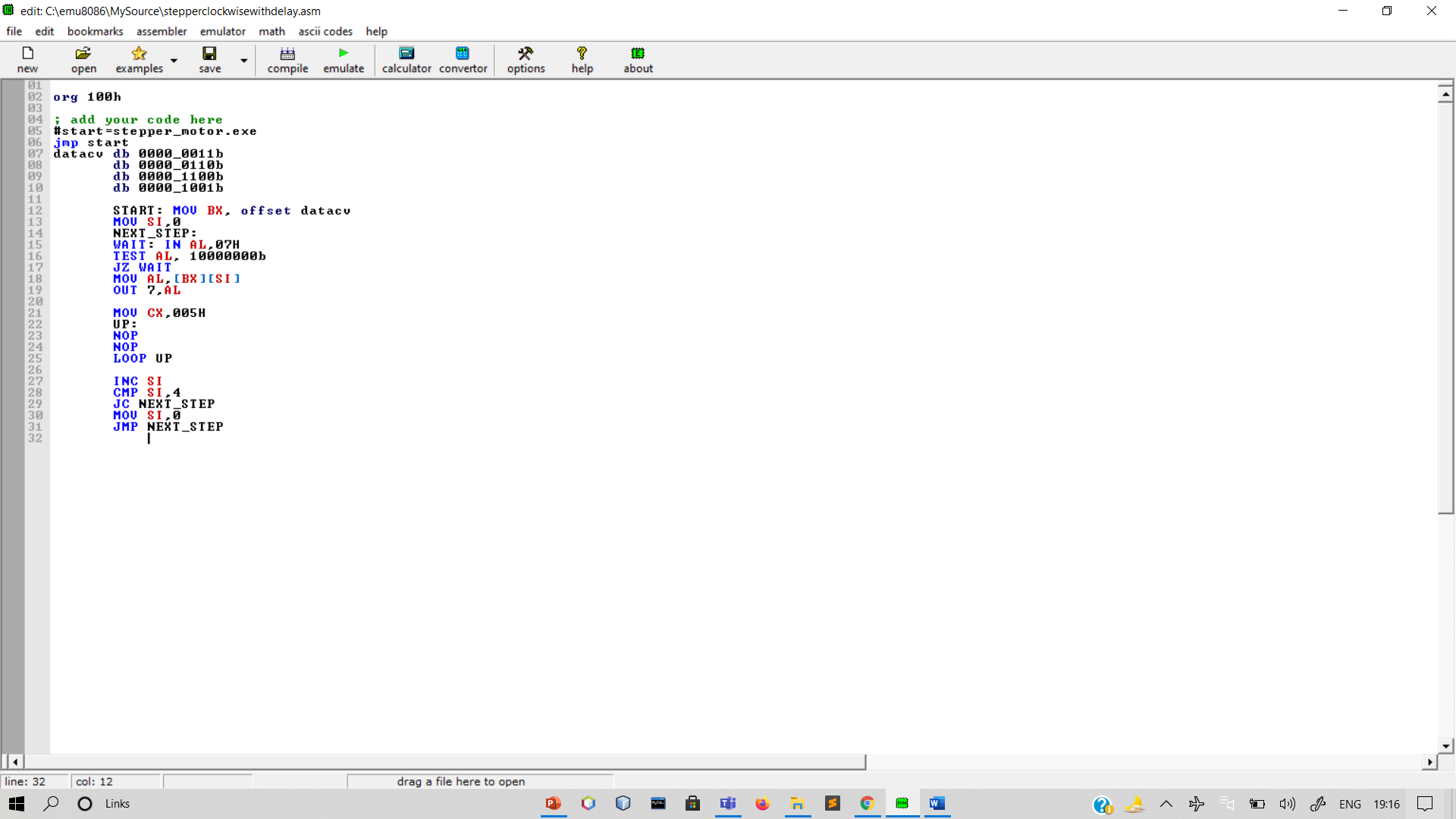
**JMP START**

**DATACW DB 0011\_0011B**

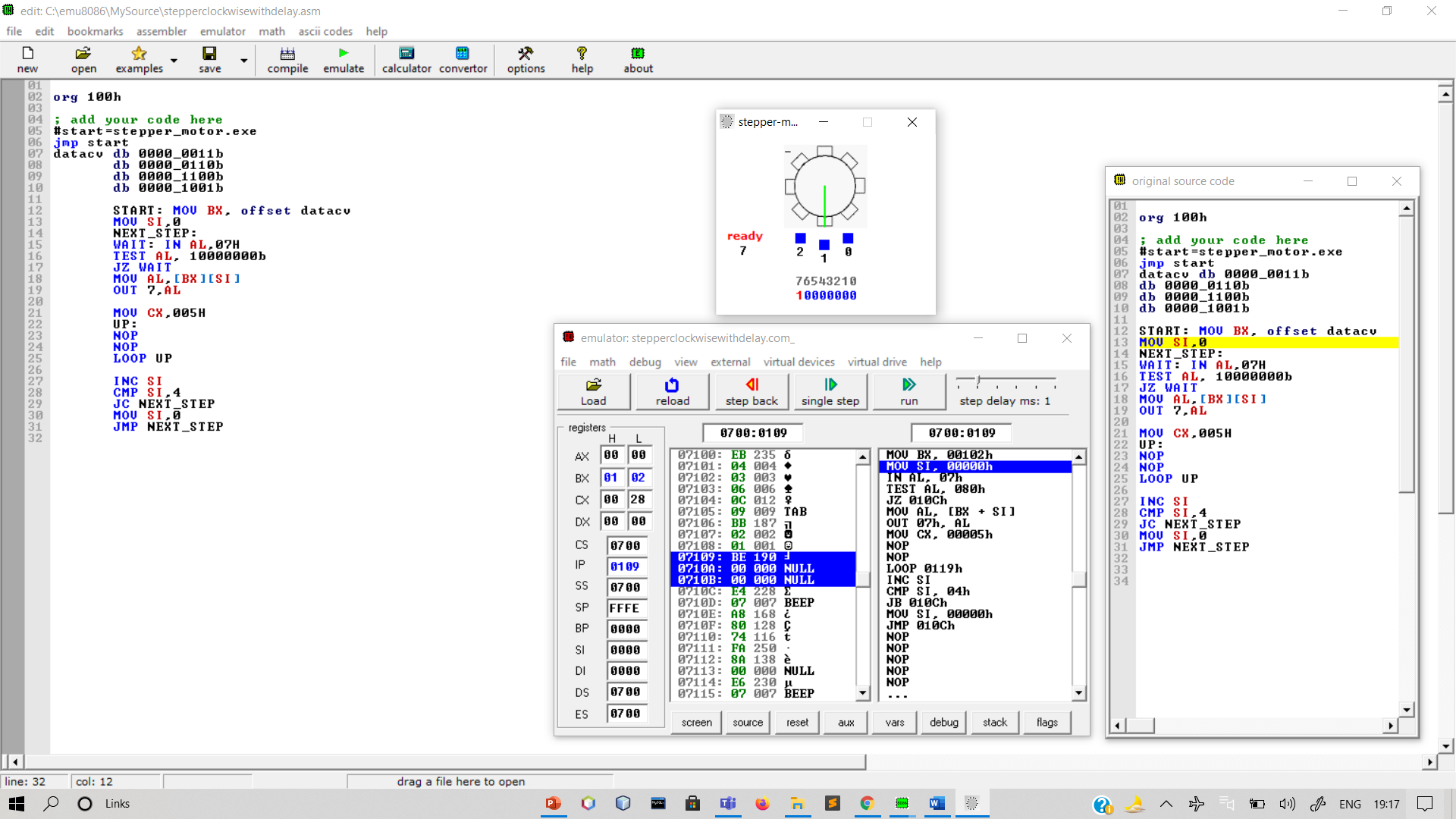
**DB 0000\_1001B**

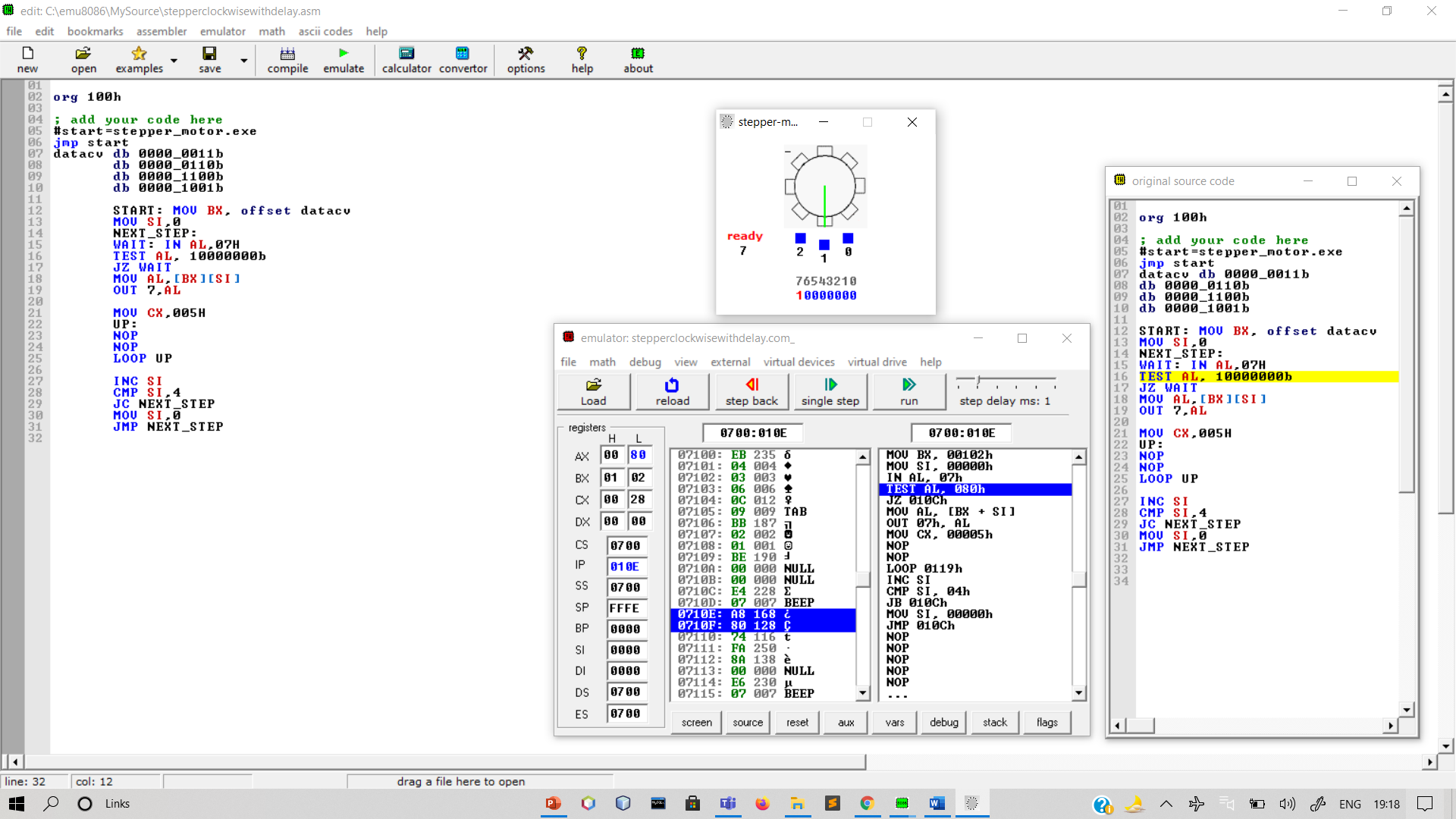
**DB 0000\_1100B**

**DB 0000\_0110B**



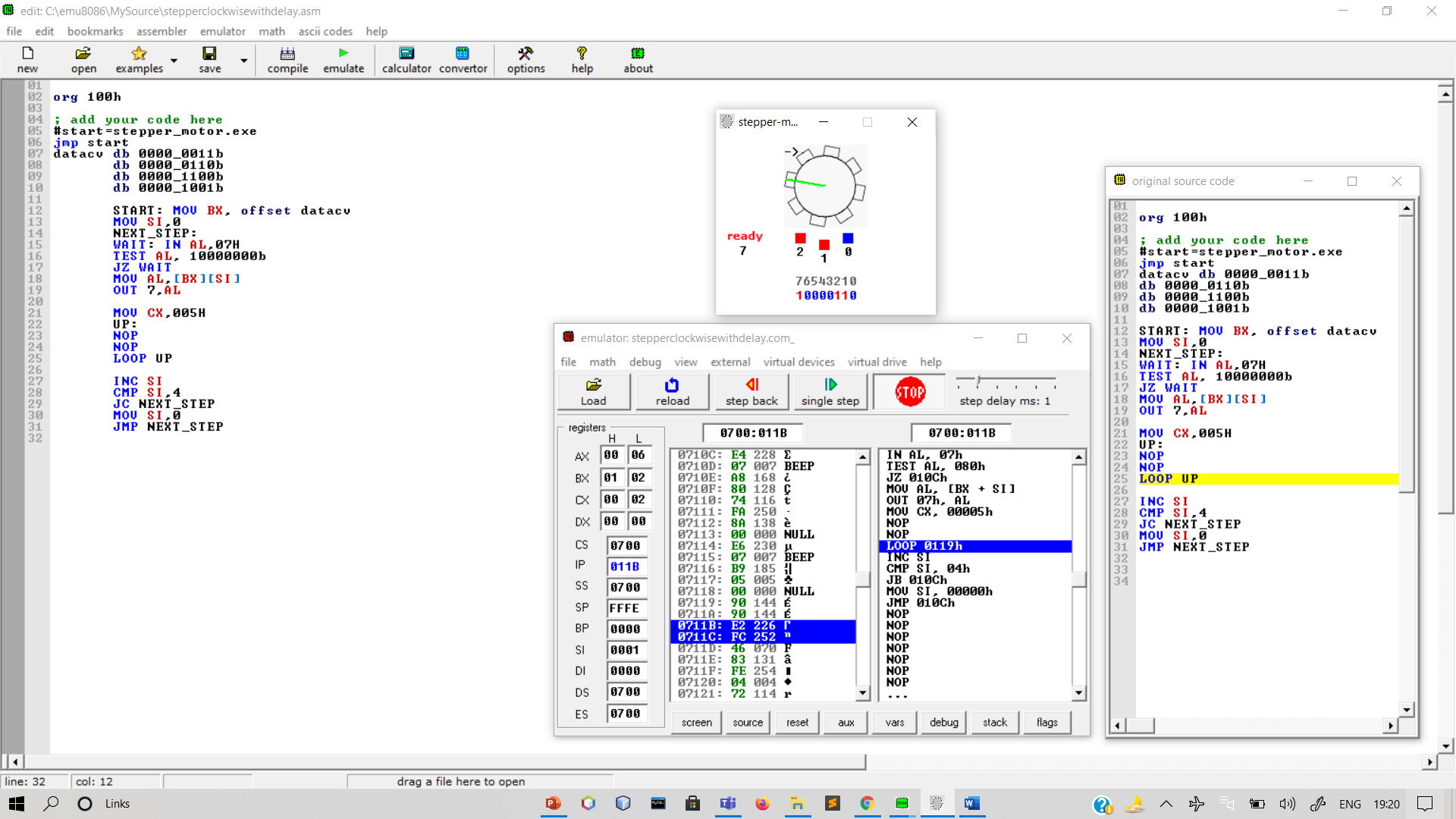






**Output:**





Results and Inference:

We can see that the Stepper Motor is running in Clock Wise Direction with a Random Delay.